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

Lang Zhou\textsuperscript{a}, Guoxing Sun\textsuperscript{b}, Yong Li\textsuperscript{b}, Weiqing Li\textsuperscript{c}, Zhiyong Su\textsuperscript{b,∗}

\textsuperscript{a}College of Information Engineering, Nanjing University of Finance and Economics, Nanjing, 210023, China
\textsuperscript{b}School of Automation, Nanjing University of Science and Technology, Nanjing, 210094, China
\textsuperscript{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

\textsuperscript{∗}Corresponding author.
Email address: su@njust.edu.cn (Zhiyong Su)
surfaces of visible objects. Theoretically, a point 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:

\[
p_i = q_i + e_i, \quad p_i, q_i, e_i \in \mathbb{R}^3,
\]

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

\[
q'_i = \mathcal{F}(p_i) = \mathcal{F}(q_i + e_i),
\]

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 \( \sigma \). 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.
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:

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

where $N(i)$ denotes the neighborhood of $i$, $w_c(x) = e^{-x^2/2\sigma_c^2}$ is a spatial smoothing function with standard deviation $\sigma_c$, and $w_s(x) = e^{-x^2/2\sigma_s^2}$ is an intensity smoothing function with standard deviation $\sigma_s$. 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

$$ p'_i = p_i + \delta_i \cdot \mathbf{n}_i, $$

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

$$ \delta_i = \frac{\sum_{p_j \in N_i(p_i)} w_d(||\mathbf{v}_{ij}||) w_n(||(\mathbf{n}_i, \mathbf{v}_{ij})||) (\mathbf{n}_i, \mathbf{v}_{ij})}{\sum_{p_j \in N_i(p_i)} w_d(||\mathbf{v}_{ij}||) w_n(||(\mathbf{n}_i, \mathbf{v}_{ij})||)}, $$

where $\mathbf{v}_{ij} = p_{ij} - p_i$, $N_i(p_i) = \{p_{ij} \in \mathcal{P} || p_{ij} - p_i || < r\}$ 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Weighting function</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digne et al. (2017)</td>
<td>Point position, point normal</td>
<td>Parallel implementation; trade-off between speed and quality</td>
</tr>
<tr>
<td>Zhang et al. (2019)</td>
<td>Point position, point normal, point color</td>
<td>Preserves sharp feature to some extent; removes small-scale noise</td>
</tr>
</tbody>
</table>

There are two 1D Gaussian functions with chosen variance $\sigma_d$ and $\sigma_n$ respectively, and $\langle \cdot, \cdot \rangle$ is the inner product of two vectors. In order to improve the computation efficiency, they implemented the proposed scheme through the OpenMP 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. 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 $\delta_i$ in Eq. (5):

$$\delta_i = \frac{\sum_{p_j \in N(p_i)} w_d(||v_{ij}||, |d_{ij}|) w_n(||v_{ij}||) \langle n_i, v_{ij} \rangle}{\sum_{p_j \in N(p_i)} w_d(||v_{ij}||, |d_{ij}|) w_n(||v_{ij}||)},$$  

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., 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 $\omega_i$ centered at the pixel $i$:

$$I^o(j) = a_i I^g(j) + b_i, \quad \forall j \in \omega_i$$  \hspace{1cm} (7)

where $a_i$ and $b_i$ are linear coefficients assumed to be constant in $\omega_i$.

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 $p_{ij} \in N(p_i)$ of $p_i$, the filtered position $p'_{ij}$ is defined as:

$$p'_{ij} = a_ip_{ij} + b_i,$$  \hspace{1cm} (8)

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

$$J(a_i, b_i) = \sum_{p_{ij} \in N(p_i)} \left( (a_ip_{ij} + b_i - p_{ij})^2 + \epsilon a_i^2 \right),$$  \hspace{1cm} (9)

