Point cloud denoising review: from classical to deep learning-based approaches

Lang Zhou[a], Guoxing Sun[b], Yong Li[b], Weiqing Li[c], Zhiyong Su[b,*]

[a] *College of Information Engineering, Nanjing University of Finance and Economics, Nanjing, 210023, China*
[b] *School of Automation, Nanjing University of Science and Technology, Nanjing, 210094, China*
[c] *School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, 210094, China*

Abstract

Over the past decade, we have witnessed an enormous amount of research effort dedicated to the design of point cloud denoising techniques. In this article, we first provide a comprehensive survey on state-of-the-art denoising solutions, which are mainly categorized into three classes: filter-based, optimization-based, and deep learning-based techniques. Methods of each class are analyzed and discussed in detail. This is done using a benchmark on different denoising models, taking into account different aspects of denoising challenges. We also review two kinds of quality assessment methods designed for evaluating denoising quality. A comprehensive comparison is performed to cover several popular or state-of-the-art methods, together with insightful observations. Finally, we discuss open challenges and future research directions in identifying new point cloud denoising strategies.

Keywords: Point cloud, Denoising, Filtering, Noise reduction, Feature preserving

1. Introduction

Point cloud, as with meshes and RGB-D images, is one of the most popular representations for 3D objects and environments. A point cloud is a large collection of individual, unstructured 3D points in the three-dimensional coordinate system that approximate the geometry of 3D data. These points are always located on the external

*Corresponding author.
Email address: su@njust.edu.cn (Zhiyong Su)

Figure 1: Part of a chemical plant and its scanned point clouds.

surfaces of visible objects. Theoretically, a pint cloud Q with n points can be defined as $Q = \{q_i, i = 1, \ldots, n\}$, where each point $q_i \in \mathbb{R}^3$ in the 3D space is associated with different attributes, such as position (x_i, y_i, z_i) and color (r_i, g_i, b_i).

Recently, the rapid advancement of geometric sensing techniques, such as laser scanning, time-of-flight range finding, structural lighting, and stereo vision, have witnessed the wide use of 3D sensors in different areas, such as 3D reconstruction [1], autonomous driving [2], robotics [3], and augmented reality [4]. Figure 1 gives an example of the point cloud scanned from part of a chemical plant by a Trimble GX200 scanner. Point clouds acquired with these sensors, however, inevitably suffer from different levels of noise and outliers caused by measurement errors [5]. A plethora of noise sources can affect point clouds, such as the acquisition device, limitations of sensors, and the lighting or reflective nature of the surface [6]. More information about the cause of noises can be found in the survey by Chen et al. [5]. This noise not only degrades the quality of point clouds, but also hinders downstream geometry processing applications. Therefore, denoising techniques have become a critical step for improving the final quality of point clouds while preserving as many essential geometric features as possible.

1.1. Definition

Point cloud denoising, which is a fundamental and vital research area in computer graphics and computer vision, aims to recover the ground-truth point cloud by removing unwanted noises from a given noisy input. A noisy point cloud, denoted by

$\mathcal{P} = \{p_i, i = 1, \ldots, n\}$, can roughly be approximated as:

$$p_i = q_i + e_i, \qquad p_i, q_i, e_i \in \mathbb{R}^3, \tag{1}$$

where q_i denotes a point from the ground-truth point cloud Q, and e_i denotes measurement noises at the location q_i. The denoising process can be formulated as

$$q_i' = \mathcal{F}(p_i) = \mathcal{F}(q_i + e_i), \tag{2}$$

where q_i' denotes an approximation of q_i, and the function $\mathcal{F}(\cdot)$ stands for a general denoising method. The majority of existing point cloud denoising methods concentrate on removing different kinds of synthetic noises [7, 8, 9, 10, 11, 12, 13], especially the additive white Gaussian noise with zero mean and standard deviation σ. However, synthetic noises with a simple distribution, such as Gaussian, are difficult to mimic the complex real-world noise [14]. Therefore, increasing attention has been paid to remove real-world noises recently [15, 16, 17, 18], which are more complex than synthetic noises.

1.2. Contribution

This survey aims to build a starting point for researchers new to the topic, act as a reference guide for the community around point cloud denoising, and to introduce exciting open research questions. In the literature, few surveys of point cloud denoising are available, such as [5, 6]. However, they mainly discussed classical methods that were proposed at least five years ago. In recent years, numerous new methods and trends have emerged, especially in the field of deep learning for point cloud denoising [7, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19]. Our survey strives to provide a comprehensive and up-to-date overview of the point cloud denoising methods proposed in the past five years, in particular the deep learning-based methods. A classification of the main point cloud denoising methods discussed in this survey is illustrated in Figure 2. However, It is worth noting that it is impossible to discuss all the published papers in this field. We refer our readers to previous surveys [5, 6] for details about the point cloud denoising methods proposed five years ago. Besides, the outlier detection problem is also not covered in this survey. Readers can refer to [20, 21] for a comprehensive survey of the outlier detection.

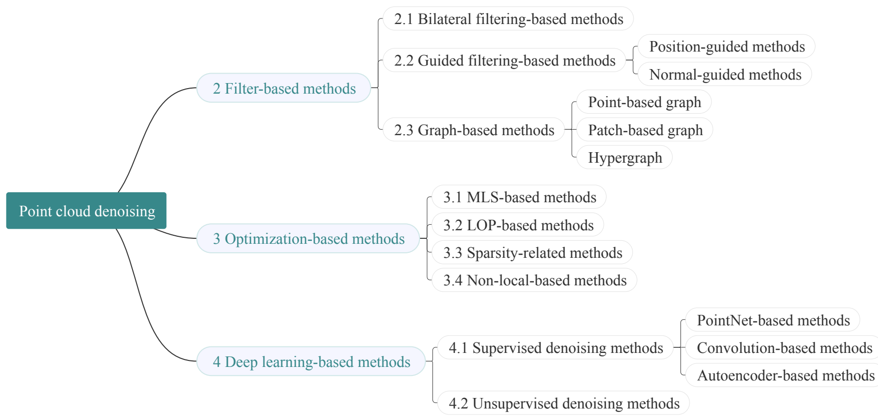


Figure 2: A taxonomy of point cloud denoising methods.

Compared with existing surveys, the major contributions of this work can be summarized as follows:

1) As opposed to existing reviews [5, 6], this paper focuses on reviewing point cloud denoising algorithms proposed during the last five years.
2) To the best of our knowledge, this is the first survey to comprehensively cover deep learning-based point cloud denoising methods as well as quality assessment approaches for denoised point clouds.
3) Comprehensive comparisons of several popular or state-of-the-art methods on selected noisy point clouds are provided, with summaries and insightful discussions being presented.

1.3. Organization

The structure of this paper is as follows. Section 2 reviews filter-based methods. Section 3 presents a review of optimization-based approaches. Deep-learning-based methods are discussed in Section 4. Section 5 provides a survey of existing quality assessment techniques for denoised point cloud. Section 6 compares some popular and state-of-the-art point cloud denoising algorithms. Section 7 discusses some open challenges and future directions. Finally, Section 8 concludes the paper.

2. Filter-based methods

Filter-based denoising methods, which are mainly inherited from ideas of image processing, usually assume that the noise is high frequency, and design filters that operate on point positions or point normals. These methods can be roughly divided into bilateral filtering-based, guided filtering-based, and graph-based methods.

2.1. Bilateral filtering-based methods

Bilateral filtering, originally designed for the image denoising by Tomasi et al. [22], is a nonlinear technique to smooth an image while preserving strong edges. The key idea is that it considers values of neighbors that are close in both position and pixel value.The bilateral filtering for an image $I(i)$ at the pixel $i = (x, y)$ can be defined as follows:

$$I'(i) = \frac{\sum_{j\in\mathcal{N}(i)} w_c(\|j-i\|)w_s(|I(i)-I(j)|)I(j)}{\sum_{j\in\mathcal{N}(i)} w_c(\|j-i\|)w_s(|I(i)-I(j)|)}, \tag{3}$$

where $\mathcal{N}(i)$ denotes the neighborhood of i, $w_c(x) = e^{-x^2/2\sigma_c^2}$ is a spatial smoothing function with standard deviation σ_c, and $w_s(x) = e^{-x^2/2\sigma_s^2}$ is an intensity smoothing function with standard deviation σ_s. This concept has been extended and widely used for denoising point clouds [5, 6].

Bilateral filtering-based point cloud denoising methods apply the bilateral filter directly to point clouds based on point position, point normal, point color, etc [23, 24]. Digne et al. [23] extended the bilateral filtering on meshes introduced by Fleishman et al. [25]to point clouds by taking into account both spatial and normal distances. The denoised position $p_i^{'}$ of p_i is updated by

$$p_i^{'} = p_i + \delta_i \cdot \mathbf{n}_i, \tag{4}$$

where $\mathbf{n}_i$ is the normal of p_i, and δ_i is a weight coefficient defined as:

$$\delta_i = \frac{\sum\limits_{p_{ij}\in\mathcal{N}_r(p_i)} w_d\left(\|\mathbf{v}_{ij}\|\right) w_n\left(|\langle\mathbf{n}_i, \mathbf{v}_{ij}\rangle|\right)\langle\mathbf{n}_i, \mathbf{v}_{ij}\rangle}{\sum\limits_{p_{ij}\in\mathcal{N}_r(p_i)} w_d\left(\|\mathbf{v}_{ij}\|\right) w_n\left(|\langle\mathbf{n}_i, \mathbf{v}_{ij}\rangle|\right)}, \tag{5}$$

where $\mathbf{v}_{ij} = p_{ij} - p_i$, $\mathcal{N}_r(p_i) = \{p_{ij} \in \mathcal{P} | \| p_{ij} - p_i \| < r\}$ is the neighbors of p_i defined in a r-ball (neighbors within radius) centered at p_i, $w_d(x) = e^{-x^2/2\sigma_d^2}$ and $w_n(x) = e^{-x^2/2\sigma_n^2}$

Table 1: Summary of bilateral filtering-based point cloud denoising methods

Method	Weighting function	Comments
Digne et al. (2017) [23]	Point position, point normal	Parallel implementation; trade-off between speed and quality
Zhang et al. (2019) [24]	Point position, point normal, point color	Preserves sharp feature to some extent; removes small-scale noise

are two 1D Gaussian functions with chosen variance σ_d and σ_n respectively, and $\langle , \rangle$ is the inner product of two vectors. In order to improve the computation efficiency, they implemented the proposed scheme through the OpenMP [26] API which supports multi-platform shared-memory parallel programming. Thus, it can balance the processing speed and denoising quality.

Besides point position and point normal, Zhang et al. [24] also considered point color and designed a bilateral filter to denoise point clouds collected by Kinect for Windows v2. Let $L(p_i)$ be the grayscale intensity value, the key of their bilateral filter is the improved weight coefficient δ_i in Eq.(5):

$$\delta_i = \frac{\sum\limits_{p_{ij} \in \mathcal{N}_r(p_i)} w_d\left(\| \mathbf{v}_{ij} \|, |d_{ij}|\right) w_n\left(|\langle \mathbf{n}_i, \mathbf{v}_{ij} \rangle|\right) \langle \mathbf{n}_i, \mathbf{v}_{ij} \rangle}{\sum\limits_{p_{ij} \in \mathcal{N}_r(p_i)} w_d\left(\| \mathbf{v}_{ij} \|, |d_{ij}|\right) w_n\left(|\langle \mathbf{n}_i, \mathbf{v}_{ij} \rangle|\right)}, \tag{6}$$

where $d_{ij} = |L(p_{ij}) - L(p_i)|$ is the gray-scale difference between point p_i and its neighboring point p_{ij}, $w_d(x, y) = e^{-(x^2+y^2)/2\sigma_d^2}$ is a standard 2D Gaussian function.

Table 1 summarizes the weighting function used in the bilateral filter of recent bilateral filtering-based approaches introduced in this section. The pros and cons of different algorithms are also listed. Overall, the process of bilateral filtering-based methods is simple and non-iterative. They consider spatial and normal closeness, as well as grey level similarity in the denoising process. Though these methods are able to preserve edges, they are not good at precise noise correction near sharp edges or corners.

2.2. *Guided filtering-based methods*

The guided filter, which was first proposed by He et al. [27], is an explicit image filter, and can perform as an edge-preserving smoothing operator. The key assumption is that the output image is a local linear model between the guidance image I^g and the filter output I^o. Given a guidance image I^g, the filtering output I^o of an input image I^i is defined as a linear transform of I^g in a window ω_i centered at the pixel i:

$$I^o(j) = a_i I^g(j) + b_i, \quad \forall j \in \omega_i \tag{7}$$

where a_i and b_i are linear coefficients assumed to be constant in ω_i.

