Denoising Point Clouds in Latent Space via Graph Convolution and Invertible Neural Network

Aihua Mao[1†*], Biao Yan[1†], Zijing Ma[1], Ying He[2]
[1]School of Computer Science and Engineering, South China University of Technology
[2]School of Computer Science and Engineering, Nanyang Technological University
ahmao@scut.edu.cn,{202221044459,202130441399}@mail.scut.edu.cn,
yhe@ntu.edu.sg

Abstract

Point clouds frequently contain noise and outliers, presenting obstacles for downstream applications. In this work, we introduce a novel denoising method for point clouds. By leveraging the latent space, we explicitly uncover noise components, allowing for the extraction of a clean latent code. This, in turn, facilitates the restoration of clean points via inverse transformation. A key component in our network is a new multi-level graph convolution network for capturing rich geometric structural features at various scales from local to global. These features are then integrated into the invertible neural network which bijectively maps the latent space, to guide the noise disentanglement process. Additionally, we employ an invertible monotone operator to model the transformation process, effectively enhancing the representation of integrated geometric features. This enhancement allows our network to precisely differentiate between noise factors and the intrinsic clean points in the latent code by projecting them onto separate channels. Both qualitative and quantitative evaluations demonstrate that our method outperforms state-of-the-art methods at various noise levels. The source code is available at https://github.com/yanbiao1/PD-LTS.

1. Introduction

Point clouds are a popular data format in 3D vision tasks due to their adaptability and efficiency. They are extensively employed across a range of fields, including autonomous driving, robotics, topographical mapping, and the development of 3D urban models. These point clouds are generally captured using LiDAR or structural light systems. However, the data collected often includes noise and anomalies that can impede subsequent processes such as 3D reconstruction and rendering. As a result, denoising is a critical problem in the management and utilization of 3D point cloud data.

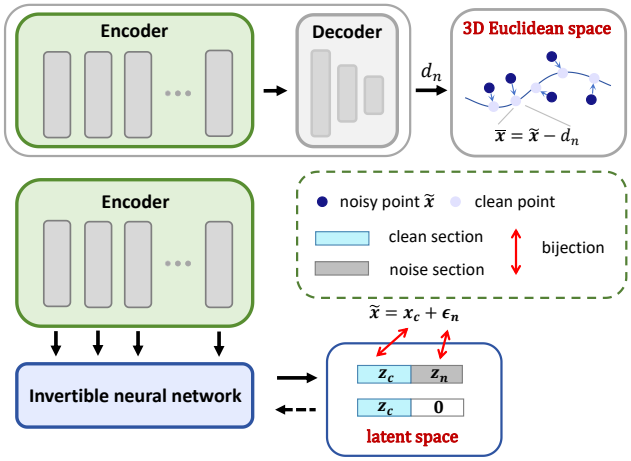

Figure 1. **Top:** Traditional denoising approaches often utilize an encoder-decoder architecture, focusing on regressing geometric features to predict 3D displacements d_n. **Bottom:** Our method employs an invertible neural network, integrating geometric features to achieve noise disentanglement in the latent space.

Traditional point cloud denoising methods [2, 3, 10, 12, 18, 19, 27, 55] typically rely on normal estimation or geometric prior information and are often unreliable in real-world scenarios. A common issue with these methods is their tendency to over-smooth surfaces, resulting in a loss of geometric details.

In response to these limitations, there has been a shift towards data-driven approaches. Pioneering deep learning methods for point cloud analysis, such as PointNet [41], PointNet++ [42], and DGCNN [45], have demonstrated their ability to capture rich shape structures. Leveraging these foundational techniques, a variety of deep learning-based methods have been developed [9, 32, 33, 35, 39, 43,

*Corresponding author. †Equal contribution.

52], offering more effective solutions for point cloud denoising.

From the perspective of denoising space, we categorize deep learning methods for point cloud denoising into two types: 1) **Denoising in 3D Euclidean space** [9, 33, 39, 43, 52]. These methods reposition noise points to clean locations in 3D space. To achieve this, they regress high-dimensional features, which are rich in geometric information, to 3D displacement. However, such regression may suppress the feature expression and often leads to issues such as patch shrinkage or surface collapse [43]. 2) **Denoising in high-dimensional feature space** [32, 35]. DMRDenoise[32] resamples clean points on an underlying 2-manifold surface, but this can lead to the loss of fine geometric details that occurs during the resampling process. PD-flow [35] employs normalizing flow [23] to establish a bijective mapping between distributions, but it faces challenges due to the insufficient local feature extraction and the need for dimension partitioning in its affine coupling layer.

This paper presents a novel point cloud denoising method that operates within the high-dimensional latent space, representing a departure from traditional displacement-based approaches. Our method is underpinned by an invertible neural network. Inspired by [1], we employ invertible monotone operators from functional analysis to model our invertible neural network. This approach, enforcing Lipschitz constraints [5], facilitates a bijective mapping for encoding process, while preserving a flexible and expressive architectural design. To augment the capabilities of our invertible neural network, particularly in terms of broadening its receptive fields and enhancing point cloud shape perception, we design a multi-level graph convolution (MLGC) module. This module utilizes EdgeConv [45] to capture local shape structures effectively. By densely connecting the multi-level features from each EdgeConv layer, we facilitate the progressive accumulation of contextual semantic information. The resulting enriched feature set is then integrated into the invertible neural network.

Our approach introduces a prior latent space where noise and intrinsic clean points are distinctly separated into different dimensions. We leverage the integrated MLGC features within the invertible neural network to parameterize the bijection mappings between the Euclidean space and the prior latent space. This strategy effectively uncovers the noise-convolved distribution of noisy point clouds. Upon isolating the noise in the latent space, we mask the noise-associated dimensions, leading to the generation of clean latent codes. Inversely transforming these codes, we obtain a noise-free point cloud.

It is worth noting that our method is fundamentally different from displacement-based methods, which typically depend on feature regression. By utilizing the geometric features from MLGC, our approach enables a more expressive representation within the high-dimensional latent space offered by the invertible neural network (see Fig. 1). Additionally, our denoising strategy focuses on the explicit extraction of noise within the latent space, rather than direct manipulation of points in the Euclidean space. This focus on latent space processing not only enhances the effectiveness of denoising but also preserves the structural integrity of the original point cloud data.