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

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

$$b_i = \overline{p_i} - a_i \overline{p_i}$$  \hspace{1cm} (11)

where $\overline{p_i}$ is the centroid of $N(p_i)$. 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:

\[
\mathbf{n}_i^{k+1} = \frac{1}{K_i} \sum_{p_j \in N(p_i)} w_d(\|p_{ij} - p_i\|) w_n(\|\mathbf{n}_i^k - n_j\|) \cdot \mathbf{n}_j^k,
\]

where

\[
K_i^k = \sum_{p_j \in N(p_i)} w_d(\|p_{ij} - p_i\|) w_n(\|\mathbf{n}_i^k - n_j\|).
\]

The initial normal \(\mathbf{n}_i^0\) of each point \(p_i\) is estimated by using the Principal Components Analysis (PCA) [36]. In the \(k\)th 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 [35] 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>
<thead>
<tr>
<th>Method</th>
<th>Guidance</th>
<th>Features</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Han et al. (2018)</td>
<td>Point position</td>
<td>Self-guided denoising</td>
<td>Does not recover sharp features</td>
</tr>
<tr>
<td>Zheng et al. (2017)</td>
<td>Point position; point normal</td>
<td>Multi-normal guided normal filter</td>
<td>Needs fine parameters tuning; does not deal with different scales of features</td>
</tr>
<tr>
<td>Han et al. (2018)</td>
<td>Point position; point normal</td>
<td>An iterative guidance normal filter</td>
<td>Time-consuming; cannot preserve sharp features</td>
</tr>
<tr>
<td>Yadav et al. (2018)</td>
<td>Point position; point normal</td>
<td>Vertex-based normal voting tensor; binary eigenvalues optimization based normal filter</td>
<td>Preserves sharp edges and corners</td>
</tr>
<tr>
<td>Liu et al. (2020)</td>
<td>Point position; point normal</td>
<td>Anisotropic second order normal filter; bi-tensor voting based feature points detection</td>
<td>Preserves different levels of geometric features</td>
</tr>
<tr>
<td>Zheng et al. (2018)</td>
<td>Point position; point normal</td>
<td>Rolling guidance normal filter</td>
<td>Prevents large-scale sharp structures from severe distortion</td>
</tr>
<tr>
<td>Sun et al. (2019)</td>
<td>Point position; point normal</td>
<td>Denoising point clouds with rich textures; rolling guidance normal filter</td>
<td>Limited efficiency; filters several detailed features with important semantic information</td>
</tr>
</tbody>
</table>

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

A weighted graph $G = (V, E, W)$ is often defined by two sets: a node set $V$ of cardinality $|V| = n$ and an edge set $E$, 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. $W$ is a weighted adjacency matrix. To reflect the degree of pairwise similarity between node $v_i$ and node $v_j$, a weight $w_{i,j} \in W$ is often assigned to each edge $e_{i,j} \in E$. 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 prox-
imity. 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 $D$ be the diagonal degree matrix of $G$, where
di, j = $\sum_{j=1}^{n} w_{i,j}$. Given $W$ and $D$, the combinatorial graph Laplacian matrix $L$ [46] can
be written as $L = D - W$. The GLR [46] is defined as:

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

(14)

where $v \in \mathbb{R}^N$ is a graph signal defined on the graph $G$, $v_i$ and $v_j$ are a pair of connected
nodes. Consider an input noisy point cloud $P$ as a graph signal defined on the graph
$G$, where $p_i$ is a scalar value assigned to node $v_i$. It has been shown in [43, 44, 45]
that minimizing the GLR $\sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} (p_i - p_j)^2$ iteratively can promote piecewise
smoothness in the reconstructed graph signal $P$ [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 min-
imization of GLR using the Mahalanobis distance matrix as a variable. Interpreting
the above Mahalanobis distance matrix as a graph Laplacian, they developed a fea-
ture 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.
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) 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. divided the point cloud into two disjoint node sets (red and blue), and constructed a bipartite graph approximation of the original k-NN graph \( G \) 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 \( m \)-th order tensor \( \mathbf{U} \in \mathbb{R}^{I_1 \times \ldots \times I_m} \) with entries \( u_{i_1 \ldots i_m} \) and an \( n \)-th order tensor \( \mathbf{V} \in \mathbb{R}^{J_1 \times \ldots \times J_n} \)
with entries $v_{j_1...j_n}$ can be defined as $O = U \circ V$, where $O \in \mathbb{R}^{I_1 \times ... \times I_m \times J_1 \times ... \times J_n}$, and $O_{(i_1...i_m)\rightarrow(j_1...j_n)} = u_{i_1...i_m} \cdot v_{j_1...j_n}$.