Guided filtering-based point cloud denoising methods seek to transfer important structural details contained in the guidance point cloud to the target point cloud. The most popular guidance information is the target point cloud itself [28] or denoised outputs from previous filtering iterations [29]. According to the type of guidance information, these methods can further be divided into two categories: position-guided [28] and normal-guided methods [29, 30, 31].

2.2.1. Position-guided methods

This kind of method employs the point position as a simple and direct guidance information. In [28], Han et al. extended the guided image filtering technique [27] to point clouds, and proposed a point position guided filtering approach. They employed the input point cloud itself as the guidance point cloud, called self-guided point cloud filtering. For each neighboring point $p_{ij} \in \mathcal{N}(p_i)$ of p_i, the filtered position $p_{ij}^{'}$ is defined as:

$$p_{ij}^{'} = a_i p_{ij} + b_i, \tag{8}$$

where a_i and b_i are coefficients of the linear model respectively, which are computed by minimizing the following function:

$$J(a_i, b_i) = \sum_{p_{ij} \in \mathcal{N}(p_i)} \left(\left(a_i p_{ij} + b_i - p_{ij} \right)^2 + \epsilon a_i^2 \right), \tag{9}$$

where ϵ is a controlling parameter. The above Eq. (9) can be solved by:

$$a_i = \frac{\frac{1}{|\mathcal{N}(p_i)|} \sum p_{ij} \cdot p_{ij} - \overline{p_i} \cdot \overline{p_i}}{\left(\frac{1}{|\mathcal{N}(p_i)|} \sum p_{ij} \cdot p_{ij} - \overline{p_i} \cdot \overline{p_i} \right) + \epsilon}, \tag{10}$$

$$b_i = \overline{p_i} - a_i \overline{p_i}, \tag{11}$$

where $\overline{p_i}$ is the centroid of $\mathcal{N}(p_i)$. However, this method cannot recover sharp features such as the corner, since only the position information of a point is considered.

2.2.2. Normal-guided methods

Besides the point position, the majority of recent guided filtering-based methods also employ the point normal as guidance signals. This kind of method usually estimates a single normal or multi normals for each point.These normals are then iteratively filtered by using the normal field updated in the previous iteration as guidance. Point positions are finally updated to match the estimated normals.

Single normal-based methods. Han et al. proposed an iterative guidance normal filter for point cloud denoising [31], which was inspired by the iterative idea of rolling guidance filter [35] for image denoising:

$$\mathbf{n}_i^{k+1} = \frac{1}{K_i} \sum_{p_{ij} \in \mathcal{N}(p_i)} w_d \left(\left\| p_{ij} - p_i \right\| \right) w_n \left(\left\| \mathbf{n}_i^k - \mathbf{n}_j^k \right\| \right) \cdot \mathbf{n}_j^k, \tag{12}$$

where

$$K_i^k = \sum_{p_{ij} \in \mathcal{N}(p_i)} w_d \left(\left\| p_{ij} - p_i \right\| \right) w_n \left(\left\| \mathbf{n}_i^k - \mathbf{n}_j^k \right\| \right). \tag{13}$$

The initial normal $\mathbf{n}_i^0$ of each point p_i is estimated by using the Principal Components Analysis (PCA) [36]. In the kth iteration, $\mathbf{n}_i^{k+1}$ is computed in a bilateral filtering form with respect to the normal field $\{\mathbf{n}_i^k\}$ in the previous iteration. The Eq.(12) uses the normal field as guidance to filter the newly adjusted normal field in the previous iteration. After obtaining filtered normals, each point is adjusted to match its estimated normal by extending the iterative point updating scheme proposed by Sun et al. [37]. However, the isotropic point updating process treats each point indiscriminately which may smooth sharp regions, such as corners or edges. To address the above problem, Yadav et al. [29] introduced an anisotropic point cloud denoising algorithm. In the point normal filtering stage, they defined a point-based Normal Voting Tensor (NVT) based on the variation of point normals. Noise and sharp features are decoupled using the spectral analysis of the point-based NVT and noise components are suppressed using Binary Eigenvalues Optimization (BEO). In the point updating stage, they classified all points into corners, edges, and planar points using the spectral analysis of a weighted anisotropic covariance matrix. Restricted quadratic error metrics, which are different for different kinds of feature points, are introduced to update point positions by utilizing distance-based constraints. These stages are iteratively applied to the input point cloud to get the final denoised output. Since single normals at feature points are ambiguous and undefinable, single normal-based approaches may lead to cross artifacts at sharp features.

Multi-normal-guided methods. To preserve sharp features better, the multi-normal strategy is widely adopted by assigning feature points with multiple normals according to their feature type [30, 32, 33]. In [30], Zheng et al. extended the guidance normal filter for mesh normal smoothing [38] to point clouds via a multi-normal strategy. The multi-normals of each feature point are estimated by partitioning its k-NN into piecewise smooth patches with each smooth patch corresponding to one normal. They evaluated a guided normal for each point normal by computing the average normal of its most consistent patch. Then, they applied the guided filter to the normal field to get a piecewise smooth one. Based on the similar idea of [30], Liu et al. [32] also adopted the multi-normal strategy and presented a feature-preserving framework to recover a noise-free point cloud. They developed an anisotropic second-order regularization method to restore the point normal field from the noisy input. A bi-tensor voting scheme combining the normal and point tensor voting is then introduced to detect feature points. Multiple normals at each feature point are estimated by using a simple yet effective Random Sample Consensus (RANSAC)-based algorithm [39].

While most point cloud denoising methods try to remove unwanted noise on the premise of preserving geometric features, on the contrary, Zheng et al. [33] adopted a multi-normal strategy to remove different scales of geometric features from noisy point clouds. They extended the mesh rolling guidance normal filter [35, 38] to process the normal field on the point cloud. To overcome the normal discontinuity along sharp features, they adopted the multi-normal strategy [30] during point position updating. Their approach is robust in removing small-scale geometric features. Therefore, Sun et al. [34] exploited the rolling guidance normal filter [33] to suppress multi-scale textures while preserving prominent structures for point clouds with rich textures. However, this method may filter several detailed features with important semantic information.

A summary of guided filtering-based approaches is presented in Table 2, with the emphasis on guidance information, features, advantages and disadvantages. Overall, the guidance information can either be static or dynamic. Static guidance (e.g., the

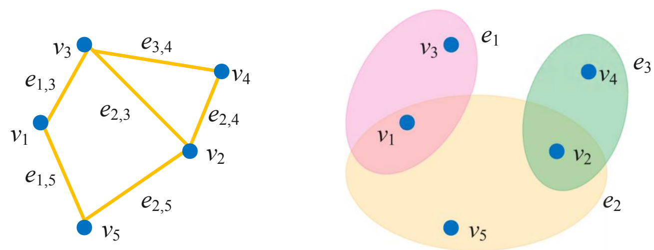
Table 2: Summary of guided filtering-based point cloud denoising methods

Method	Guidance	Features	Comments
Han et al. (2018) [28]	Point position	Self-guided denoising	Does not recover sharp features
Zheng et al. (2017) [30]	Point position; point normal	Multi-normal guided normal filter	Needs fine parameters tuning; does not deal with different scales of features
Han et al. (2018) [31]	Point position; point normal	An iterative guidance normal filter	Time-consuming; cannot preserve sharp features
Yadav et al. (2018) [29]	Point position; point normal	Vertex-based normal voting tensor; binary eigenvalues optimization based normal filter	Preserves sharp edges and corners
Liu et al. (2020) [32]	Point position; point normal	Anisotropic second order normal filter; bitensor voting based feature points detection	Preserves different levels of geometric features
Zheng et al. (2018) [33]	Point position; point normal	Rolling guidance normal filte	Prevents large-scale sharp structures from severe distortion
Sun et al. (2019) [34]	Point position; point normal	Denoising point clouds with rich textures; rolling guidance normal filter	Limited efficiency; filters several detailed features with important semantic information

point cloud itself [28]) provides direct and intuitive control over the denoising process. However, the static guidance should be specified beforehand, and remains static during the denoising processing. Dynamic guidance (e.g., point normals [29, 30, 31, 32, 33]) is automatically updated according to the previous iterate, but can be less robust when there are outliers or noises in the input point cloud.

2.3. Graph-based methods

Graph-based point cloud denoising methods first interpret the input point cloud as a graph signal, and then perform denoising via chosen graph filters. There are several clear connections between graph features and point cloud characteristics. For example, the flatness of surfaces in point clouds can be described by the smoothness over a graph. Due to the graph's ability to capture underlying geometric structures of point clouds,



(a) (b)

Figure 3: An illustration of graph and hypergraph. (a) A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$ and $\mathcal{E} = \{e_{1,5}, e_{1,3}, e_{2,3}, e_{2,4}, e_{2,5}, e_{3,4}\}$. (b) A hypergraph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$ with $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$ and set of hyperedges $\mathcal{E} = \{e_1, e_2, e_3\}$

recently, the graph signal processing technique has shown great success in point cloud processing [40].

A weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ is often defined by two sets: a node set $\mathcal{V}$ of cardinality $|\mathcal{V}| = n$ and an edge set $\mathcal{E}$, as illustrated in Figure 3(a). Nodes in a graph represent entities (e.g., people in a social network or points in a point cloud), whereas edges represent relationships between those entities. $\mathbf{W}$ is a weighted adjacency matrix. To reflect the degree of pairwise similarity between node v_i and node v_j, a weight $w_{i,j} \in \mathbf{W}$ is often assigned to each edge $e_{i,j} \in \mathcal{E}$. According to the graph construction approach, current graph-based methods can be divided into point-based, patch-based and hypergraph-based methods.

2.3.1. *Point-based graph*

These methods define each point as a node, and each node is connected through edges to its k nearest neighbors (k-NN). Duan et al. constructed a k-NN graph based on the Euclidean distance between points, whose nodes are points and edges are proximities of points, to capture local and global geometric structures of the input point cloud [41]. Instead of directly smoothing positions or normals of points, they proposed a weighted multi-projection (WMP) denoising algorithm. A tangent plane was estimated at each point to locally approximate the underlying manifold based on the graph structure. Then, they projected each point to its neighbors' tangent planes, and averaged the multiple projections to obtain the denoised point. Besides employing the geometry information in the Euclidean space to build the graph, Irfan et al. [42] took advantage of the correlation between geometry and color attribute of a point cloud, and generated a suitable k-NN graph based on both color similarity and geometry proximity. They applied a graph-based convex optimization to obtain the denoised point cloud.

2.3.2. Patch-based graph

These methods build the graph based on surface patches of point clouds where each patch is defined as a node, and employ the Graph Laplacian Regularizer (GLR) to denoise point clouds [43, 44, 45]. Let $\mathbf{D}$ be the diagonal degree matrix of $\mathcal{G}$, where $d_{i,i} = \sum_{j=1}^{n} w_{i,j}$. Given $\mathbf{W}$ and $\mathbf{D}$, the combinatorial graph Laplacian matrix $\mathbf{L}$ [46] can be written as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. The GLR [46] is defined as:

$$\mathbf{v}^{\top}\mathbf{L}\mathbf{v} = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\left(v_i - v_j\right)^2, \tag{14}$$

where $\mathbf{v} \in \mathbb{R}^N$ is a graph signal defined on the graph $\mathcal{G}$, v_i and v_j are a pair of connected nodes. Consider an input noisy point cloud $\mathcal{P}$ as a graph signal defined on the graph $\mathcal{G}$, where p_i is a scalar value assigned to node v_i. It has been shown in [43, 44, 45] that minimizing the GLR $\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\left(p_i - p_j\right)^2$ iteratively can promote piecewise smoothness in the reconstructed graph signal $\mathcal{P}$ [43, 44, 45].

Hu et al. [43] divided the input point cloud into a set of overlapping patches, which are aligned via translation so each patch has its center at the origin. A k-NN graph was then built over each pair of adjacent patches by using the k-NN algorithm to search the nearest patches of each patch as the neighbors based on the Euclidean distance between patch centers. After that, they formulated the problem of point cloud denoising as minimization of GLR using the Mahalanobis distance matrix as a variable. Interpreting the above Mahalanobis distance matrix as a graph Laplacian, they developed a feature graph learning scheme to determine edge weights, where they employed positions and surface normals as relevant features for each point. An alternating algorithm was introduced to efficiently solve the formulated problem by alternately optimizing the denoised point cloud and the Mahalanobis distance matrix. Similar to [43], Zeng et al.