We make the following contributions in the paper:

- We propose a point cloud denoising algorithm, based on a prior latent space where noise components are disentangled. By modeling bijection mappings, we indirectly denoise point cloud of Euclidean space by masking the uncovered noise in the prior latent space.
- For bijection mappings, we introduce invertible monotone operator to model a powerful and lightweight invertible neural network, which has an unconstrained and expressive architecture by enforcing a Lipschitz constraint.
- To improve the capture and representation of point cloud shape structure, we design a multi-level graph convolution module by employing EdgeConv and dense connections, and integrate these features to invertible neural network for further expression in high-dimenstional feature space.

2. Related Work

Traditional methods. The bilateral filtering-based methods [10, 53] aggregate neighbor points and normal information to design specialized weighted functions, and combining them with normal to estimate displacement. Previous work [19, 29, 55] leverage the parameters calculated from local normals and coordinates to iteratively update the normals to guide points movement. The graph-based methods characterize the underlying geometric structure of point clouds using the features of adjacency graphs. Some researchers [15, 21] constructed k-nn graph between points and estimating a tangent plane to locally approximate the underlying manifold, while others [12, 20, 51] constructed graph between patches and employ the Graph Laplacian Regularizer (GLR). Among the optimization technique, Moving Least Squares(MLS)-based methods [3, 16, 26, 37] estimate the underlying surface by computing explicit parameters like n-order polynomial [6]. Similarly, Locally Optimal Projection(LOP)-based methods estimate the underlying surface by producing point sets. Recent works [31, 47] separately proposed estimating MLS and LOP with anisotropic weights by asymmetric directional neighborhood to avoid common over-smooth problem. The sparse-based methods focus on sparse non-smooth surface feature and reconstruct sparse surface normals by $l1$ regularization [25, 36] or dictionary learning [11]. The non-local-based methods [46, 54] consider the relationship between non-

local patches and combine similar patches to form a patch matrix, then employ low rank recovery algorithm to denoise. In general, these methods mentioned above are not driven by data, which can result in insufficient assumptions or a demands for geometric attributes.

Deep learning based methods. Pointcleannet [43] first leveraged PointNet[41] architecture combined with residual connections to predict displacement, and move noise points iteratively to eliminate residual noise. Similarly, Pointfilter [52] employes PointNet [41] to capture patch structure, and it introduced normals in the loss function to guide the displacement prediction. EC-Net [49] proposes an edge-aware joint loss function to encourage good edge perception for point cloud consolidation. Pistilli et al. [39] proposes GPDNet to design a special graph convolution neural network which predict adaptive weights to aggregate local edge features. Luo et al. [33] proposed ScoreDenoise to model the distribution of noise-clean convolution distribution, and regarded the underlying clean surface as the area with the highest probability density, then score matching technique was used to learn the gradient-log of convolution distribution to perform gradient ascent for denoising. DMRDenoise [32] proposed resampling based method and attempted to reconstruct local underlying surface for the sampling of clean points. This resampling method, however, struggled with fine local shape preservation. Mao et al. proposed PD-flow [35] to model the distribution of noisy point cloud using normalizing flow [23], and applied consistency loss in latent space to filter noise component and obtain clean latent code. However, it is difficult to grasp shape structure for denoising with just a single normalizing flow. Recently, Edirimuni et al. [9] proposed IterativePFN, which utilize dynamic graph convolution [45] to design stacked denoising modules, and simulated the iterative filtering during training process to encourage faster convergence. These works inspire us to further express geometric features from graph convolution in high-dimensional latent space for point cloud denoising.

Normalizing flow for point cloud analysis. Normalizing flow was initially proposed for image generation and density estimation tasks [13, 14, 23]. Owning to the invertible transformation and distribution learning ablility, it is increasingly applied to point cloud downstream tasks in recent years. For point cloud generation, Pointflow [48] first utilizes continuous flow [7, 17] to model the distribution of point clouds by mapping them to latent space with Gaussian distribution, and generates point clouds via the inverse transforamtion. Klokov et al. [24] proposes DPF-Net to model discrete flow for point cloud generation, which has better inference efficiency compared to Pointflow [48]. Further, Go-with-the-flow [40] unsupervises learns to split a point cloud into several parts, and separately learns the distribution of them using discrete flow. Soft-flow [22] bridges the gap between 3D Gaussian distribution and the underlying 2D manifold distribution of point clouds by adding noise. On other tasks, PU-flow [34] and PD-flow [35] both utilize Glow [23] to model invertible neural network for the upsampling and denoising tasks. In recent years, Behrmann ta al. [5] proposed invertible residual neural network based on Lipschitz constraint function and Banach fixed point theorem, which is a new family of normalizing flow. Further, based on the Lipschitz network and invertible theory, many research works [1, 8, 30, 38] improved network architecture and propose new invertible expressions to achieve better transformation ability. Inspired by these works, we introduce the invertible monotone operator from [1, 4] as the theoretical fundamentals of our invertible transformation to bijectively connect a prior latent space.