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 $s = [s_1 \ s_2 \ \ldots \ s_n]^T$. 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 $A = (a_{i_1i_2...i_m}) \in \mathbb{R}^{n^m}$, whose entry in position $(i_1, i_2, \ldots, i_m)$ is labeled as $a_{i_1i_2...i_m}$. They referred the adjacency tensor as the representing tensor $A$, in which each entry $(a_{i_1i_2...i_m})$ indicates whether nodes $v_1, v_2, \ldots, v_m$ are connected in the hyperedges. The representing tensor $A$ can be decomposed via

$$A \approx \sum_{r=1}^{n} \lambda_r \cdot f_r \circ \ldots \circ f_r,$$  \hspace{1cm} (15)

where $f_r$s are orthonormal basis vectors called spectrum components, and $\lambda_r$ are frequency coefficients related to the hypergraph frequency. For clean point clouds, they estimated their spectrum components $f_r$s based on the hypergraph stationary process and optimally determined their frequency coefficients $\lambda_r$ based on smoothness to recover the original hypergraph structure. Then, given the original signal $s = [s_1 \ \ldots \ s_n]^T$, the hypergraph signal is defined as the $(m-1)$ times tensor outer product of $s$, i.e.,

$$s^{[m-1]} = s \circ \ldots \circ s.$$  \hspace{1cm} (16)

Given the hypergraph signal $s^{[m-1]}$ and the representing tensor $A$, they jointly estimated hypergraph spectrum pairs $(f_r, \lambda_r)$ 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Graph construction</th>
<th>Filter</th>
<th>Feature</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duan et al. (2018)</td>
<td>k-NN</td>
<td>Weighted multi-projection</td>
<td>Projects each point to its neighbors' tangent planes</td>
<td>Over-smoothing</td>
</tr>
<tr>
<td>Hu et al. (2020)</td>
<td>k-NN on patches</td>
<td>Graph laplacian regularizer using the Mahalanobis distance matrix</td>
<td>Feature graph learning for denoising</td>
<td>Requires small amounts of data for a stable estimation</td>
</tr>
<tr>
<td>Zeng et al. (2020)</td>
<td>k-NN on a manifold</td>
<td>Signal-dependent feature graph Laplacian regularizer</td>
<td>Uses the patch manifold prior</td>
<td>Preserves structural detail</td>
</tr>
<tr>
<td>Dinesh et al. (2020)</td>
<td>Bipartite graph</td>
<td>Signal-dependent feature graph Laplacian regularizer</td>
<td>Applied on 3D coordinates and surface normals</td>
<td>Removes Gaussian and Laplacian noise only</td>
</tr>
<tr>
<td>Irfan et al. (2021)</td>
<td>k-NN graph based on geometry and color</td>
<td>The regularity of the color, and its correlation with the proximity of points</td>
<td>Takes advantage of the correlation between the geometry and color attribute</td>
<td>Both for geometry and color denoising</td>
</tr>
<tr>
<td>Zhang et al. (2021)</td>
<td>Spectrum-based hypergraph</td>
<td>Tensor-based methods</td>
<td>Hypergraph signal processing</td>
<td>Both for sampling and denoising</td>
</tr>
</tbody>
</table>
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 method 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.
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 \[67\]:

\[
\min_{C} \frac{1}{2} \| X - DC \|_F^2 + \lambda \| C \|_p, \tag{17}
\]