[44] generated surface patches using the same method as in [43]. They constructed a patch graph based on extracted surface patches by developing a discrete patch distance measure to quantify the similarity between two same-sized surface patches. Assuming that surface patches in the point cloud lie on a manifold of low dimension, they extended a previously proposed low-dimensional manifold model (LDMM) [47] for image patches to surface patches in the point cloud. To efficiently adopt the LDMM to discrete point cloud patches, they approximated the computation of the patch-manifold dimension defined in continuous domain with a discrete patch-based graph Laplacian regularizer. Finally, they exploited the surface self-similarity characteristic and simultaneously denoised similar patches by minimizing the manifold dimension. However, it focuses on removing a high level of noise but it dose not preserve the sharp structures. Different from [43] and [44], Dinesh et al. [45] divided the point cloud into two disjoint node sets (red and blue), and constructed a bipartite graph approximation of the original k-NN graph $\mathcal{G}$ by simplifying the method in [48]. They developed a new GLR term called signal-dependent feature graph Laplacian regularizer (SDFGLR) to optimize the red and blue nodes' coordinates alternately.

2.3.3. Hypergraph

Except for the traditional graph [41, 43, 44, 45], the hypergraph is also introduced to denoise noisy point clouds. The problem of denoising a signal on a hypergraph is formulated as a convex minimization problem with the constraints that denoised signals should be smooth over the hypergraph [40].

A hypergraph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$ consists of a set of nodes $\mathcal{V}$ and a set of hyperedges $\mathcal{E}$. In a traditional graph, each edge can only connect two nodes, constraining graph-based models to describe only pairwise relationships. However, the hypergraph is a high-dimensional graph model, in which each hyperedge can connect more than two nodes, as illustrated in Figure 3(b). Hence, it can characterize the multilateral relationship among several related nodes. The hypergraph signal processing is a tensor-based framework [49]. A tensor is a high-dimensional generalization of matrix, which can be interpreted as multi-dimensional arrays. The tensor outer product between an mth-order tensor $\mathbf{U} \in \mathbb{R}^{I_1 \times \ldots \times I_m}$ with entries $u_{i_1 \cdots i_m}$ and an nth-order tensor $\mathbf{V} \in \mathbb{R}^{J_1 \times \ldots \times J_n}$

with entries $v_{j_1\cdots j_n}$ can be defined as $\mathbf{O} = \mathbf{U} \circ \mathbf{V}$, where $\mathbf{O} \in \mathbb{R}^{I_1\times\ldots\times I_m\times J_1\times\ldots\times J_n}$, and $o_{i_1\cdots i_m j_1\cdots j_n} = u_{i_1\cdots i_m} \cdot v_{j_1\cdots j_n}$.

In [40], Zhang et al. explored the hypergraph model and developed hypergraph signal processing tools for effectively denoising point clouds. The noisy point cloud with n nodes is denoted by a location matrix $\mathbf{s} = [s_1 \;\; s_2 \;\; \ldots \;\; s_n]^{\mathrm{T}}$. Given a hypergraph with n nodes and longest hyperedge connecting m nodes, it can be represented by an m-th order n-dimension representing tensor $\mathbf{A} = (a_{i_1 i_2\cdots i_m}) \in \mathbb{R}^{n^m}$, whose entry in position $(i_1, i_2, \cdots, i_m)$ is labeled as $a_{i_1 i_2\cdots i_m}$. They referred the adjacency tensor as the representing tensor $\mathbf{A}$, in which each entry $(a_{i_1 i_2\cdots i_m})$ indicates whether nodes $v_1, v_2, \cdots, v_m$ are connected in the hyperedges. The representing tensor $\mathbf{A}$ can be decomposed via

$$\mathbf{A} \approx \sum_{r=1}^{n} \lambda_r \cdot \underbrace{\mathbf{f}_r \circ \ldots \circ \mathbf{f}_r}_{m \text{ times}}, \tag{15}$$

where $\mathbf{f}_r's$ are orthonormal basis vectors called spectrum components, and λ_r are frequency coefficients related to the hypergraph frequency. For clean point clouds, they estimated their spectrum components $\mathbf{f}_r's$ based on the hypergraph stationary process and optimally determined their frequency coefficients λ_r based on smoothness to recover the original hypergraph structure. Then, given the original signal $\mathbf{s} = [s_1 \;\; \ldots \;\; s_n]^{\mathrm{T}}$, the hypergraph signal is defined as the $(m-1)$ times tensor outer product of $\mathbf{s}$, i.e.,

$$\mathbf{s}^{[m-1]} = \underbrace{\mathbf{s} \circ \ldots \circ \mathbf{s}}_{m-1 \text{ times}}. \tag{16}$$

Given the hypergraph signal $\mathbf{s}^{[m-1]}$ and the representing tensor $\mathbf{A}$, they jointly estimated hypergraph spectrum pairs $(\mathbf{f}_r, \lambda_r)$ and denoised noisy point clouds.

Table 3 summarizes the characteristics of graph-based approaches discussed in this section, with the emphasis on the graph construction method, filter, features, strengths and weaknesses. All in all, graph-based techniques have proved to achieve very strong performance when the noise level is low. However, at high noise levels, the graph construction can become unstable, negatively affecting the denoising performance [10, 11].

Table 3: Summary of graph-based point cloud denoising methods

Method	Graph construction	Filter	Feature	Comments
Duan et al. (2018) [41]	k-NN	Weighted multi-projection	Projects each point to its neighbors' tangent planes	Over-smoothing
Hu et al. (2020) [43]	k-NN on patches	Graph laplacian regularizer using the Mahalanobis distance matrix	Feature graph learning for denoising	Requires small amounts of data for a stable estimation
Zeng et al. (2020) [44]	k-NN on a manifold	Signal-dependent feature graph Laplacian regularizer	Uses the patch manifold prior	Preserves structural detail
Dinesh et al. (2020) [45]	Bipartite graph	Signal-dependent feature graph laplacian regularizer	Applied on 3D coordinates and surface normals	Removes Gaussian and Laplacian noise only
Irfan et al. (2021) [42]	k-NN graph based on geometry and color	The regularity of the color, and its correlation with the proximity of points	Takes advantage of the correlation between the geometry and color attribute	Both for geometry and color denoising
Zhang et al. (2021) [40]	Spectrum-based hypergraph	Tensor-based methods	Hypergraph signal processing	Both for sampling and denoising

3. Optimization-based methods

Optimization-based denoising methods formulate the denoising process as an optimization problem. It seeks for a denoised point cloud that can best fit the input point cloud, and a set of constraints defined by the priors of the ground-truth geometry and noise distribution. Optimization-based methods usually involve a number of parameters and require careful trial-and-error parameter tuning to achieve decent results, especially for complex models. These methods can be generally classified into four groups: Moving Least Squares (MLS)-based, Locally Optimal Projection (LOP)-based, sparsity-based, and non-local-based methods.

3.1. MLS-based methods

MLS-based methods reconstruct a smooth surface from the input point cloud, and iteratively project input points onto the approximated underlying surface. As a pioneering work, Alexa et al. [50] first extended the MLS method proposed by [51] to define a smooth manifold surface from a set of points close to the original surface. Afterwards, several modified MLS methods for feature preservation have been proposed, such as robust MLS (RMLS) [52], and robust implicit MLS (RIMLS) [53]. However, due to the isotropic weights in the MLS, these methods tend to over-smooth sharp features in the point cloud. Recently, Xu et al. [54] developed an anisotropic denoising algorithm based on a dense aggregation of MLS estimates defined on asymmetric directional neighborhoods. For each point, its local coordinate system (LCS) is first constructed by using local PCA. Then, they employed the Local Polynomial Approximation (LPA)-Intersection of Confidence Intervals (ICI) technique [55] to automatically determine four adaptive directional neighborhoods. Thereby, four local MLS estimates are computed for each point. A novel strategy is introduced to aggregate the overlapping adaptive local estimates of each point to gain a stable and accurate estimation of the point. Thanks to asymmetric directional neighborhoods, their method can adapt to edges and discontinuities using much larger supports than classical MLS based on symmetric weights.

All in all, MLS-based methods provide a degree of robustness in presence of outliers. However, these methods are designed to reconstruct surfaces with piecewise smooth priors. Therefore, sharp features can be easily removed together with noise.

3.2. LOP-based methods

Unlike MLS-based methods, instead of computing explicit parameters for the surface, LOP-based methods aim to produce a set of points to represent the underlying surface while enforcing a uniform distribution over the point cloud. The original parameterization-free LOP-based denoising technique, which was first proposed by Lipman et al. [56], consists of two optimization terms: a data term and a repulsion term. The data term projects a set of points onto the latent geometry of the input point cloud. While the repulsion term strives to keep the distribution of projected points fair. However, the original LOP may fail to converge and cannot distribute points uniformly under significant non-uniformity of input point clouds. Therefore, many modifications to [56] have been proposed, such as weighted LOP (WLOP) [57], edge-aware resampling (EAR) [58], feature-preserving LOP operator (FLOP) [59], and continuous weighted LOP (CLOP) [60]. However, like most MLS-based methods, LOP-based methods may also suffer from over-smoothing when the noise level is high because of their inherent isotropic nature [13]. To address this issue, Lu et al. [61] proposed a Gaussian Mixture Model (GMM) inspired anisotropic LOP approach for point cloud denoising (GPF). The GPF mehtod also contains a data term and a repulsion term. A minor difference is that its data term is inspired by the GMM and incorporates the normal information. They assumed that the distribution of noisy input points follows a GMM, which is defined by a set of centroids and covariances. After smoothing normals of the input point cloud with the bilateral filter [22], they projected points onto the underlying surface by formulating the projection problem using a GMM. Features in the projected point cloud can be automatically preserved by considering the filtered normal information during projection. They introduced energy terms to preserve geometric features and obtained a uniform point distribution on the surface. However, the GPF method may expand the volume of the input point cloud since it pushes points into the edge region.

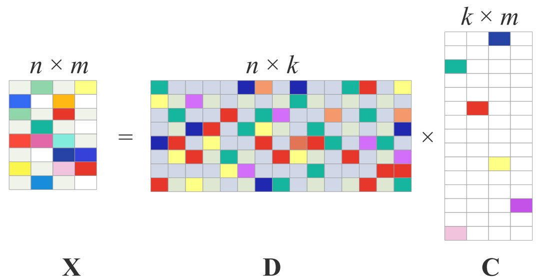


Figure 4: An illustration of the sparse reconstruction.

3.3. Sparsity-based methods

Sparsity-based methods, which are based on the sparse representation theory [65], hold the assumption that many common surfaces are piecewise smooth. In other words, the surface is smooth almost everywhere except at some small number of sparse features that form sharp features [66]. These methods seek to represent the point cloud as a linear combination of a few elementary signals from a possibly redundant dictionary [62]. To learn a good dictionary over which the signals will be sparsely decomposed, as illustrated in Figure 4, a typical sparse reconstruction problem can be defined as the minimization of [67]:

$$\min_{\mathbf{C}} \frac{1}{2} \parallel \mathbf{X} - \mathbf{DC} \parallel_F^2 + \lambda \parallel \mathbf{C} \parallel_p, \tag{17}$$

where $\mathbf{X} \in \mathcal{R}^{n \times m}$ is the data matrix constructed from the input noisy point cloud $\mathcal{P}$, whose columns are signals, $\mathbf{D} \in \mathcal{R}^{n \times k}$ is a dictionary which is possibly over-complete, $\mathbf{C} \in \mathcal{R}^{k \times m}$ is the matrix of sparse coefficients, λ controls the trade-off between sparsity and reconstruction error, F denotes the Frobenius norm, and $0 \leq p \leq 1$. If $p = 0$, Eq. (17) is a non-convex norm such that it is quite hard to obtain the optimal result . Otherwise, when $p = 1$, Eq. (17) is convex, and a lot of approaches have been invented to solve it [67].

Sparsity-based methods generally cover two main steps. In the first step, a sparse reconstruction of surface normals is obtained by solving a global minimization problem with sparsity regularization. In the second step, each point is updated by solving global minimization based on reconstructed normals and local planarity hypothesis.