3. Method

3.1. Motivation

As mentioned in Sec. 1, we category deep learning denoising methods into two types. The first type is denoising in 3D Euclidean space by directly moving points, also known as displacement estimation methods, whose general process is shown in Fig. 1 (top). They regress geometric features to 3D displacement d_i for noisy point $\widetilde{x}_i$, and the denoising objective is $\bar{x}_i = \widetilde{x}_i - d_i$. This non-trivial process limits the feature expressiveness, because empirically grasping elusive shape structures always requires high-dimensional feature space. Inspired by the second type which denoising in high-dimensional feature space, we aim to leverage latent space to uncover noise component and extract clean latent code. To intuitively describe this process, we analogize the denoising problem as the scenario of projecting outlier points onto a plane(see Fig. 2). Looking at a noisy point cloud in 3D Euclidean space can be analogized as the 2D vertical view of the plane, where the positional relationship between the points and the plane is unclear, since noisy components are hidden in 3D coordinates in the form of addition: $\widetilde{x}_i = x_i + \epsilon$. Consequently, we could change the visual angle to obtain explicit positional relationship from the front view of 3D. This process can be analogized as finding a representation of point clouds in high-dimensional latent space, where the noise components are explicitly disentangled and uncovered. As shown in bottom-right of Fig. 2, the noise component and intrinsic clean points are disentangled into different channels of latent code: $\widetilde{z}_i = [z_c, z_n]$, then we can set noise channels $z_n = 0$ to obtain the clean latent code. Through forming a bijection mapping, setting $z_n = 0$ in the latent space could exactly reflect a bijective change: $\bar{x}_i = \widetilde{x}_i - d_i$ in the coordinate space. As a result, our method's training objective is to match the latent space where noise can be uncovered. To this end, we first model an invertible neural network based on the theory of

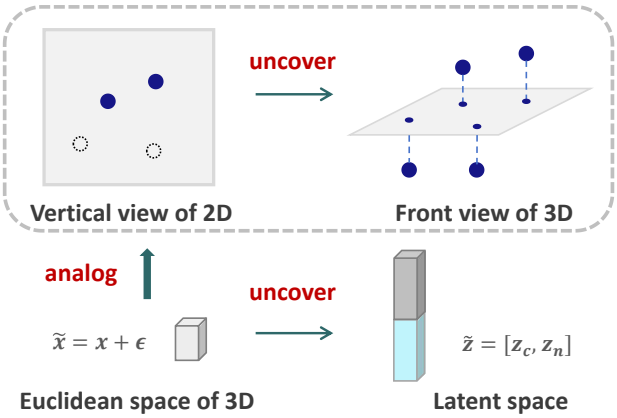

Figure 2. Visual analog of uncovering noise components in the latent space.

invertible monotone operator, then we design a multi-level graph convolution(MLGC) module to extract shape structural features and integrate them into the invertible neural network to guide the noise disentanglement process. The overall network architecture is shown in Fig. 3.

3.2. Invertible Neural Network

In the field of partial differential equations and functional analysis, the monotone operator has a rich function space and invertible properties [1, 4]. Inspired by this, we introduce an invertible neural network based on monotone operator [1] to model bijection mappings.

3.2.1 Invertible Transformation

Let $G : \mathbb{R}^n \to \mathbb{R}^n$ be a Lipschitz-continuous function with Lipschitz constant $L < 1$. The forward transformation based on G is defined as the following function $F : \mathbb{R}^n \to \mathbb{R}^n$

$$F(x) = \left(\frac{\mathrm{Id} + G}{2}\right)^{-1}(x) - x, \tag{1}$$

where Id denotes the identity function. According to the theory of invertible monotone operators [1, 4], F is monotone $\Leftrightarrow$ G is 1-Lipschitz($L < 1$), and strictly monotone continuous function $F : \mathbb{R}^n \to \mathbb{R}^n$ is invertible. Thus, for Eq. (1), we have the inverse formulation $F^{-1}(x)$:

$$F^{-1}(x) = \left(\frac{\mathrm{Id} - G}{2}\right)^{-1}(x) - x \tag{2}$$

We refer readers to [1] for the proof and more details about the invertible transformation. Further, we note that according to the Banach fixed point theorem [5], $(\mathrm{Id} + G)$ has an unique inverse, to calculate the result of the inverse function $y = (\mathrm{Id} + G)^{-1}(x)$ in Eq. (1), we iterate $y = x - G(y)$ a certain number of steps to converge to y, which satisfies the accuracy requirement, and the same goes for Eq. (2).

3.2.2 Modeling Invertible Neural Network

To implement the 1-Lipschitz function G in Eq. (1) with neural network blocks, we compose linear mappings with 1-Lipschitz activation function. For a simple case: let $g_\theta(x) = \psi(Wx + b)$, where ψ is 1-Lipschitz activation function, we have:

$$Lip(g_\theta) < 1, \text{if } \|W\|_2 < 1 \tag{3}$$

where $\|\cdot\|_2$ denotes the spectral norm. As a result, we can perform spectral regularization on W to enforce $\|W\|_2 < 1$. Then, we compose several blocks of g_θ satisfying $Lip(g_\theta) < 1$ to model the 1-Lipschitz function G in Eq. (1), which has an unconstrained and expressive architecture, as it does not require dimension partitioning compared to INN based on affine coupling layers [23, 35], this enables our INN to be effectively combined with multi-level graph convolution network, to implement invertible encoding process for denoising.

3.3. Multi-level Graph Convolution

DGCNN [45] proposes a dynamic graph convolution framework to capture rich geometric properties of point clouds. It dynamically constructs directed graph between points and extract edge features by EdgeConv, which recover the involved topology and reflects the relationships between neighboring points. Therefore, we introduce the framework of DGCNN for geometric feature extraction, and add dense connections [28] between each EdgeConv layers to capture contextual semantic information. To be specific, we initially construct k-nearest neighbor (k-NN) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ based on the 3D Euclidean distance between points, where $\mathcal{V} = \{1, ..., n\}$ and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ are the vertices and edges respectively, then we fix the graph in each subsequent EdgeConv layers. Given a point cloud $\mathbf{X} = \{x_i \in \mathbb{R}^3\}_{i=1}^N$, the initial feature vector is produced as: $\mathbf{h}_i^{(0)} = \sum_{j:(i,j)\in\mathcal{E}} \mathrm{MLP}(x_i \| x_i - x_j)$, where i is a vertex on the graph, (i, j) denotes an edge and $\|$ represents concatenation, and then in each subsequent layer l, we generate new features by EdgeConv: $\mathbf{f}_i^{(l)} = \sum_{j:(i,j)\in\mathcal{E}} \mathrm{MLP}(\mathbf{h}_i^{(l-1)} \| \mathbf{h}_i^{(l-1)} - \mathbf{h}_j^{(l-1)})$, which captures local and global geometric structure from $\mathbf{h}_i^{(l-1)} - \mathbf{h}_j^{(l-1)}$ and $\mathbf{h}_i^{(l-1)}$ respectively. To implement dense connection, starting from 1st EdgeConv layer, we use the output of all previous layers as input of each layer, and use its own output as the input of all subsequent layers. As a result, with $\mathbf{f}_i^{(l)}$ of i-th vertex through l-th EdgeConv layer, the input