where \( X \in \mathbb{R}^{n \times m} \) is the data matrix constructed from the input noisy point cloud \( P \), whose columns are signals, \( D \in \mathbb{R}^{n \times k} \) is a dictionary which is possibly over-complete, \( C \in \mathbb{R}^{k \times m} \) is the matrix of sparse coefficients, \( \lambda \) 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 $\ell_1$ \cite{68} and $\ell_0$ \cite{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 \cite{32}. The Moving Robust Principal Components Analysis (MRPCA) proposed by Mattei et al. \cite{62} used weighted $\ell_1$ 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 $\ell_1$ 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. \cite{63} proposed a new model to reconstruct and smooth point clouds, combining $\ell_1$-median filtering with sparse $\ell_1$ 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 $\ell_1$-median height filter and $\ell_1$ 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 $\ell_1$ \cite{68} and $\ell_0$ \cite{66} based regularization methods, Digne et al. \cite{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

<table>
<thead>
<tr>
<th>Method</th>
<th>Sparse reconstruction</th>
<th>Feature</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRPCA (2017) [62]</td>
<td>$\ell_1$ based regularization</td>
<td>Robust PCA solver for convex optimization</td>
<td>May produce an over-sharpened result in the region of curved sharp edges</td>
</tr>
<tr>
<td>Leal et al. (2020)</td>
<td>$\ell_1$ based regularization</td>
<td>$\ell_1$-median filtering with sparse $\ell_1$ regularization</td>
<td>Needs parameter tuning</td>
</tr>
<tr>
<td>LPF (2018) [64]</td>
<td>Dictionary learning</td>
<td>Jointly learns the set of LPFs</td>
<td>Sensitive to the dictionary size</td>
</tr>
</tbody>
</table>

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
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 $X$ be the matrix constructed from a patch group of the input noisy point cloud $P$. For each patch group, since its patches share the most similar geometry structures than the other patches, the constructed patch matrix $X$ 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 $A$ 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]:

$$
\min_A \text{rank}(A) + \lambda \|X - A\|_F^2
$$

(18)

where $\text{rank}(\cdot)$ is the rank function of a matrix, $\lambda$ 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 $A$.

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>
<thead>
<tr>
<th>Method</th>
<th>Features</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lu et al. (2020)</td>
<td>Local isotropic structure patch in the normal field; low-rank matrix approximation</td>
<td>Suitable for offline geometry processing</td>
</tr>
<tr>
<td>Chen et al. (2020)</td>
<td>Rotation-invariant height-map patch in the surface height field; low-rank matrix approximation</td>
<td>Time-consuming; over-sharpen some non-sharp regions</td>
</tr>
<tr>
<td>Zhou et al. (2021)</td>
<td>Projective height vector in the normal height field; low-rank matrix approximation</td>
<td>Structure preservation; sensitive to the density of point clouds</td>
</tr>
<tr>
<td>Hu et al. (2020)</td>
<td>( k )-NN on patches; Graph laplacian regularizer using the Mahalanobis distance matrix</td>
<td>Requires small amounts of data for a stable estimation</td>
</tr>
<tr>
<td>Zeng et al. (2020)</td>
<td>Low-dimensional manifold model; discrete patch-based graph laplacian regularizer</td>
<td>High complexity</td>
</tr>
<tr>
<td>LPF (2018)</td>
<td>Consolidate local probing fields (LPF) as a local frame, dictionary learning</td>
<td>Sensitive to the dictionary size</td>
</tr>
</tbody>
</table>
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]. 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...
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) \cite{8, 12}, or represent it as graphs followed by graph convolution operations \cite{10, 11}, as illustrated in Figure 7 \cite{81}.