In earlier works, the ℓ_1 [68] and ℓ_0 [66] based regularization methods used the sparsity of first order information to remove noise. These methods preserve sharp features well but may suffer from undesired staircase effects in smoothly curved regions because of their high sparsity requirement [32]. The Moving Robust Principal Components Analysis (MRPCA) proposed by Mattei et al. [62] used weighted ℓ_1 minimization of the point deviations from the local reference plane to preserve sharp features. They modeled the point cloud as a collection of overlapping two-dimensional subspaces, where each point will be a member of multiple overlapping neighborhoods. They introduced a method in which all the independent estimates for a single point are pooled to yield a collaborative estimate for that point. Sharp features are preserved via a weighted ℓ_1 minimization, where the weight measures the similarity between normal vectors in a local neighborhood. However, their method, which depends on sparse and low-rank modeling, tends to over-sharpen smoothly curved features. In the same spirit of MRPCA, Leal et al. [63] proposed a new model to reconstruct and smooth point clouds, combining ℓ_1-median filtering with sparse ℓ_1 regularization. Different from MRPCA, they used sparsity in both data fitting and the prior term. Their approach comprises two iterative steps, namely normal denoising and point position updating. In the normal denoising step, they first found a regression plane equidistant to all heights in a local neighborhood. Then, they calculated the normal at the plane by integrating ℓ_1-median height filter and ℓ_1 regularization of total variation. In the point position updating step, based on the estimated normals, they updated each point by using the orthogonal distance of the noisy point to the local regression plane, shifting the point along the normal direction projecting it onto the plane.

In addition to the above ℓ_1 [68] and ℓ_0 [66] based regularization methods, Digne et al. [64] introduced the dictionary learning algorithm and developed a statistical method to discover the structures of a given shape by building a dictionary of its local variations yielding a sparse description of the surface. The dictionary constructed from learned examples contains atoms which can be understood as approximating the patches using a sparse linear combination of atoms. To do so, they first presented a shape analysis approach based on the non-local analysis of local shape variations, called Local Probing Field (LPF). The LPF captures both local geometrical and topological variations of

Table 4: Summary of sparsity-based point cloud denoising methods

Method	Sparse reconstruction	Feature	Comments
MRPCA (2017) [62]	ℓ_1 based regularization	Robust PCA solver for convex optimization	May produce an over-sharpened result in the region of curved sharp edges
Leal et al. (2020) [63]	ℓ_1 based regularization	ℓ_1-median filtering with sparse ℓ_1 regularization	Needs parameter tuning
LPF (2018) [64]	Dictionary learning	Jointly learns the set of LPFs	Sensitive to the dictionary size

the shape. Then, by carefully optimizing the position and orientation of each descriptor, they provided a new tool to capture shape similarities, and gathered them into a geometrically relevant dictionary over which the shape decomposes sparsely.

Table 4 summarizes the sparsity-based methods discussed in this section, underlining the sparse reconstruction method, features, merit and demerit. These approaches may lead to over-smoothing or over-sharpening at high levels of noise because of the poor normal estimation. Another limitation of these approaches is the computational complexity. They always undergo several denoising iterations.

3.4. Non-local-based methods

Non-local-based methods, which originate from the image denoising field [69], are inspired by the geometric statistics which indicate that a number of surface patches sharing approximate geometric properties always exist within a 3D model. They exploit the non-local self-similarity that exists between patches in point clouds to better preserve fine shape features [70, 71, 72, 73]. While many local-based methods have been shown to produce promising results in denoising point clouds, such as RIMLS [53], EAR [58], MRPCA [62] and GPF [61], they are often criticized for over-smoothing since they only utilized the local structure information of each point, overlooking the non-local structures with self-similarity. However, it is challenging to adopt non-local similarity from regular images to irregular point clouds, and two main problems should be solved [74]: the representation of irregular local structures of point clouds, and the feature descriptor for robustly measuring the similarity between local

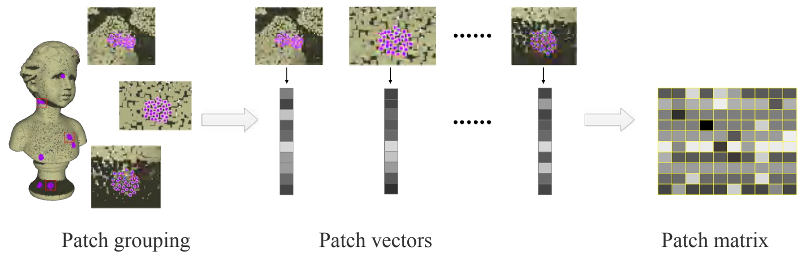


Figure 5: An illustration of patch grouping and patch matrix construction.

structures.

Generally, non-local-based methods partition the input point cloud into small patches, and pack similar patches into patch groups, which are normally represented by matrices, as illustrated in Figure 5. Let $\mathbf{X}$ be the matrix constructed from a patch group of the input noisy point cloud $\mathcal{P}$. For each patch group, since its patches share the most similar geometry structures than the other patches, the constructed patch matrix $\mathbf{X}$ should be low-rank and live in a low dimensional subspace. Thus, the point cloud denoising process can be transformed into a low-rank recovery problem. Let $\mathbf{A}$ be the low-rank matrix that we wish to recover, we can formulate the following low-rank recovery model based on a low-rank prior to efficiently recover the ground-truth matrix in each patch group [75]:

$$\min_{\mathbf{A}} \operatorname{rank}(\mathbf{A}) + \lambda \|\mathbf{X} - \mathbf{A}\|_F^2 \tag{18}$$

where $\operatorname{rank}(\cdot)$ is the rank function of a matrix, λ is a parameter which balances the noise measurement and the low-rankness, and F denotes the Frobenius norm. By solving Eq.(18), the denoised point cloud can be reconstructed from the recovered matrix $\mathbf{A}$.

Lu et al. [76] extend the non-local method to the normal field and proposed a robust normal estimation method for point clouds using a low-rank matrix approximation algorithm. They defined a local isotropic structure as a subset of points that are on the same isotropic surface with the representative normal. Non-local similar structures are searched and organized into a matrix in the context of isotropic surfaces rather than anisotropic surfaces. A low-rank matrix approximation algorithm was derived to estimate normals via weighted nuclear norm minimization on non-local similar structures.

Instead of packing similar patches in the normal field [76], Chen et al. [74] devised a multi-patch collaborative point cloud denoising method in the surface height field. They defined a rotation-invariant height-map patch (HMP) for each point by sampling the local surface height over a well-established local frame. Assumed that the constructed HMP group matrices satisfy the low-rank structure, they grouped its non-local similar patches and packed them into a height-map patch-group matrix. An improved low-rank matrix recovery method with graph constraints was proposed to filter noise. However, the HMP requires a high density of point cloud. Inspired by [76] and [74], Zhou et al. [77] projected the neighboring points of each point onto its normal, called normal height projection. They developed a structure-aware descriptor called projective height vector to capture the local height variations by normal height projection. The most similar non-local projective height vectors are grouped into a height matrix, which is then optimized by an improved weighted nuclear norm minimization.

In addition to the low-rank matrix approximation, non-local-based methods are also strongly related to the dictionary learning technique. Given a set of signals, these methods aim at finding a dictionary and a set of coefficients that best describe the signals. For example, in [64], Digne et al. proposed a shape analysis framework that reveals the shape similarities and its local dimensions. They consolidated local probing fields to represent the deformation of a pattern onto the local shape regardless of its local dimensionality. A geometrically relevant shape dictionary is then constructed as a new tool for sparse shape description.

Non-local methods have also been widely used in graph-based methods. For example, in [43] and [44], they partitioned the input point cloud into several overlapping surface patches. The k-NN algorithm is employed to search the nearest patches of each patch as neighbors. Finally, the patch-based graph is built over each pair of adjacent patches.

A summary of the approaches discussed in this section can be found in Table 5. The concept of non-local self-similarity combined with low-rank matrix recovery or dictionary learning technique has remained the potential idea for most of the state-of-the-art point cloud denoising methods, such as graph-based methods, sparsity-based methods, etc. However, these approaches may suffer from artefacts and performance degrada-

Table 5: Summary of non-local-based point cloud denoising methods

Method	Features	Comments
Lu et al. (2020) [76]	Local isotropic structure patch in the normal field; low-rank matrix approximation	Suitable for offline geometry processing
Chen et al. (2020) [74]	Rotation-invariant height-map patch in the surface height field; low-rank matrix approximation	Time-consuming; over-sharpens some non-sharp regions
Zhou et al. (2021) [77]	Projective height vector in the normal height field; low-rank matrix approximation	Structure preservation; sensitive to the density of point clouds
Hu et al. (2020) [43]	*k*-NN on patches; Graph laplacian regularizer using the Mahalanobis distance matrix	Requires small amounts of data for a stable estimation
Zeng et al. (2020) [44]	Low-dimensional manifold model; discrete patch-based graph laplacian regularizer	High complexity
LPF (2018) [64]	Consolidate local probing fields (LPF) as a local frame, dictionary learning	Sensitive to the dictionary size

tion when the input point cloud lacks in similar patches. Besides, the computational complexity of these methods is usually high.

4. Deep learning-based methods

Currently, driven by the success of deep learning in diverse computer vision, computer graphics, and image processing tasks, deep-learning-based methods have made their debut for point cloud denoising [7, 8, 9, 10, 11, 12, 13, 16, 17, 18]. After learning a mapping from the noisy inputs to their ground-truth counterparts in an offline stage, they can be automatically executed on new cases sharing similar geometry and noise characteristics of trained models in the runtime stage. Based on the availability of input data's labels, deep learning-based methods can be categorized into two types: supervised denoising methods and unsupervised denoising methods.

4.1. Supervised denoising methods

Supervised approaches rely on pairs of clean and noisy point clouds, which in practice are produced by adding noise (i.e. Gaussian noise, impulsive noise) to synthetic point clouds. According to the network architecture used for the feature learning of each point, supervised methods can be classified into PointNet-based, convolution-based, and encoder-decoder-based methods.

4.1.1. PointNet-based architecture

These methods employ the PointNet [78] or PointNet++ [79] to learn the feature of each point. The pioneering PointNet [78] directly takes point clouds as the input and learns features for each point independently with several shared Multi-Layer Perceptrons (MLPs), as shown in Figure 6. To capture the local structural information between points, Qi et al. [79] proposed a hierarchical network PointNet++ to capture fine geometric structures from the neighborhood of each point. Because of its simplicity and strong representation ability, recently, a few point cloud denoising methods have been developed based on PointNet.

Inspired by the PointNet [78], Guerrero1 et al. [80] proposed a local variant of PointNet, called PCPNet, to estimate local 3D shape properties in point clouds, which

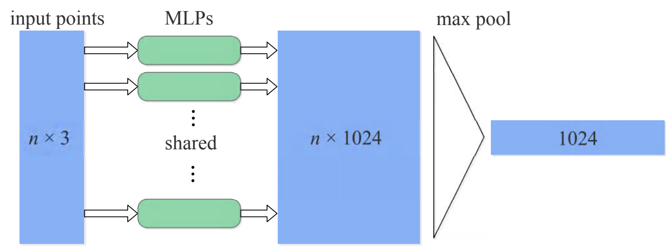
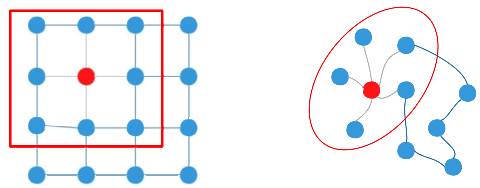


Figure 6: An illustration of the architecture of PointNet [78].

gives better results for shape details and is applicable to denoise point clouds. Since the PointNet extracts the local feature based on the position of a single point only, and do not include any neighborhood information, the proposed PCPNet is applied to local patches, centered at points with a fixed radius r proportional to the point cloud?s bounding box extent. They constrained the first spatial transformer of PointNet to the domain of rotations, and exchanged the max symmetric operation with a sum. The PCPNet learns a set of k non-linear functions in the local patch neighborhoods, and gives a k-dimensional feature vector per patch that can then be used to regress various local features. Based on the denoising architecture of PCPNet [80], Rakotosaona et al. [9] proposed a two-stage data-driven denoising architecture, called PointCleanNet, which involves a local outlier detection network and a denoising network. The local outlier detection network uses an architecture based on the PCPNet [80] to detect and remove outliers. The denoising network, with a similar architecture of the PCPNet [80] but a different loss, aims at reducing the noise level by estimating correcting displacement vectors, which results in the denoised point cloud.