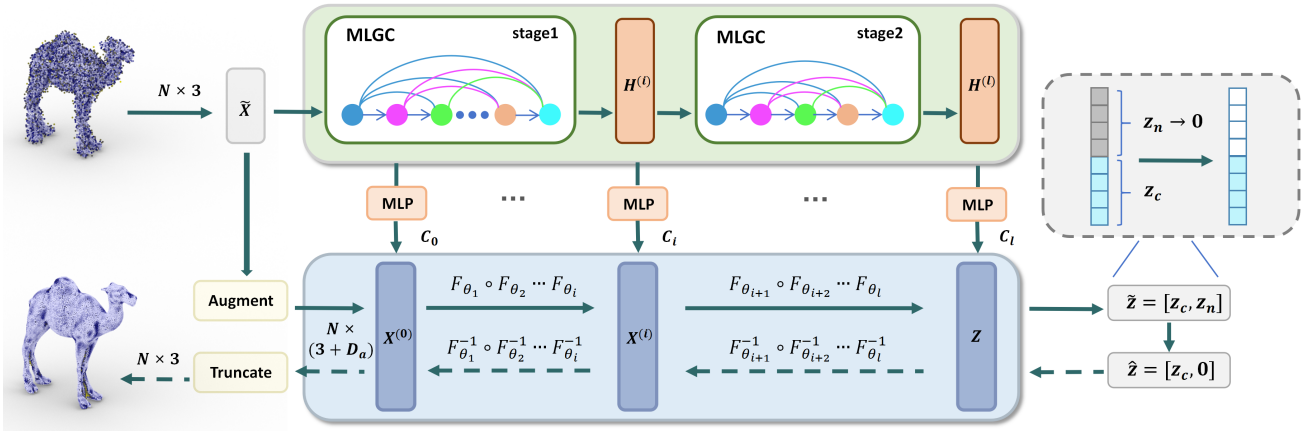

Figure 3. **Our network architecture**. The **top** region illustrates the multi-level graph convolution (MLGC) module. The **bottom** region shows the invertible neural network, which absorbs augmented points and geometry features from MLGC to obtain a disentangled latent code. The **right** region represents the process of noise factor disentanglement.

$\mathbf{h}_i^{(l)}$ for (l+1)-th layer can be described as:

$$\mathbf{h}_i^{(l)} = [\mathbf{f}_i^{(l)}, \mathbf{f}_i^{(l-1)}, ..., \mathbf{f}_i^{(0)}] \tag{4}$$

where [.] denotes concatenation. We perform dense connections in two stages separately, which constructs our hierarchical multi-level graph convolution(MLGC) network. The architecture of one stage is shown in Fig. 4.

3.4. Invertible Encoding Process

The objective of encoding process is to obtain the latent representation $\widetilde{z} = [z_c, z_n]$ of noisy point cloud, where noise and clean part are separated into z_c and z_n respectively. To achieve this, our encoding process needs to meet two requirements: 1) The encoding process is invertible, which allows to recover the original clean point cloud from the clean latent representation z_c without any additional decoder learning. 2) Effectively capturing local shape structure, which is essential to accurately predict and isolate clean representation.

We derive it from Eq. (1) that the neural network of $F(x)$ cannot pool the neighboring features to provide local area receptive field, which leads to the loss of local shape perception. Thus, we integrate features of the MLCG network to invertible neural network(INN) based on Eq. (1). To implement this, we first hierarchically construct the invertible network. According to Eq. (1), the function family of single-block forward transformation is $F := \{f : f = \left(\frac{\mathrm{Id}+G}{2}\right)^{-1} - \mathrm{Id}, Lip(G) < 1\}$, where Id denotes the identity. Further, the function family of l-th forward layer with j-block can be defined by composition $F_{\theta_l} := \{f : f = f_1 \circ \cdots \circ f_j, f_1, \cdots, f_j \in F\}$. Finally, the entire invertible transformation is the composition of several forward layers. Therefore, with the densely connected EdgeConv features $H^{(l)} \in R^{N \times C}$ from Eq. (4), we first perform dimension adaption and also multi-level features fusion through: $C^{(l)} = \mathbf{MLP}(H^{(l)})$, then we integrate $C^{(l)}$ before each F_{θ_l} transformation through addition $X_c^{(l)} = X^{(l)} + C^{(l)}$, where the enhanced transformation of l-th forward layer can be denoted as: $X^{(l+1)} = F_{\theta_l}(X_c^{(l)})$. Owning to the unconstrained and expressive architecture of INN, it can further enhance the representation capabilities of MLGC features while maintain invertibility, which makes INN and MLGC networks complement each other subtly, and effectively encode and obtain disentangled latent representation, as shown in Fig. 3.

3.5. Dimension Augmentation

According to Eq. (1), a single forward transformation F must output the results with the same dimension with the input data. Thus, if we directly feed raw 3D points to F, the feature dimension between each forward block will be consistent with 3D points, which constrain the transformation capability. Therefore, to maintain high-dimensional feature space throughout the encoding process, we augment the dimension of each points before input. Given a point x_i, we first extract its augmented features by $\mathrm{h}_i^a = f(x_i) + \sum_{x_j \in N(x_i)} g(x_j \| x_j - x_i)$, where $N(x_i)$ refers to the n nearest neighbors of x_i, f and g are parameterized by MLP. Then we concatenate them to obtain the input for 1st invertible transformation layer:

$$X^{(0)} = \{x_i^{(0)} = [x_i, \mathrm{h}_i^a]\}_{i=1}^N \tag{5}$$

where $x_i \in \mathbb{R}^3$, $\mathrm{h}_i^a \in \mathbb{R}^{Da}$ and $x_i^{(0)} \in \mathbb{R}^{(3+Da)}$. We use augmented $X^{(0)}$ as the input of the first layer of invertible encoding process, which maintains $(3 + D_a)$ dimension between each forward block, and creates high-