Riccardo et al. \cite{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>
<thead>
<tr>
<th>Method</th>
<th>Network architecture</th>
<th>Loss Functions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCPNet (2018)</td>
<td>PointNet</td>
<td>Normals: Euclidean distance, angle difference;</td>
<td>Fail in the presence of large flat areas</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Curvatures: rectified error for training and the RMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>for evaluation</td>
<td></td>
</tr>
<tr>
<td>PointCleanNet (2020)</td>
<td>PCPNet; outlier detection network; denoising network</td>
<td>Proximity to the surface; regular distribution on the surface</td>
<td>Sensitive to outliers; point cloud shrinking</td>
</tr>
<tr>
<td>EC-Net (2018)</td>
<td>PointNet++; Edge distance regression; coordinate regression</td>
<td>Surface loss; edge loss; repulsion loss; edge distance regression loss</td>
<td>Manually annotates sharp edges; fixed patch size; poor in tiny structures</td>
</tr>
<tr>
<td>PointProNet (2018)</td>
<td>Heightmap generation network; heightmap denoising network</td>
<td>Distance between denoised and ground truth heightmap</td>
<td>Fails in large holes and high level noise; artifacts in extreme sharp edges</td>
</tr>
<tr>
<td>Lu et al. (2020)</td>
<td>LeNet for classification; ResNet18 for normal estimation</td>
<td>Weighted $\ell_2$ distance</td>
<td>Fails in severe noise and significant outliers</td>
</tr>
<tr>
<td>GPDNet (2020)</td>
<td>Graph-convolutional layer</td>
<td>MSE; MSE-SP</td>
<td>Robust to high level of noise and structured noise distributions</td>
</tr>
<tr>
<td>PF (2021)</td>
<td>Encoder-decoder network</td>
<td>Projection distance</td>
<td>Requires ground-truth point normals in the training stage</td>
</tr>
</tbody>
</table>
PointNet \cite{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 \cite{16}. In the image denoising field, Noise2Noise \cite{84}, and its extensions Noise2Self \cite{85} and Noise2Void \cite{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 \cite{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>
<thead>
<tr>
<th>Method</th>
<th>Network architecture</th>
<th>Loss Functions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TotalDenoising (2019)</td>
<td>Unstructured encoder-decoder based on Monte Carlo convolution</td>
<td>$\ell_2$ loss</td>
<td>Cannot preserve shape features; sensitive to outliers</td>
</tr>
<tr>
<td>Chen et al. (2020) [17]</td>
<td>Encoder: PointNet; Decoder: folding module, graph-topology-inference module, graph-filtering module</td>
<td>Augmented Chamfer distance</td>
<td>Effective in reconstruction, visualization, and transfer classification</td>
</tr>
<tr>
<td>Luo et al. (2020) [18]</td>
<td>Encoder: DGCNN; Decoder: MLP</td>
<td>Supervised training loss: Chamfer distance, Earth Mover’s distance; unsupervised training loss: $\ell_2$ loss</td>
<td>Trained in either a supervised or unsupervised fashion</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>Point-to-point</td>
</tr>
<tr>
<td>RMSE</td>
<td>Point-to-point</td>
</tr>
<tr>
<td>PSNR</td>
<td>Point-to-point</td>
</tr>
<tr>
<td>CD</td>
<td>Point-to-point</td>
</tr>
<tr>
<td>Javaheri</td>
<td>Point-to-plane</td>
</tr>
<tr>
<td>Alexiou</td>
<td>Plane-to-plane</td>
</tr>
<tr>
<td>Javaheria</td>
<td>Point-to-distribution</td>
</tr>
</tbody>
</table>

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 [90], 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 \( \mathcal{A} \):

\[
e(p, \mathcal{A}) = \min_{p_i \in \mathcal{A}} d(p, p_i^\mathcal{A}),
\]

where \( d(,.) \) is the Euclidean 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:

\[
H_{asy}(\mathcal{A}, \mathcal{B}) = \max_{p_i^\mathcal{A} \in \mathcal{A}} e(p_i^\mathcal{A}, \mathcal{B}).
\]

The symmetric Hausdorff distance is then defined as follows:

\[
H_{sym}(\mathcal{A}, \mathcal{B}) = \max\{H_{asy}(\mathcal{A}, \mathcal{B}), H_{asy}(\mathcal{B}, \mathcal{A})\}.
\]