In addition to the PointNet [78], Yu et al. [7] proposed the first deep-learning-based edge-aware network (EC-Net) based on the PointNet++ [79] for consolidating point clouds. They partitioned the point cloud into patches such that the points in a patch are geodesically close to one another over the underlying surface. The PointNet++ [79] was then used to encode the local geometry into a feature vector for each point in an input patch, followed by a feature expansion mechanism. Then, they regressed the



(a) (b)

Figure 7: An illustration of 2D convolution and graph convolution. (a) Convolution operations on an image. The neighbors of each pixel are ordered and have a fixed size. (b) Convolution operations on a graph. To get the hidden representation of the red node, a typical graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. The neighbors of each node in a graph are unordered and variable in size.

residual point coordinates and the point-to-edge distances from the expanded features. They formulated a regression component to simultaneously recover 3D point coordinates by adding the original point coordinates to the residual, and an edge-aware joint loss function to directly minimize distances from output points to 3D meshes and to edges.

4.1.2. Convolution-based architecture

The main drawback of the above PointNet-based techniques is that they work on individual points, and cannot exploit the local structure of the neighborhood well. Besides, these solutions are still limited by the fact that their networks cannot learn hierarchical feature representations, like standard Convolutional Neural Networks (CNN). Therefore, convolution-based architectures have been introduced to denoise point clouds. However, compared with kernels defined on 2D grid structures (e.g., images), it is difficult to design convolutional kernels for point clouds due to their irregular structures. Existing methods either project the input point cloud into 2D images (e.g. heightmap) [8, 12], or represent it as graphs followed by graph convolution operations [10, 11], as illustrated in Figure 7 [81].

Riccardo et al.[8] presented the first deep learning method for local point cloud processing with a fully differentiable, CNN-based deep learning architecture, called

PointProNet. Similar to [7], their approach was also designed based on local patches. Specifically, they represented a patch of geometry around a point as an oriented 2D heightmap that stores the distance to the sample points in the neighborhood along a given direction. Their main idea is to learn a local mapping that transforms each set of points extracted from a local patch to its consolidated version, where the output points sample the underlying surface very accurately and densely. To this end, two fully differentiable components called heightmap generation network (HGN) and heightmap denoising network (HDN), were designed to learn the mapping from a noisy patch of points to its consolidated version. The HGN learns a local coordinate frame for projection, projects the points onto the corresponding image plane with a projection module, and resamples the resulting 2D heightmap to obtain the regularly sampled image. Then, the HDN uses image convolutions to transform the noisy 2D heightmap into a denoised version. Finally, patches of points are denoised by projecting them to a learned local frame and using CNN in a supervised setup to move the points back to the surface. Similar to [8], Lu et al. [12] also developed a CNN-based feature-preserving normal estimation framework based on 2D heightmap for point cloud denoising. In the training stage, to meet the classical CNN requirement, they first represented each point and its neighbors as a 2D heightmap by a simple projection approach based on PCA. Then, they classified points into feature points and non-feature points via a classification network based on LeNet [82]. The normal estimation network with ResNet18 [83] as the backbone is then trained on feature points and non-feature points, respectively. In the testing stage, they employed the efficient point update algorithm [76] to match estimated normals.

Except for the CNN defined on 2D heightmap [8, 12], Pistilli et al. [10, 11] presented a Graph-convolutional Point Denoising Network (GPDNet) to denoise point clouds based on graph-convolutional layers. Graph convolution is a generalization of convolution to data that are defined over the nodes of a general graph rather than a grid. The proposed graph-convolutional layer has two inputs: a tensor representing a feature vector for each point, and a graph where nodes are points and edges represent similarities between points. Different from [9] which works on fixed-size patches, the proposed architecture has an elegant fully-convolutional behavior that can build hier-

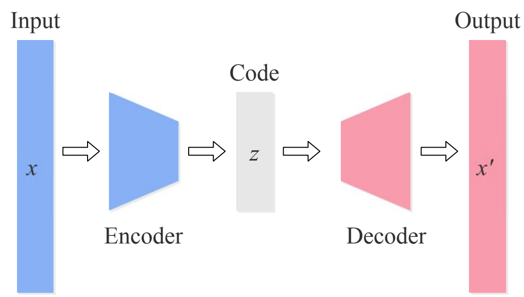


Figure 8: An illustration of the architecture of autoencoders. The encoder layer encodes the input signal x as a compressed representation z in a reduced dimension. The decoder layer decodes the encoded signal z back to the original dimension x'.

archies of local or non-local features to effectively regularize the denoising problem. They transformed the 3D space into an F-dimensional feature space gradually using a simple block composed of three single-point convolutions. Then, they employed a cascade of two residual blocks with an input-output skip connection to reduce vanishing gradient issues. Each residual block is composed of three graph-convolutional layers. They adopt a dynamic graph construction strategy by searching the k-NN of each point in terms of Euclidean distances in the feature space after every residual block. Finally, the last graph-convolutional layer projects the features back to the 3D space.

4.1.3. Autoencoder-based architecture

In addition to PointNet-based architectures and convolution-based architectures, the autoencoder architecture has also been employed to denoise point clouds. Figure 8 shows an example of autoencoders which consist of three layers: encoder, code, and decoder. The Pointfilter [13] proposed by Zhang et al. is a typical encoder-decoder architecture network for point cloud denoising. They straightforwardly took the raw neighboring points of each noisy point as input, and regressed a displacement vector to push this noisy point back to its ground truth position. Given a noisy patch, they used PCA for alignment and fed the aligned patch into the neural network. The encoder consists of two main parts: feature extractors and a collector.They employed the

Table 6: Summary of supervised point cloud denoising methods

Method	Network architecture	Loss Functions	Comments
PCPNet (2018) [80]	PointNet	Normals: Euclidean distance, angle difference; Curvatures: rectified error for training and the RMS for evaluation	Fail in the presence of large flat areas
PointCleanNet (2020) [9]	PCPNet; outlier detection network; denoising network	Proximity to the surface; regular distribution on the surface	Sensitive to outliers; point cloud shrinking
EC-Net (2018) [7]	PointNet++; Edge distance regression; coordinate regression	Surface loss; edge loss; repulsion loss; edge distance regression loss	Manually annotates sharp edges; fixed patch size; poor in tiny structures
PointProNet (2018) [8]	Heightmap generation network; heightmap denoising network	Distance between denoised and ground truth heightmap	Fails in large holes and high level noise; artifacts in extreme sharp edges
Lu et al. (2020) [12]	LeNet for classification; ResNet18 for normal estimation	Weighted ℓ_2 distance	Fails in severe noise and significant outliers
GPDNet (2020) [10, 11]	Graph-convolutional layer	MSE; MSE-SP	Robust to high level of noise and structured noise distributions
PF (2021) [13]	Encoder-decoder network	Projection distance	Requires ground-truth point normals in the training stage

PointNet [78] as the backbone in the feature extractors. The collector aggregates each point feature by a max-pooling layer. In the decoder module, a regressor constructed by three fully connected layers is employed to evaluate the displacement vectors with the latent representation vector as input. The trained neural network can automatically generate a set of clean points from the input noisy point cloud.

Table 6 summarizes the supervised denoising approaches discussed in this section. Although the differences between these methods are very large, they are essentially combinations of fundamental components, such as network architectures and loss functions. Supervised denoising approaches have achieved impressive results for point cloud denoising. However, these methods heavily depend on expensive training over massive datasets, most of which are generally produced by adding different noises to synthetic point clouds. They may also suffer from performance degradation when the data to be denoised deviates significantly from the training datasets.

4.2. Unsupervised denoising methods

Unsupervised methods strive to denoise point clouds directly without the need of pairs of clean and noisy point clouds, since it is unavailable and difficult to generate ground-truth data in various contexts [16]. In the image denoising field, Noise2Noise [84], and its extensions Noise2Self [85] and Noise2Void [86] have demonstrated how denoising can be achieved in an unsupervised manner without clean data. However, few related works have been reported for point cloud denoising in the literature. Existing unsupervised denoising methods commonly adopt an autoencoder-based architecture. A summary of the network architecture and loss functions employed in the unsupervised approaches discussed in this section can be found in Table 7.

TotalDenoising [16] is the first unsupervised learning method for denoising point clouds without needing access to clean examples, and not even noisy pairs. It is based on the assumption that points with denser surroundings are closer to the underlying surface. They employed an unstructured encoder-decoder based on Monte Carlo convolution that maps the point cloud to itself in combination with a spatial locality and a bilateral appearance prior. By imposing priors, the method can directly train on noisy data without needing ground truth examples or even noisy pairs. However, this

Table 7: Summary of unsupervised point cloud denoising methods

Method	Network architecture	Loss Functions	Comments
TotalDenoising (2019) [16]	Unstructured encoder-decoder based on Monte Carlo convolution	ℓ_2 loss	Cannot preserve share features; sensitive to outliers
Chen et al. (2020) [17]	Encoder: PointNet; Decoder: folding module, graph-topology-inference module, graph-filtering module	Augmented Chamfer distance	Effective in reconstruction, visualization, and transfer classification
Luo et al. (2020) [18]	Encoder: DGCNN; Decoder: MLP	Supervised training loss: Chamfer distance, Earth Mover's distance; unsupervised training loss: ℓ_2 loss	Trained in either a supervised or unsupervised fashion

method cannot preserve sharp features due to a lack of sharp feature information during the training stage. To explore geometric structures of point clouds, Chen et al. [17] proposed a deep autoencoder with graph topology inference and filtering to achieve compact representations of unorganized point clouds in an unsupervised manner. The encoder adopts similar architectures as in PointNet [78], and the graph-based decoder consists of three novel modules: the folding module, the graph-topology-inference module, and the graph-filtering module. The folding module folds a canonical 2D lattice to the underlying surface of a 3D point cloud, achieving coarse reconstruction. The graph-topology-inference module learns a graph topology to represent pairwise relationships between 3D points, pushing the latent code to preserve both coordinates and pairwise relationships of points in 3D point clouds The graph-filtering module, which is designed based on a learnable graph topology, couples the above two modules, refining the coarse reconstruction through a learned graph topology to obtain the final reconstruction.

Different from [16] and [17] which infer the displacement of noisy points from the underlying surface and implicitly recover the point cloud, Luo et al. [18] proposed to explicitly learn the underlying manifold of a noisy point cloud for denoising via an autoencoder-like network. The encoder, which builds on the Dynamic Graph

CNN (DGCNN) [87], learns both local and non-local feature representations of each point and then samples points with low noise that tend to be closer to the underlying surfaces via an adaptive differentiable pooling operation. Then, the decoder infers underlying patch manifolds by transforming each sampled point along with its embedded neighborhood feature to a local surface. The clean point cloud is finally obtained by resampling on the reconstructed manifold. Their network can be trained end-to-end in either a supervised or unsupervised fashion because of the included unsupervised training loss.

5. Quality assessment

Different quality assessment metrics have been proposed to evaluate the performance of various point cloud denoising algorithms. These methods can be classified into subjective assessment methods and objective assessment methods.

5.1. Subjective assessment methods

Subjective assessment of point clouds is generally performed through visual comparisons [54, 61, 88, 89, 90, 91, 92]. According to the representation of rendered point clouds, these methods can be divided into raw point clouds based and surface reconstruction based methods.

5.1.1. Raw point clouds based methods

This kind of method renders original noisy point clouds and corresponding denoised ones produced by different denoising algorithms from the same view. The point of view should be carefully selected to show denoised regions, such as sharp edges and corners [32, 33]. The performance of different denoising algorithms are evaluated by comparing the rendered raw point clouds subjectively. However, subjective evaluation may be difficult if point clouds are displayed directly to the observer without the surface reconstruction [88].

Table 8: Objective point cloud quality assessment methods

Method	Feature
HD [98, 99]	Point-to-point
RMSE [98]	Point-to-point
PSNR [100, 101]	Point-to-point
CD [13]	Point-to-point
Javaheri [96, 100]	Point-to-plane
Alexiou [102]	Plane-to-plane
Javaheria [103]	Point-to-distribution

5.1.2. Surface reconstruction based methods

This type of method performs the quality evaluation after reconstructing denoised point clouds [54, 61, 89, 90, 91, 92]. Different surface reconstruction algorithms may have different robustness against introduced degradations [93]. For example, Xu et al. [54] employed the surface reconstruction toolbox [94] to reconstruct 3D surfaces of denoised point clouds. Lu et al. [61] adopted a feature preserving surface reconstruction method (i.e., RIMLS [53] in Meshlab [95]) to reconstruct denoised results. In [96], Javaheri et al. rotated the point cloud around a vertical and horizontal axis, producing a slowly continuously changing point of view. Then, they employed the Poisson surface reconstruction algorithm [97] to generate 2D video sequences from different point of views. Note that all point clouds should be reconstructed via identical surface reconstruction parameters for fair comparison.