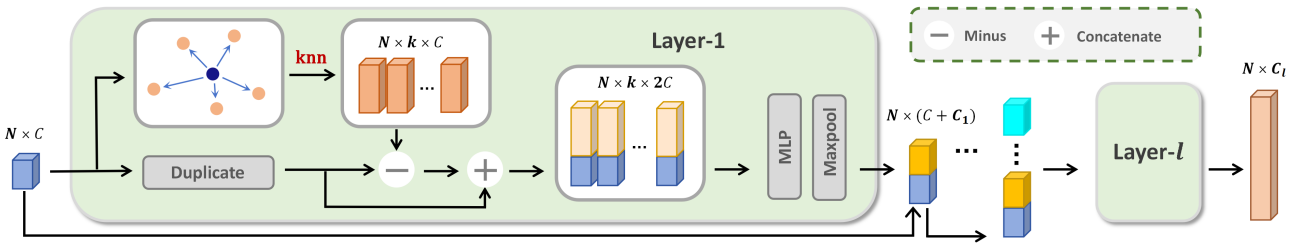

Figure 4. **One stage of multi-level graph convolution (MLGC).** Within a stage, the outputs of a graph convolution layer are concatenated with its inputs, forming the input for the subsequent layer.

dimensional spaces for further MLGC features expression and noise components uncovering.

3.6. Training Objective

Reconstruction loss. We employ the Earth Mover's Distance(EMD) to allocate clean points for supervising each denoised points, which is calculated as:

$$\mathbf{EMD}(\widehat{X}, X) = \min_{\Phi:\widehat{X}\to X} \sum_{\hat{x}\in\widehat{X}} \|\hat{x} - \Phi(\hat{x})\| \qquad (6)$$

where Φ is a bijection mapping between the denoised point cloud $\widehat{X}$ and ground truth X, which is consistent with the bijective property of our invertible neural network. We found that the displacement based methods [9, 33], typically allocate the nearest points in ground truth for each noisy points to calculate the objective displacement in loss function. However, this could lead to a considerable portion of clean points being missed during supervision, while other clean points may supervise multiple denoised points, which have a significant impact on the uniformity. Instead, EMD addresses this drawback well by maintaining the bijective supervision. We will further discuss this issue in the experiment.

Moreover, we do not employ the distribution learning loss $\log p_x(X)$ based on the prior of Gaussian distribution $p_z(Z)$, which means that we do not need to calculate the Jacobian matrix of the invertible transformations. Specifically, our prior is the latent space where noise components are disentangled and located in specific noise channels. From this perspective, the process of reconstructing points originated from clean section z_c via EMD is actually the process of mapping the prior latent space.

4. Experiments

4.1. Setup

Dataset. We train and test our framework on PUNet [50], which is same with the SOTA works of IterativePFN [9], PD-flow [35], ScoreDenoise [33]. The point clouds for training dataset are sampled from the initial 40 meshs, including resolutions of 10K, 30K and 50K, then we follow IterativePFN [9] to add Gaussian noise with standard deviation ranging from 0.05 to 0.2 of the bounding sphere radius. For the testing dataset, there are 20 meshes and the point clouds are sampled with the resolutions of 10K and 50K. Then, Gaussian noise with standard deviation including 0.1, 0.2, 0.25 were added to these point clouds for testing. Before training, we normalize each point cloud into the unit sphere and sample patches with size of 1K by employing FPS and k-NN neighbor algorithm. During the testing process, we employ the patch stitching method proposed by IterativePFN [9] which preserves points closer to the center of patches to restore point cloud from overlapping denoised patches. Furthermore, for the purpose of evaluating the generalization ability, we introduce the Paris-rue-Madame [44] dataset to visually test the denoising results of unseen real world point clouds obtained by laser scanners.

Implementation. Our framework is trained and tested on NVIDIA RTX3090 GPUs using PyTorch 1.11.0 with CUDA 11.3. We trained two versions of our framework: 1) Light-version: a single denoising module with 12 invertible transformations and EdgeConv layers, with the augmented dimension $D_a = 48$ and 679K parameters. 2) Heavy-version: 3 denoising modules which are stacked to perform the iterative denoising process internally. Each module contains 10 invertible transformations and EdgeConv layers, with the augmented dimension $D_a = 32$ and 1.4M parameters. For the training of heavy-version, we generate 2 intermediate states between noisy point cloud and ground truth for supervision during the training process. The details of implementation could be found in [9]. We train both versions of our framework for 40 epochs, with the Adam optimizer and a learning rate of 2×10^{-3}.

4.2. Quantitative Results

We evaluate our method on Chamfer distance(CD) and Point-to-mesh(P2M) distance metrics, as presented in Tab. 1. Both versions of our framework show significant advantage on CD and the state-of-the-art results on P2M.

Method	10K points						50K points					
	1% noise		2% noise		2.5% noise		1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M
Noisy	36.9	16.03	79.39	47.72	105.02	70.03	18.69	12.82	50.48	41.36	72.49	62.03
PCN [43]	36.86	15.99	79.26	47.59	104.86	69.87	11.03	6.46	19.78	13.7	32.03	24.86
GPDNet [39]	23.1	7.14	42.84	18.55	58.37	30.66	10.49	6.35	32.88	25.03	50.85	41.34
DMRDenoise [32]	47.12	21.96	50.85	25.23	52.77	26.69	12.05	7.62	14.43	9.7	16.96	11.9
PD-flow [35]	21.26	6.74	32.46	13.24	36.27	17.02	6.51	4.16	12.7	9.21	18.74	14.26
Score [33]	25.22	7.54	36.83	13.8	42.32	19.04	7.16	4.0	12.89	8.33	14.45	9.58
Pointfilter [52]	24.61	7.3	35.34	11.55	40.99	15.05	7.58	4.32	9.07	5.07	10.99	6.29
IterativePFN [9]	20.56	5.01	30.43	8.45	33.52	10.45	6.05	3.02	8.03	4.36	10.15	5.88
Ours(light)	**18.25**	**4.96**	**25.67**	**8.24**	**28.49**	**10.34**	**4.96**	3.06	**7.06**	4.47	**9.18**	5.92
Ours(heavy)	**17.81**	**4.65**	**24.41**	**7.58**	**26.99**	**9.67**	**4.70**	**2.95**	**6.46**	**4.25**	**8.63**	**5.81**

Table 1. Denoising results of different methods on PUNet [50]. CD and P2M distances are multiplied by 10^5.