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, Java-heri et al. extended the classical Hausdorff distance and proposed a quality evaluation strategy called the generalized Hausdorff distance [99]

\[
H_{gen}(\mathcal{A}, \mathcal{B}) = \max_{p_i^\mathcal{A} \in \mathcal{A}} \per_{K}^{th} d(p_i^\mathcal{A}, \mathcal{B}),
\]

where \( \per_{K}^{th} d(p_i^\mathcal{A}, \mathcal{B}) \) 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 $\mathcal{A}$ \cite{96}:

$$\text{RMSE}(\mathcal{A}, \mathcal{B}) = \sqrt{\text{MSE}(\mathcal{A}, \mathcal{B})} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \| p^A_i - p^B_i \|^2}, \quad (23)$$

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

**Mean City-block Distance (MCD).** MCD is similar to MSE with $\ell_2$ norm replaced with $\ell_1$ norm \cite{44}:

$$\text{MCD}(\mathcal{A}, \mathcal{B}) = \frac{1}{n} \sum_{i=1}^{n} | p^A_i - p^B_i |. \quad (24)$$

**Peak signal to noise ratio (PSNR).** PSNR is defined as $10\log_{10}$ of the ratio of a peak value $M$ to the square root of MSE between the noisy and the reference point cloud \cite{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:

$$\text{PSNR}(\mathcal{A}, \mathcal{B}) = 10 \log_{10} \left( \frac{M^2}{\text{MSE}(\mathcal{A}, \mathcal{B})} \right). \quad (25)$$

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 \cite{9,13,15}. CD is defined as:

$$\text{CD}(\mathcal{A}, \mathcal{B}) = \frac{1}{|\mathcal{A}|} \sum_{p^A_i \in \mathcal{A}} \min_{p^B_j \in \mathcal{B}} (\| p^A_i - p^B_j \|^2) + \frac{1}{|\mathcal{B}|} \sum_{p^B_j \in \mathcal{B}} \min_{p^A_i \in \mathcal{A}} (\| p^B_j - p^A_i \|^2). \quad (26)$$

### 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 \( p_i^A \) in point cloud \( A \), find the nearest neighbor point \( p_j^B \) in point
  cloud \( B \) as its corresponding point.

- Compute the unit normal vector \( n_i^A \) on point \( p_i^A \) in the reference point cloud \( A \),
  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 \( p_i^A \) to \( p_j^B \).

- Project the error vector \( E(i, j) \) along the normal direction \( n_i^A \) to get the point-to-
  plane error:

\[
e(A, B) = \frac{1}{|A|} \sum_{p_i^A \in A} (E(i, j) \cdot n_i^A)^2. \tag{27}
\]

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 in-
clude RIMLS [53], CLOP [60], GPF [61], LPF [64], PointCleanNet [9], Pointfilter
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).