5.2. Objective assessment methods

Although subjective assessment has higher accuracy, it is typically rather cumbersome, time-consuming, and expensive. Therefore, most works available in the literature use objective quality metrics. Objective assessment methods can be distinguished into three categories: (a) point-to-point, (b) point- to-plane, and (c) point-to-distribution.Table 8 summarizes objective quality assessment metrics introduced in this section.

5.2.1. Point-to-point distance

Point-to-point distance metrics compute the distance between points in the noisy point cloud and points in the corresponding clean point cloud [96].

Hausdorff Distance (HD). The classical HD generally falls into two categories: asymmetric Hausdorff distance and symmetric Hausdorff distance [98]. Let $e(p, \mathcal{A})$ denote the distance from a point p to the point cloud $\mathcal{A}$:

$$e(p, \mathcal{A}) = \min_{p_i^{\mathcal{A}} \in \mathcal{A}} d(p, p_i^{\mathcal{A}}), \tag{19}$$

where $d(,)$ is the Euclidian distance, and $p_i^{\mathcal{A}}$ is the i-th point of $\mathcal{A}$. Then, the *asymmetric Hausdorff distance* between two point clouds $\mathcal{A}$ and $\mathcal{B}$ is defined as:

$$H_{asy}(\mathcal{A}, \mathcal{B}) = \max_{p_i^{\mathcal{A}} \in \mathcal{A}} e(p_i^{\mathcal{A}}, \mathcal{B}). \tag{20}$$

The *symmetric Hausdorff distance* is then defined as follows:

$$H_{sym}(\mathcal{A}, \mathcal{B}) = \max\{H_{asy}(\mathcal{A}, \mathcal{B}), H_{asy}(\mathcal{B}, \mathcal{A})\}. \tag{21}$$

However, the above classical Hausdorff distance based geometry quality metrics are sensitive to outliers. Besides, points with a large error magnitude will dominate the final quality score even for cases where these points may not even be visible due to self-occlusion, which will lead to low objective-subjective correlation. Therefore, Javaheri et al. extended the classical Hausdorff distance and proposed a quality evaluation strategy called the *generalized Hausdorff distance* [99]

$$H_{gen}(\mathcal{A}, \mathcal{B}) =^{per} K^{th}_{p_i^{\mathcal{A}} \in \mathcal{A}} d(p_i^{\mathcal{A}}, \mathcal{B}), \tag{22}$$

where ${}^{per}K^{th}_{p_i^{\mathcal{A}} \in \mathcal{A}}$ is the K^{th} ranked distance. Instead of taking the maximum distance over all the distances as in the classical Hausdorff distance, the generalized Hausdorff distance for the rank K is computed using only the K lowest distance values after ranking all distances in ascending order. Therefore, it can be used to identify the best performing quality metric in terms of correlation with the Mean Opinion Score (MOS) scores obtained from a subjective test campaign.

Root Mean Square Error (RMSE). RMSE is the square root of the mean square error (MSE), which is measured by averaging the distance from all the points in $\mathcal{B}$ to their nearest neighbor points in the reference point cloud $\mathcal{A}$ [96]:

$$\mathrm{RMSE}(\mathcal{A}, \mathcal{B}) = \sqrt{\mathrm{MSE}(\mathcal{A}, \mathcal{B})} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \| p_i^{\mathcal{A}} - p_i^{\mathcal{B}} \|^2}, \tag{23}$$

where n is the number of points, and $p_i^{\mathcal{B}}$ is the point of $\mathcal{B}$ corresponding to the point $p_i^{\mathcal{A}}$ of $\mathcal{A}$.

Mean City-block Distance (MCD). MCD is similar to MSE with ℓ_2 norm replaced with ℓ_1 norm [44]:

$$\mathrm{MCD}(\mathcal{A}, \mathcal{B}) = \frac{1}{n} \sum_{i=1}^{n} |p_i^{\mathcal{A}} - p_i^{\mathcal{B}}|. \tag{24}$$

Peak signal to noise ratio (PSNR). PSNR is defined as 10log 10 of the ratio of a peak value M to the square root of MSE between the noisy and the reference point cloud [100, 101]. The MSE proposed so far could be reported as a metric, and sometimes it is preferred because it carries the physical units in 3D space. However, people often found it hard to understand MSEs between multiple point clouds. Therefore, PSNR is introduced to convert MSEs into PSNR numbers, normalizing the metrics with respect to the peak value M. Mathematically, PSNR is defined as follows :

$$\mathrm{PSNR}(\mathcal{A}, \mathcal{B}) = 10 \log 10 \left(\frac{M^2}{MSE(\mathcal{A}, \mathcal{B})} \right). \tag{25}$$

The peak value M is usually defined as the diagonal distance of a bounding box of the point cloud.

Chamfer distance (CD). The CD metric can be viewed as an indicator function that measures the similarity between two point clouds. It finds the nearest neighbor in the other point cloud, and sums the squared distances up [9, 13, 16]. CD is defined as :

$$\begin{aligned} \mathrm{CD}(A, \mathcal{B}) = & \frac{1}{|\mathcal{A}|} \sum_{p_i^{\mathcal{A}} \in \mathcal{A}} \min_{p_j^{\mathcal{B}} \in \mathcal{B}} (\| p_i^{\mathcal{A}} - p_j^{\mathcal{B}} \|_2^2) + \\ & \frac{1}{|\mathcal{B}|} \sum_{p_j^{\mathcal{B}} \in \mathcal{B}} \min_{p_i^{\mathcal{A}} \in \mathcal{A}} (\| p_j^{\mathcal{B}} - p_i^{A} \|_2^2). \end{aligned} \tag{26}$$

5.2.2. Point-to-plane distance

The point-to-plane distance first computes the normal of the surface at every point in the reference point cloud as an indication of the local surface. The displacement of every corresponding point in the noisy point cloud is then projected onto the normal to calculate the point-to-plane distance [54, 96, 100] . The steps of computing point-to-plane distance are listed as follows:

- For each point $p_i^{\mathcal{A}}$ in point cloud $\mathcal{A}$, find the nearest neighbor point $p_j^{\mathcal{B}}$ in point cloud $\mathcal{B}$ as its corresponding point.
- Compute the unit normal vector $\mathbf{n}_i^{\mathcal{A}}$ on point $p_i^{\mathcal{A}}$ in the reference point cloud $\mathcal{A}$, if available. Otherwise, the normal vector would be estimated on the fly using a state-of-the-art method [104].
- Compute the error vector $E(i, j)$ by connecting $p_i^{\mathcal{A}}$ to $p_j^{\mathcal{B}}$.
- Project the error vector $E(i, j)$ along the normal direction $\mathbf{n}_i^{\mathcal{A}}$ to get the point-to-plane error:

$$e(\mathcal{A}, \mathcal{B}) = \frac{1}{|\mathcal{A}|} \sum_{p_i^{\mathcal{A}} \in \mathcal{A}} (E(i, j) \cdot \mathbf{n}_i^{\mathcal{A}})^2. \tag{27}$$

5.2.3. Point-to-distribution distance

The point-to-distribution distance adopts the correspondence between a point and the distribution of points from a small point cloud region [103]. The basic idea is to statistically characterize the point cloud surface, notably through the covariance of points within some local regions. It employs the Mahalanobis distance to measure the distance between a point and a distribution. In addition, it is accurate when the noisy and reference point clouds have different characteristics, such as precision, density, and structure.

6. Experimental results and discussions

In this section, extensive experiments are conducted to investigate the performances of selected point cloud denoising methods.

6.1. Experimental settings

The compared popular and state-of-the-art point cloud denoising algorithms include RIMLS [53], CLOP [60], GPF [61], LPF [64], PointCleanNet [9], Pointfilter

[13], and AD [54]. The CLOP [60], GPF [61], LPF [64],PointCleanNet [9], Pointfilter [13], and AD [54] are implemented from the source codes provided by their respective authors for fairness. For RIMLS [53], we used the corresponding function integrated into the Meshlab software [95]. For fair comparisons and visualization purposes, we manually tune the main parameters of each comparison algorithm to achieve as good visual results as possible (PointCleanNet [9] and Pointfilter [13] have fixed parameters). The PSNR introduced in Section 5.2 is employed to evaluate the quality of denoising results. 25 typical 3D clean point clouds with different features and their corresponding noisy models are synthesized by adding Gaussian noise with a standard deviation of 1% of the clean models' bounding box diagonal length.

6.2. Visual Comparisons

Figure 9 presents the visual assessment of six denoised point clouds. Overall, it can be observed that Pointfilter [13] and AD [54] generate visually better results in terms of noise removal and feature preservation, as shown in Figure 9. GPF [61] yields positional distortion around sharp edges, and it depends greatly on the capability of normal filters which become less robust when meeting large noise. RIMLS [53] may smooth out some sharp features when using a large filter scale for removing noise. And, it requires quality normals and trial-and-error parameter tuning. Despite that the CLOP [60] is good at generating smooth results, they rely on the support radius to generate desirable results and still fail to retain sharp features. For noisy models with both smooth and sharp edges, such as the cube_sphere model, all these methods seem to encounter difficulty at balancing the trade-off between preserving sharp features and smooth areas to varying degrees.

6.3. Quantitive Comparisons

Table 9 summarizes quantitative results of different methods. There is no doubt that Pointfilter [13] and AD [54] achieve a better filtering performance. As a non-deep-learning-based method, AD [54] generates comparable results to Pointfilter [13]. For non-deep-learning-based methods, they may require trial-and-error parameter tuning to obtain satisfactory results, which is tedious, time-consuming, and especially difficult

Table 9: PSNR results of different methods on 25 noisy point clouds, which are synthesized by adding Gaussian noise with a standard deviation of 1% of their clean models' bounding box diagonal length (dB)

Models(points)	RIMLS [53]	CLOP [60]	GPF [61]	LPF [64]	PointCleanNet [9]	Pointfilter [13]	AD [54]
bunny(35947)	48.1857	48.4146	42.8261	48.7172	48.3412	**49.2297**	49.0393
Cube(24578)	45.5341	44.5173	42.2118	45.268	44.2063	45.5069	**45.8145**
Eight(29580)	45.5466	44.6984	36.5813	45.8541	44.6491	**45.8858**	45.2505
Joint(69544)	46.2033	45.0441	41.6862	46.4824	45.1500	**46.7316**	46.5271
Kitten(24956)	45.4730	44.4748	40.0056	45.8713	44.4175	**46.0214**	45.6003
Plane-sphere(18531)	48.8509	48.1370	42.7978	49.4534	47.9223	**49.4595**	49.4422
block(32370)	42.9914	42.3013	42.1779	43.1771	42.2128	**43.6166**	43.3533
child(50002)	47.6989	48.2588	43.3147	48.3816	48.4493	48.6345	**49.1511**
chinese_lion(50003)	41.9390	43.2764	38.6749	42.7804	43.6654	43.3093	**44.1240**
eros100K(50002)	45.2728	45.3624	39.9963	45.5562	45.8594	44.8889	**46.1356**
fertility(13971)	46.3205	46.3539	44.1436	46.7658	46.0688	**47.4635**	47.3425
genus3(29663)	47.7402	48.3228	44.3067	48.4552	48.3011	**49.6790**	49.2529
grayloc(34274)	45.0994	46.3928	41.5385	45.8805	46.7177	**47.4295**	47.2238
horse(48485)	40.1681	45.2767	35.4816	42.1123	45.2834	45.9057	**46.1371**
part_Lp(25994)	47.6773	46.6724	43.5530	47.7082	46.6810	**48.1388**	47.9187
pulley(50000)	43.4536	45.0524	42.3665	44.7348	45.3355	45.1186	**45.3711**
bumpy_torus(16815)	**46.3100**	44.4732	39.7064	45.8164	44.7101	44.9595	45.6648
pyramid(12290)	45.1285	44.1825	44.6298	45.1055	44.3652	45.4668	**45.7149**
rolling_stage(49988)	47.9001	48.2946	41.5853	48.5863	48.5864	49.0758	**49.6429**
screwdriver(27152)	36.1560	41.1380	36.6015	38.5769	42.5170	**43.6648**	42.6747
sharp_sphere(28051)	45.4127	44.3338	42.1241	45.3046	45.0015	45.6641	**45.6852**
smooth-feature(24871)	44.6808	43.4985	44.7482	44.6553	43.3647	44.7917	**44.9690**
sphere(16386)	46.2519	44.6893	42.1789	45.6329	44.8021	46.2177	**46.2686**
star(18146)	48.2372	46.8414	41.4964	48.1927	46.8238	**48.3193**	48.0599
trim-star(24467)	48.1751	47.5566	46.2349	48.5228	47.6599	**49.0233**	48.7306