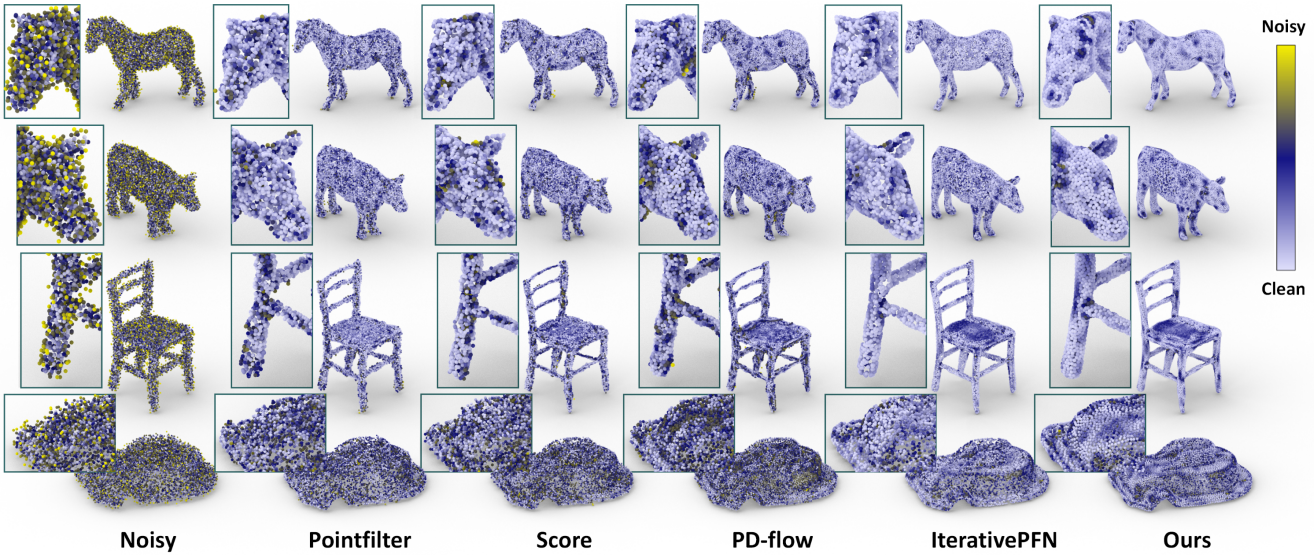

Figure 5. Visual results of point-wise P2M distance for 10K resolution shapes with 2% Gaussian noise on the bounding sphere radius.

Specifically, PCN [43] and DMRDenoise [32] are good at denoising on dense point clouds and high noise scale situations respectively. While PD-flow [35] and GPDNet [39] has good performance at low noise scale situations, which is also found in ScoreDenoise [33] with the gradient ascend denoising process. All these methods have a bias towards specific noise scales or densities. IterativePFN [9] and PointFilter [52] can reach excellent P2M especially under high noise scale and dense conditions, but our results still outperform them well, since they focus to project noisy points to the underlying clean surface by allocating the nearest points in ground truth for supervising, which misses lots of clean points and reduces uniformity. To demonstrate this, we further compare the uniformity metric of our heavy-version network with SOTA methods in Tab. 2, which shows our method have a significant advantage in terms of uniformity. Moreover, we provide quantitative comparison with traditional methods in supplementary.

4.3. Qualitative Results

Fig. 5 shows the qualitative results of SOTA works and our method on PUNet with 10K points and %2 Gaussian noise, where the color depth of each point depends on the value of P2M. We can observe from the color distribution that our method in heavy-version generates the cleanest point clouds in all methods from the noisy input shown in the first column. Furthermore, when looking at the details, we see that our denoised points are uniformly distributed on the mesh surface. While the denoised points in other methods frequently gather together, following with lots of hollow areas,

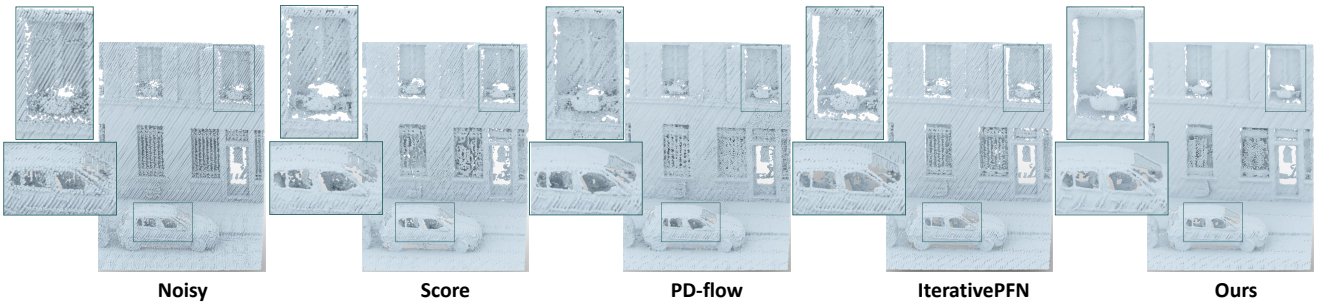

Figure 6. Visual results of our denoiser on the real-world dataset Paris-rue-Madame [44].

which influences uniformity. Fig. 6 shows the denoised results on a scenes in the RueMadame database. Our method in heavy-version not only effectively restores objects such as window sills and cars, but also eliminates the linear traces caused by laser scanning.