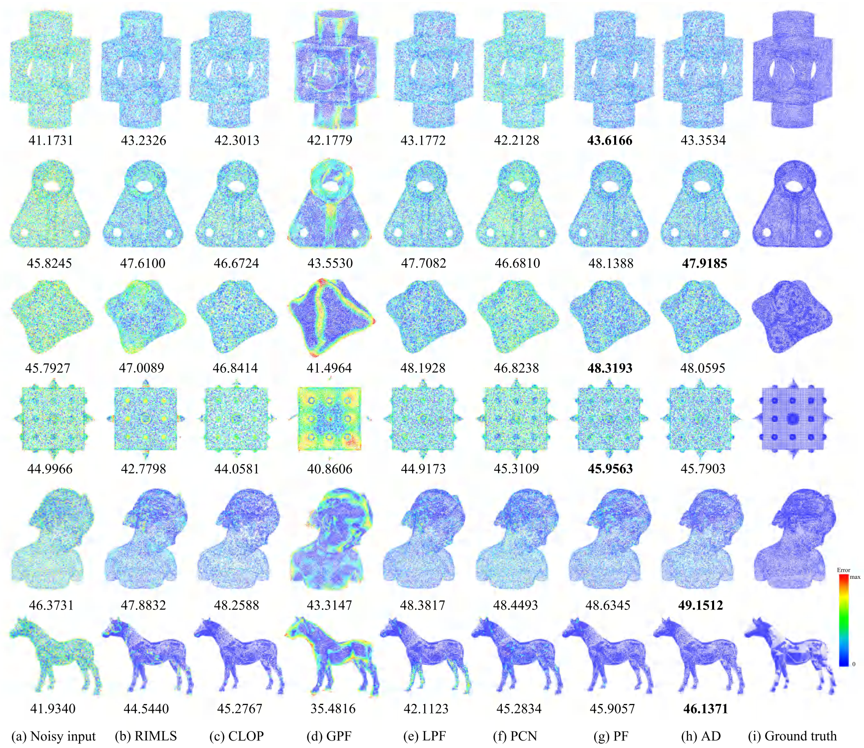


Figure 9: Comparison of denoising results on six noisy point clouds with the color of each point indicating its PSNR. The noisy point clouds are synthesized by adding Gaussian noise with a standard deviation of 1% of their clean models' bounding box diagonal length. From top to bottom: block, part_Lp, star, cube_sphere, child, and horse. From the left column to the right: the noisy input, the denoising results of RIMLS [53], CLOP [60], GPF [61], LPF [64], PointCleanNet (PCN) [9], Pointfilter (PF) [13], and AD [54].

for users who do not have any background knowledge. In contrast, deep-learning-based methods, such as Pointfilter [13] and PointCleanNet [9], are automatic and easy to use. The main challenges for these approaches are that the training is very time-consuming, and the training results are greatly affected by the training set (i.e. the features not included in the training set cannot be recovered).

7. Open challenges and future directions

In this section, we discuss several open challenges and future directions of point cloud denoising.

7.1. Real-world noise

The majority of existing point cloud denoising methods were designed for specific kinds of noise with different levels, such as Gaussian noise [13]. In particular, most deep-learning-based techniques learned denoising models from pairs of clean and synthetic noisy point clouds [7, 8, 9, 10, 11, 12, 13]. However, the assumption of specific noise is too ideal to be true for real-world noisy point clouds, where the noise is much more complex and varies with different scenes and sensors [5]. Although there have been a few unsupervised methods developed for real-world noisy point cloud denoising [15, 16, 17, 18]. These algorithms aim to directly learn latent representations for denoising from the noisy point cloud in an unsupervised manner. However, their overall performance on real-world noise is still limited. Therefore, it is desirable to further investigate the problem of real-world noisy point cloud denoising.

7.2. Automatic parameter tuning

Most classical point cloud denoising methods contain user-defined parameters, and thus parameter tuning is of great importance to these algorithms for faster convergence rate and better visual quality of denoised point clouds. In most cases, users must carefully tune these parameters, which are empirically assigned as constant numbers, to achieve the best quality results for *each* input noisy point cloud [13, 74]. It is a quite labor-intensive and time-consuming job that largely reduces the applicability. Therefore, how to automatically optimize parameters is an exciting and challenging problem.

7.3. Benchmarking point cloud denoising algorithms

Although an extensive variety of point cloud denoising algorithms has been proposed, no single method performs consistently well on all degraded point clouds. For applications where a large set of heterogeneous noisy point clouds is processed, it is preferable to optimize the denoising algorithm for each point cloud individually. However, it is not practical for a user to manually select a denoising algorithm for each point cloud. Therefore, benchmarking point cloud denoising algorithms is of great importance to yield the perceptually most satisfying result. However, to the best of our knowledge, no related work has been reported for this open challenge in the literature.

8. Conclusion

In this paper, we have made an earnest effort to give an up-to-date survey of point cloud denoising techniques proposed in the past five years. These methods have been mainly divided into three categories: filter-based, optimization-based, and deep learning-based methods. While it is nearly impossible to cover all of them, we have covered each category with several representative methods. We have also reviewed subjective and objective assessment metrics for evaluating the quality of denoised point clouds. A comprehensive performance comparison of some representative or state-of-the-art methods has been presented. Last but not least, we have discussed some open challenging issues and listed potential research directions.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1004904, and in part by the Fundamental Research Funds for the Central Universities under Grant 30918012203. The authors would like to acknowledge the helpful comments and kindly suggestions provided by anonymous referees.

References

[1] L. Cheng, Z. Wei, M. Sun, S. Xin, A. Sharf, Y. Li, B. Chen, C. Tu, Deeppipes: Learning 3d pipelines reconstruction from point clouds, Graphical Models 111 (2020) 101079.

[2] Y. Cui, R. Chen, W. Chu, L. Chen, D. Tian, Y. Li, D. Cao, Deep learning for image and point cloud fusion in autonomous driving: A review, IEEE Transactions on Intelligent Transportation Systems (2021) 1–18doi:10.1109/TITS.2020.3023541.

[3] M. Liu, Robotic online path planning on point cloud, IEEE Transactions on Cybernetics 46 (5) (2016) 1217–1228.

[4] L. Han, T. Zheng, Y. Zhu, L. Xu, L. Fang, Live semantic 3d perception for immersive augmented reality, IEEE Transactions on Visualization and Computer Graphics 26 (5) (2020) 2012–2022.

[5] H. Chen, J. Shen, Denoising of point cloud data for computer-aided design, engineering, and manufacturing, Engineering with Computers 34 (2018) 523–541.

[6] X. Han, J. S. Jin, M. Wang, W. Jiang, L. Gao, L. Xiao, A review of algorithms for filtering the 3d point cloud, Signal Processing: Image Communication 57 (2017) 103–112.

[7] LequanYu, X. Li, C.-W. Fu, D. Cohen-Or, P.-A. Heng, Ec-net: an edge-aware point set consolidation network, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 386–402.

[8] R. Roveri, A. C. Öztireli, I. Pandele, M. Gross, Pointpronets: Consolidation of point clouds with convolutional neural networks, Computer Graphics Forum 37 (2) (2018) 87–99.

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

[10] F. Pistilli, G. Fracastoro, D. Valsesia, , E. Magli, Learning graph-convolutional representations for point cloud denoising, in: Proceedings of the European Conference on Computer Vision (ECCV), 2020, pp. 103–118.

[11] F. Pistilli, G. Fracastoro, D. Valsesia, , E. Magli, Learning robust graph-convolutional representations for point cloud denoising, IEEE Journal of Selected Topics in Signal Processing 15 (2) (2021) 402–414.

[12] D. Lu, X. Lu, Y. Sun, J. Wang, Deep feature-preserving normal estimation for point cloud filtering, Computer-Aided Design 125 (2020) 1–12.

[13] D. Zhang, X. Lu, H. Qin, Y. He, Pointfilter: Point cloud filtering via encoder-decoder modeling, IEEE Transactions on Visualization and Computer Graphics 27 (3) (2021) 2015–2027.

[14] C. Chen, Z. Xiong, X. Tian, Z.-J. Zha, F. Wu, Real-world image denoising with deep boosting, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (12) (2020) 3071–3087.

[15] V. Sterzentsenko, L. Saroglou, A. Chatzitofis, S. Thermos, N. Zioulis, A. Doumanoglou, D. Zarpalas, P. Daras, Self-supervised deep depth denoising, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 1242–1251.

[16] P. Hermosilla, T. Ritschel, T. Ropinski, Total denoising: Unsupervised learning of 3d point cloud cleaning, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 52–60.

[17] S. Chen, C. Duan, Y. Yang, D. Li, C. Feng, D. Tian, Deep unsupervised learning of 3d point clouds via graph topology inference and filtering, IEEE transactions on image processing 29 (2020) 3183–3198.

[18] S. Luo, W. Hu, Differentiable manifold reconstruction for point cloud denoising, in: Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 1330–1338.

[19] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun, Deep learning for 3d point clouds: A survey, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020) 1–1doi:10.1109/TPAMI.2020.3005434.

[20] L. Akoglu, H. Tong, D. Koutra, Graph based anomaly detection and description: A survey, Data Mining and Knowledge Discovery 29 (3) (2015) 626–688.

[21] A. Boukerche, L. Zheng, O. Alfandi, Outlier detection: Methods, models, and classification, ACM Computing Surveys 53 (3) (2020) 55:1–55:37.

[22] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in: Sixth International Conference on Computer Vision, 1998, pp. 839–846.

[23] J. Digne, C. D. Franchis, The bilateral filter for point clouds, Image Processing On Line 7 (2017) 278–287.

[24] F. Zhang, C. Zhang, H. Yang, L. Zhao, Point cloud denoising with principal component analysis and a novel bilateral filter, Traitement du Signal 36 (5) (2019) 393–398.

[25] S. Fleishman, I. Drori, D. Cohen-Or, Bilateral mesh denoising, ACM Transactions on Graphics 22 (3) (2003) 950–953.

[26] Openmp (2021).
URL https://www.openmp.org

[27] K. He, J. Sun, X. Tang, Guided image filtering, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (6) (2013) 1397–1409.

[28] X. Han, J. S. Jin, M. Wang, W. Jiang, Guided 3d point cloud filtering, Multimedia Tools and Applications 77 (2018) 17397–17411.

[29] S. K. Yadav, U. Reitebuch, M. Skrodzki, E. Zimmermann, K. Polthier, Constraint-based point set denoising using normal voting tensor and restricted quadratic error metrics, Computers & Graphics 74 (2018) 234–243.

[30] Y. Zheng, G. Li, S. Wu, Y. Liu, Y. Gao, Guided point cloud denoising via sharp feature skeletons, The Visual Computer 33 (2017) 857–867.

[31] X. Han, J. S. Jin, M. Wang, W. Jiang, Iterative guidance normal filter for point cloud, Multimedia Tools and Applications 77 (2018) 16887–16902.

[32] Z. Liu, X. Xiao, S. Zhong, W. Wang, Y. Li, L. Zhang, Z. Xie, A feature-preserving framework for point cloud denoising, Computer-Aided Design 127 (2020) 102857:1–102857:12.

[33] Y. Zheng, G. Li, X. Xu, S. Wu, Y. Nie, Rolling normal filtering for point clouds, Computer Aided Geometric Design 62 (2018) 16–28.

[34] Y. Sun, H. Chen, J. Qin, H. Li, M. Wei, H. Zong, Reliable rolling-guided point normal filtering for surface texture removal, Computer Graphics Forum 38 (7) (2019) 721–732.

[35] Q. Zhang, X. Shen, L. Xu, J. Jia, Rolling guidance filter, in: Computer Vision – ECCV 2014, 2014, pp. 815–830.

[36] E. J. Candès, X. Li, Y. Ma, J. Wright, Robust principal component analysis?, Journal of the ACM 58 (3).

[37] X. Sun, P. L. Rosin, R. Martin, F. Langbein, Fast and effective feature-preserving mesh denoising, IEEE Transactions on Visualization and Computer Graphics 13 (5) (2007) 925–938.