Noise	Methods	Uniformity for different p				
		0.4%	0.6%	0.8%	1.0%	1.2%
1%	Score [33]	1.128	1.448	1.795	2.168	2.579
	PD-flow [35]	0.346	0.474	0.597	0.745	0.925
	IterativePFN [9]	0.583	0.711	0.815	0.954	1.154
	Ours	**0.129**	**0.193**	**0.271**	**0.384**	**0.444**
2%	Score [33]	1.951	2.401	2.886	3.422	4.008
	PD-flow [35]	0.879	1.147	1.467	1.789	2.154
	IterativePFN [9]	1.042	1.161	1.331	1.531	1.800
	Ours	**0.214**	**0.417**	**0.466**	**0.545**	**0.594**
2.5%	Score [33]	2.153	2.615	3.113	3.652	4.202
	PD-flow [35]	1.099	1.392	1.715	2.061	2.474
	IterativePFN [9]	1.016	1.025	1.163	1.332	1.580
	Ours	**0.208**	**0.311**	**0.585**	**0.719**	**0.738**

Table 2. Uniformity Comparison of different methods on 10K points under various Gaussian noise scales. It is estimated in the local area of different radii p, and all results are multiplied by 10.

Ablation	**10K points**					
	1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M
w/o MLGC	25.93	10.36	38.85	16.45	50.11	24.56
w/o stage-2	18.42	5.12	26.12	8.60	29.47	11.03
1-inv-layer	23.01	7.95	34.74	13.56	43.66	20.04
6-inv-layers	18.35	**4.94**	25.94	8.27	29.36	10.76
8dim-Aug	18.59	5.13	26.32	8.63	29.78	11.09
64dim-Aug	18.25	4.96	**25.64**	**8.22**	28.61	10.49
Ours(light)	**18.25**	4.96	25.67	8.24	**28.49**	**10.34**

Table 3. Ablation study on the PUNet dataset of 10K points. CD and P2M distances are multiplied by 10^5.

4.4. Ablation Study

In order to demonstrate the effectiveness of each modules, we conduct ablation study on our light version framework shown in Tab. 3 with the following terms: (1) MLGC network. The results without the entire MLGC network becomes very poor, which demonstrates the extreme importance of its local shape information for noise disentanglement, and the adverse effect of missing stage-2 of MLGC also demonstrates the positive impact of the higher-level geometric features from stage-2. (2) Invertible neural network. We shorten our invertible transformations to 1 layer and 6 layers, respectively. We can find from the results that without the high-level latent space provided by sufficiently deep invertible transformations, it is difficult to disentangle noise components solely based on graph convolutional features. (3) Augmented dimension D_a. Compared with our light-version framework with $D_a = 48$, the feature space of $D_a = 8$ between each forward layers is not wide enough for geometric feature expression and noise disentanglement, while higher dimension like $D_a = 64$ makes minimum improvement, and its generalization performance on 2.5% noise even deteriorates. In summary, both MLGC network and the invertible neural network are indispensable. Further ablation study including noise channel z_n of latent code $\hat{z} = [z_c, z_n]$ can be found in supplementary.

5. Conclusion

In this paper, we indirectly denoise point clouds by uncovering noise in latent space. To model the encoding process, we propose a novel invertible encoding framework which integrates the features of multi-level graph convolution to the invertible neural network. This encoding process forms a bijective mapping between point cloud and latent space while capturing rich geometric structure. In latent space, noise components are disentangled and located in specific channels. We could simply mask noise channels and obtain clean points from the subsequent clean latent code via inverse transformation. Experimental results demonstrate that our method outperforms state-of-the-art methods.

References

[1] Byeongkeun Ahn, Chiyoon Kim, Youngjoon Hong, and Hyunwoo J Kim. Invertible monotone operators for normalizing flows. *Advances in Neural Information Processing Systems*, 35:16836–16848, 2022. 2, 3, 4

[2] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Point set surfaces. In *Proceedings Visualization, 2001. VIS'01.*, pages 21–29. IEEE, 2001. 1

[3] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on visualization and computer graphics*, 9(1):3–15, 2003. 1, 2

[4] HH Bauschke and PL Combettes. Convex analysis and monotone operator theory in hilbert spaces, 2011. *CMS books in mathematics). DOI*, 10:978–1. 3, 4

[5] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International conference on machine learning*, pages 573–582. PMLR, 2019. 2, 3, 4

[6] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005. 2

[7] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018. 3

[8] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[9] Dasith de Silva Edirimuni, Xuequan Lu, Zhiwen Shao, Gang Li, Antonio Robles-Kelly, and Ying He. Iterativepfn: True iterative point cloud filtering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13530–13539, 2023. 1, 2, 3, 6, 7, 8

[10] Julie Digne and Carlo De Franchis. The bilateral filter for point clouds. *Image Processing On Line*, 7:278–287, 2017. 1, 2

[11] Julie Digne, Sébastien Valette, and Raphaëlle Chaine. Sparse geometric representation through local shape probing. *IEEE transactions on visualization and computer graphics*, 24(7): 2238–2250, 2017. 2

[12] Chinthaka Dinesh, Gene Cheung, and Ivan V Bajić. Point cloud denoising via feature graph laplacian regularization. *IEEE Transactions on Image Processing*, 29:4143–4158, 2020. 1, 2

[13] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 3

[14] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 3

[15] Chaojing Duan, Siheng Chen, and Jelena Kovačević. Weighted multi-projection: 3d point cloud denoising with estimated tangent planes. *arXiv preprint arXiv:1807.00253*, 2018. 2

[16] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. Robust moving least-squares fitting with sharp features. *ACM transactions on graphics (TOG)*, 24(3):544–552, 2005. 2

[17] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018. 3

[18] Xian-Feng Han, Jesse S Jin, Ming-Jie Wang, and Wei Jiang. Guided 3d point cloud filtering. *Multimedia Tools and Applications*, 77:17397–17411, 2018. 1

[19] Xian-Feng Han, Jesse S Jin, Ming-Jie Wang, and Wei Jiang. Iterative guidance normal filter for point cloud. *Multimedia Tools and Applications*, 77:16887–16902, 2018. 1, 2

[20] Wei Hu, Xiang Gao, Gene Cheung, and Zongming Guo. Feature graph learning for 3d point cloud denoising. *IEEE Transactions on Signal Processing*, 68:2841–2856, 2020. 2