[38] P.-S. Wang, X.-M. Fu, Y. Liu, X. Tong, S.-L. Liu, B. Guo, Rolling guidance normal filter for geometric processing, ACM Transactions on Graphics 34 (6) (2015) 173:1–173:9.

[39] M. A. Fischler, R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, in: Readings in Computer Vision, Morgan Kaufmann, San Francisco (CA), 1987, pp. 726–740.

[40] S. Zhang, S. Cui, Z. Ding, Hypergraph spectral analysis and processing in 3d point cloud, IEEE Transactions on Image Processing 30 (2021) 1193–1206.

[41] C. Duan, S. Chen, J. Kovacevic, Weighted multi-projection: 3d point cloud denoising with estimated tangent planes, in: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2018, pp. 725–729.

[42] M. A. Irfan, E. Magli, Exploiting color for graph-based 3d point cloud denoising, Journal of Visual Communication and Image Representation 75 (2021) 103027.

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

[44] Z. Jin, C. Gene, N. Michael, P. Jiahao, Y. Cheng, 3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model, IEEE Transactions on Image Processing 29 (2020) 3474–3489.

[45] C. Dinesh, G. Cheung, I. V. Bajić, Point cloud denoising via feature graph laplacian regularization, IEEE Transactions on Image Processing 29 (2020) 4143–4158.

[46] A. Ortega, P. Frossard, J. Kova?evi?, J. M. F. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, Proceedings of the IEEE 106 (5) (2018) 808–828.

[47] S. Osher, Z. Shi, W. Zhu, Low dimensional manifold model for image processing, SIAM Journal on Imaging Sciences 10 (4) (2017) 1669–1690.

[48] J. Zeng, G. Cheung, A. Ortega, Bipartite approximation for graph wavelet signal decomposition, IEEE Transactions on Signal Processing 65 (20) (2017) 5466–5480.

[49] S. Zhang, Z. Ding, S. Cui, Introducing hypergraph signal processing: Theoretical foundation and practical applications, IEEE Internet of Things Journal 7 (1) (2020) 639–660. doi:10.1109/JIOT.2019.2950213.

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

[51] D. Levin, The approximation power of moving least-squares, Mathematics of Computation 67 (224) (1998) 1517–1531.

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

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

[54] Z. Xu, A. Foi, Anisotropic denoising of 3d point clouds by aggregation of multiple surface-adaptive estimates, IEEE Transactions on Visualization and Computer Graphics 27 (6) (2021) 2851–2868.

[55] V. Katkovnik, A new method for varying adaptive bandwidth selection, IEEE Transactions on Signal Processing 47 (9) (1999) 2567–2571.

[56] Y. Lipman, D. Cohen-Or, D. Levin, H. Tal-Ezer, Parameterization-free projection for geometry reconstruction, ACM Transactions on Graphics 26 (3) (2007) 22:1–22:5.

[57] H. Huang, D. Li, H. Zhang, A. Uri, D. Cohen-Or, Consolidation of unorganized point clouds for surface reconstruction, ACM Transactions on Graphics 28 (5) (2009) 176:1–176:7.

[58] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, H. Z. (Richard), Edge-aware point set resampling, ACM Transactions on Graphics 32 (1) (2013) 9:1–9:12.

[59] B. Liao, C. Xiao, L. Jin, HongboFu, Efficient feature-preserving local projection operator for geometry reconstruction, Computer-Aided Design 45 (5) (2013) 861–874.

[60] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, M. Wimmer, Continuous projection for fast l¡sub¿1¡/sub¿ reconstruction, ACM Transactions on Graphics 33 (4) (2014) 47:1–47:13.

[61] X. Lu, S. Wu, H. Chen, S.-K. Yeung, W. Chen, M. Zwicker, Gpf: Gmm-inspired feature-preserving point set filtering, IEEE Transactions on Visualization and Computer Graphics 24 (8) (2018) 2315–2326.

[62] E. Mattei, A. Castrodad, Point cloud denoising via moving rpca, Computer Graphics Forum 36 (8) (2017) 123–137.

[63] E. Leal, G. Sanchez-Torres, J. W. Branch, Sparse regularization-based approach for point cloud denoising and sharp features enhancement, Sensors 20 (11).

[64] J. Digne, S. Valette, R. Chaine, Sparse geometric representation through local shape probing, IEEE Transactions on Visualization and Computer Graphics 24 (7) (2018) 2238–2250.

[65] Z. Zhang, Y. Xu, J. Yang, X. Li, D. Zhang, A survey of sparse representation: Algorithms and applications, IEEE Access 3 (2015) 490–530.

[66] Y. Sun, S. Schaefer, W. Wang, Denoising point sets via l0 minimization, Computer Aided Geometric Design 35-36 (2015) 2–15.

[67] L. Xu, R. Wang, J. Zhang, Z. Yang, J. Deng, F. Chen, L. Liu, Survey on sparsity in geometric modeling and processing, Graphical Models 82 (2015) 160–180.

[68] H. Avron, A. Sharf, C. Greif, D. Cohen-Or, L1-sparse reconstruction of sharp point set surfaces, ACM Transactions on Graphics 29 (5) (2010) 135:1–135:12.

[69] A. Buades, B. Coll, J.-M. Morel, A non-local algorithm for image denoising, in: 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, 2005, pp. 60–65.

[70] Q. Zheng, A. Sharf, G. Wan, Y. Li, N. J. Mitra, D. Cohen-Or, B. Chen, Non-local scan consolidation for 3d urban scenes, ACM Transactions on Graphics 29 (4) (2010) 94:1–94:9.

[71] J. Digne, Similarity based filtering of point clouds, in: 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2012, pp. 73–79.

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

[73] J. Digne, R. Chaine, S. Valette, Self-similarity for accurate compression of point sampled surfaces, Computer Graphics Forum 33 (2) (2014) 155–164.

[74] H. Chen, M. Wei, Y. Sun, X. Xie, J. Wang, Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint, IEEE Transactions on Visualization and Computer Graphics 26 (11) (2020) 3255–3270.

[75] M. Wei, J. Huang, X. Xie, L. Liu, J. Wang, J. Qin, Mesh denoising guided by patch normal co-filtering via kernel low-rank recovery, IEEE Transactions on Visualization and Computer Graphics 25 (10) (2019) 2910–2926. doi:10.1109/TVCG.2018.2865363.

[76] X. Lu, S. Schaefer, J. Luo, L. Ma, Y. He, Low rank matrix approximation for 3d geometry filtering, IEEE Transactions on Visualization and Computer Graphics (2020) 1–1doi:10.1109/TVCG.2020.3026785.

[77] Y. Zhou, R. Chen, Y. Zhao, X. Ai, G. Zhou, Point cloud denoising using non-local collaborative projections, Pattern Recognition 120 (2021) 108128.

[78] R. Q. Charles, H. Su, M. Kaichun, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 77–85.

[79] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 5105–5114.

[80] P. Guerrero, Y. Kleiman, M. Ovsjanikov, N. J. Mitra, Pcpnet: Learning local shape properties from raw point clouds, Computer Graphics Forum 37 (2) (2018) 75–85.

[81] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems 32 (1) (2021) 4–24. doi:10.1109/TNNLS.2020.2978386.

[82] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[83] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[84] V. Naumova, K. Schnass, Dictionary learning from incomplete data for efficient image restoration, in: Proceedings of the 25th European Signal Processing Conference (EUSIPCO), 2017, pp. 1425–1429.

[85] J. Batson, L. Royer, Noise2self: Blind denoising by self-supervision, in: Proceedings of the 36th International Conference on Machine Learning (ICML), 2019.

[86] A. Krull, T.-O. Buchholz, F. Jug, Noise2void-learning denoising from single noisy imagesg, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 2124–2132.

[87] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Trans. Graph. 38 (5).

[88] E. Dumic, C. R. Duarte, L. A. da Silva Cruz, Subjective evaluation and objective measures for point clouds - state of the art, in: 2018 First International Colloquium on Smart Grid Metrology (SmaGriMet), 2018.

[89] E. Alexiou, T. Ebrahimi, On subjective and objective quality evaluation of point cloud geometry, in: 2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX), 2017, pp. 1–3.

[90] a. C. B. Alireza Javaheri, F. Pereira, J. Ascens, Subjective and objective quality evaluation of compressed point clouds, in: 2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP), 2017, pp. 1–6.

[91] da Silva Cruz Luis A, E. Dumić, E. Alexiou, J. Prazeres, R. Duarte, M. Pereira, A. Pinheiro, T. Ebrahimi, Point cloud quality evaluation: Towards a definition

for test conditions, in: 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), 2019, pp. 1–6.

[92] S. Perry, H. P. Cong, L. A. da Silva Cruz, J. Prazeres, M. Pereira, A. Pinheiro, E. Dumic, E. Alexiou, T. Ebrahimi, Quality evaluation of point clouds encoded using mpeg codecs, in: 2020 IEEE International Conference on Image Processing (ICIP), 2020, pp. 3428–3432.

[93] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, C. T. Silva, A benchmark for surface reconstruction, ACM Transactions on Graphics 32 (2) (2013) 20:1–20:17.

[94] L. Giaccari, Surface reconstruction toolbox (2017).
URL `https://github.com/LuigiGiaccari/Surface-Reconstruction-Toolbox/releases`

[95] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, MeshLab: an Open-Source Mesh Processing Tool, in: Eurographics Italian Chapter Conference, 2008, pp. 129–136.

[96] A. Javaheri, C. Brites, F. Pereira, J. ao Ascenso, Subjective and objective quality evaluation of 3d point cloud denoising algorithms, in: 2017 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), 2017, pp. 1–6.

[97] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, ACM Transactions on Graphics 32 (3) (2013) 29:1–29:13.

[98] G. Lavoué, M. Corsini, A comparison of perceptually-based metrics for objective evaluation of geometry processing, IEEE Transactions on Multimedia 12 (2010) 636–649.

[99] A. Javaheri, C. Brites, F. Pereira, J. Ascenso, A generalized hausdorff distance based quality metric for point cloud geometry, in: 2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX), 2020, pp. 1–6.

[100] D. Tian, H. Ochimizu, C. Feng, R. Cohen, A. Vetro, Geometric distortion metrics for point cloud compression, in: 2017 IEEE International Conference on Image Processing (ICIP), 2017, pp. 3460–3464.

[101] G. Garg, M. Juneja, A survey of denoising techniques for multi-parametric prostate mri, Multimedia Tools and Applications 78 (2019) 12689–12722.

[102] E. Alexiou, T. Ebrahimi, Point cloud quality assessment metric based on angular similarity, in: 2018 IEEE International Conference on Multimedia and Expo (ICME), 2018, pp. 1–6.

[103] A. Javaheri, C. Brites, F. Pereira, J. Ascenso, Mahalanobis based point to distribution metric for point cloud geometry quality evaluation, IEEE Signal Processing Letters 27 (2020) 1350–1354.

[104] J. Zhang, J. Cao, X. Liu, H. Chen, B. Li, L. Liu, Multi-normal estimation via pair consistency voting, IEEE Transactions on Visualization and Computer Graphics 25 (4) (2019) 1693–1706.

[105] J. Xu, L. Zhang, D. Zhang, External prior guided internal prior learning for real-world noisy image denoising, IEEE Transactions on Image Processing 27 (6) (2018) 2996–3010.

[106] J. Xu, L. Zhang, D. Zhang, A trilateral weighted sparse coding scheme for real-world image denoising, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 21–38.

[107] J. Chen, J. Chen, H. Chao, M. Yang, Image blind denoising with generative adversarial network based noise modeling, in: 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 3155–3164.

[108] S. Anwar, N. Barnes, Real image denoising with feature attention, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 3155–3164.

[109] Y. Hou, J. Xu, M. Liu, G. Liu, L. Liu, F. Zhu, L. Shao, Nlh: A blind pixel-level non-local method for real-world image denoising, IEEE Transactions on Image Processing 29 (2020) 5121–5135.

[110] X. Zhu, P. Milanfar, Automatic parameter selection for denoising algorithms using a no-reference measure of image content, IEEE Transactions on Image Processing 19 (12) (2010) 3116–3132.

[111] H. Liang, D. S. Weller, Comparison-based image quality assessment for selecting image restoration parameters, IEEE Transactions on Image Processing 25 (11) (2016) 5118–5130.

[112] K. Gu, D. Tao, J. Qiao, W. Lin, Learning a no-reference quality assessment model of enhanced images with big data, IEEE Transactions on Neural Networks and Learning Systems 29 (4) (2018) 1301–1313.