[21] Muhammad Abeer Irfan and Enrico Magli. Exploiting color for graph-based 3d point cloud denoising. *Journal of Visual Communication and Image Representation*, 75:103027, 2021. 2

[22] Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. Softflow: Probabilistic framework for normalizing flow on manifolds. *Advances in Neural Information Processing Systems*, 33:16388–16397, 2020. 3

[23] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018. 2, 3, 4

[24] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *European Conference on Computer Vision*, pages 694–710. Springer, 2020. 3

[25] Esmeide Leal, German Sanchez-Torres, and John W Branch. Sparse regularization-based approach for point cloud denoising and sharp features enhancement. *Sensors*, 20(11):3206, 2020. 2

[26] David Levin. The approximation power of moving least-squares. *Mathematics of computation*, 67(224):1517–1531, 1998. 2

[27] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics (TOG)*, 26(3): 22–es, 2007. 1

[28] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5239–5248, 2019. 4

[29] Zheng Liu, Xiaowen Xiao, Saishang Zhong, Weina Wang, Yanlei Li, Ling Zhang, and Zhong Xie. A feature-preserving framework for point cloud denoising. *Computer-Aided Design*, 127:102857, 2020. 2

[30] Cheng Lu, Jianfei Chen, Chongxuan Li, Qiuhao Wang, and Jun Zhu. Implicit normalizing flows. *arXiv preprint arXiv:2103.09527*, 2021. 3

[31] Xuequan Lu, Shihao Wu, Honghua Chen, Sai-Kit Yeung, Wenzhi Chen, and Matthias Zwicker. Gpf: Gmm-inspired feature-preserving point set filtering. *IEEE transactions on visualization and computer graphics*, 24(8):2315–2326, 2017. 2

[32] Shitong Luo and Wei Hu. Differentiable manifold reconstruction for point cloud denoising. In *Proceedings of the 28th ACM international conference on multimedia*, pages 1330–1338, 2020. 1, 2, 3, 7

[33] Shitong Luo and Wei Hu. Score-based point cloud denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4583–4592, 2021. 1, 2, 3, 6, 7, 8

[34] Aihua Mao, Zihui Du, Junhui Hou, Yaqi Duan, Yong-jin Liu, and Ying He. Pu-flow: A point cloud upsampling network with normalizing flows. *IEEE Transactions on Visualization and Computer Graphics*, 2022. 3

[35] Aihua Mao, Zihui Du, Yu-Hui Wen, Jun Xuan, and Yong-Jin Liu. Pd-flow: A point cloud denoising framework with normalizing flows. In *European Conference on Computer Vision*, pages 398–415. Springer, 2022. 1, 2, 3, 4, 6, 7, 8

[36] Enrico Mattei and Alexey Castrodad. Point cloud denoising via moving rpca. In *Computer Graphics Forum*, pages 123–137. Wiley Online Library, 2017. 2

[37] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer graphics forum*, pages 493–501. Wiley Online Library, 2009. 2

[38] Yura Perugachi-Diaz, Jakub Tomczak, and Sandjai Bhulai. Invertible densenets with concatenated lipswish. *Advances in Neural Information Processing Systems*, 34:17246–17257, 2021. 3

[39] Francesca Pistilli, Giulia Fracastoro, Diego Valsesia, and Enrico Magli. Learning graph-convolutional representations for point cloud denoising. In *European conference on computer vision*, pages 103–118. Springer, 2020. 1, 2, 3, 7

[40] Janis Postels, Mengya Liu, Riccardo Spezialetti, Luc Van Gool, and Federico Tombari. Go with the flows: Mixtures of normalizing flows for point cloud generation and reconstruction. In *2021 International Conference on 3D Vision (3DV)*, pages 1249–1258. IEEE, 2021. 3

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 3

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 1

[43] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. In *Computer graphics forum*, pages 185–203. Wiley Online Library, 2020. 1, 2, 3, 7

[44] Andrés Serna, Beatriz Marcotegui, François Goulette, and Jean-Emmanuel Deschaud. Paris-rue-madame database: a 3d mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In *4th international conference on pattern recognition, applications and methods ICPRAM 2014*, 2014. 6, 8

[45] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019. 1, 2, 3, 4

[46] Mingqiang Wei, Jin Huang, Xingyu Xie, Ligang Liu, Jun Wang, and Jing Qin. Mesh denoising guided by patch normal co-filtering via kernel low-rank recovery. *IEEE transactions on visualization and computer graphics*, 25(10):2910–2926, 2018. 2

[47] Zhongwei Xu and Alessandro Foi. Anisotropic denoising of 3d point clouds by aggregation of multiple surface-adaptive estimates. *IEEE transactions on visualization and computer graphics*, 27(6):2851–2868, 2019. 2

[48] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4541–4550, 2019. 3

[49] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 386–402, 2018. 3

[50] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2790–2799, 2018. 6, 7

[51] Jin Zeng, Gene Cheung, Michael Ng, Jiahao Pang, and Cheng Yang. 3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model. *IEEE Transactions on Image Processing*, 29:3474–3489, 2019. 2

[52] Dongbo Zhang, Xuequan Lu, Hong Qin, and Ying He. Pointfilter: Point cloud filtering via encoder-decoder modeling. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2015–2027, 2020. 2, 3, 7

[53] Feng Zhang, Chao Zhang, Huamin Yang, and Lin Zhao. Point cloud denoising with principal component analysis and a novel bilateral filter. *Traitement du signal*, 36(5), 2019. 2

[54] Qian Zheng, Andrei Sharf, Guowei Wan, Yangyan Li, Niloy J Mitra, Daniel Cohen-Or, and Baoquan Chen. Non-local scan consolidation for 3d urban scenes. *ACM Trans. Graph.*, 29(4):94–1, 2010. 2

[55] Yinglong Zheng, Guiqing Li, Xuemiao Xu, Shihao Wu, and Yongwei Nie. Rolling normal filtering for point clouds. *Computer Aided Geometric Design*, 62:16–28, 2018. 1, 2