<table>
<thead>
<tr>
<th>Models(points)</th>
<th>RIMLS</th>
<th>CLOP</th>
<th>GPF</th>
<th>LPF</th>
<th>PointCleanNet</th>
<th>Pointfilter</th>
<th>AD</th>
</tr>
</thead>
<tbody>
<tr>
<td>bunny(35947)</td>
<td>48.1857</td>
<td>48.4146</td>
<td>42.8261</td>
<td>48.7172</td>
<td>48.3412</td>
<td>49.2297</td>
<td>49.0393</td>
</tr>
<tr>
<td>Cube(24578)</td>
<td>45.5341</td>
<td>44.5173</td>
<td>42.2118</td>
<td>45.268</td>
<td>44.2063</td>
<td>45.5069</td>
<td>45.8145</td>
</tr>
<tr>
<td>Eight(29580)</td>
<td>45.5466</td>
<td>44.6984</td>
<td>36.5813</td>
<td>45.8541</td>
<td>44.6491</td>
<td>45.8858</td>
<td>45.2505</td>
</tr>
<tr>
<td>Joint(69544)</td>
<td>46.2033</td>
<td>45.0441</td>
<td>41.6862</td>
<td>46.4824</td>
<td>45.1500</td>
<td>46.7316</td>
<td>46.5271</td>
</tr>
<tr>
<td>Kitten(24956)</td>
<td>45.4730</td>
<td>44.4748</td>
<td>40.0056</td>
<td>45.8713</td>
<td>44.4175</td>
<td>46.0124</td>
<td>45.6003</td>
</tr>
<tr>
<td>Plane-sphere(18531)</td>
<td>48.8509</td>
<td>48.1370</td>
<td>42.7978</td>
<td>49.4534</td>
<td>47.9223</td>
<td>49.4595</td>
<td>49.4422</td>
</tr>
<tr>
<td>block(32370)</td>
<td>42.9914</td>
<td>42.3013</td>
<td>42.1779</td>
<td>43.1771</td>
<td>42.2128</td>
<td>43.6166</td>
<td>43.3533</td>
</tr>
<tr>
<td>child(50002)</td>
<td>47.6989</td>
<td>48.2588</td>
<td>43.3147</td>
<td>48.3816</td>
<td>48.4493</td>
<td>48.6345</td>
<td>49.1511</td>
</tr>
<tr>
<td>chinese_Joint(50003)</td>
<td>41.9390</td>
<td>43.2764</td>
<td>38.6749</td>
<td>42.7804</td>
<td>43.6654</td>
<td>43.3093</td>
<td>44.1240</td>
</tr>
<tr>
<td>eros100K(50002)</td>
<td>45.2728</td>
<td>45.3624</td>
<td>39.9963</td>
<td>45.5562</td>
<td>45.8594</td>
<td>44.8889</td>
<td>46.1356</td>
</tr>
<tr>
<td>fertility(13971)</td>
<td>46.3205</td>
<td>46.3539</td>
<td>44.1436</td>
<td>46.7658</td>
<td>46.0688</td>
<td>47.4635</td>
<td>47.3425</td>
</tr>
<tr>
<td>genus3r(29663)</td>
<td>47.7402</td>
<td>48.3228</td>
<td>44.3067</td>
<td>48.4552</td>
<td>48.3011</td>
<td>49.6790</td>
<td>49.2529</td>
</tr>
<tr>
<td>grayloc(34274)</td>
<td>45.0994</td>
<td>46.3928</td>
<td>41.5385</td>
<td>45.8805</td>
<td>46.7177</td>
<td>47.4295</td>
<td>47.2238</td>
</tr>
<tr>
<td>horse(48485)</td>
<td>40.1681</td>
<td>45.2767</td>
<td>35.4816</td>
<td>42.1123</td>
<td>45.2834</td>
<td>45.9057</td>
<td>46.1371</td>
</tr>
<tr>
<td>part_Lp(25994)</td>
<td>47.6773</td>
<td>46.6724</td>
<td>43.5330</td>
<td>47.7082</td>
<td>46.6810</td>
<td>48.1388</td>
<td>47.9187</td>
</tr>
<tr>
<td>pulley(50000)</td>
<td>43.4536</td>
<td>45.0524</td>
<td>42.3665</td>
<td>44.7348</td>
<td>45.3355</td>
<td>45.1186</td>
<td>45.3711</td>
</tr>
<tr>
<td>bumpy_torus(16815)</td>
<td>46.3100</td>
<td>44.4732</td>
<td>39.7064</td>
<td>45.8164</td>
<td>44.7101</td>
<td>44.9595</td>
<td>45.6648</td>
</tr>
<tr>
<td>pyramid(12290)</td>
<td>45.1285</td>
<td>44.1825</td>
<td>44.6298</td>
<td>45.1055</td>
<td>44.3652</td>
<td>45.4668</td>
<td>45.7149</td>
</tr>
<tr>
<td>rolling_stage(49988)</td>
<td>47.9001</td>
<td>48.2946</td>
<td>41.5853</td>
<td>48.5863</td>
<td>48.5864</td>
<td>49.0758</td>
<td>49.6429</td>
</tr>
<tr>
<td>screwdriver(21752)</td>
<td>36.1560</td>
<td>41.1380</td>
<td>36.6015</td>
<td>38.5769</td>
<td>42.5170</td>
<td>43.6648</td>
<td>42.6747</td>
</tr>
<tr>
<td>sharp_sphere(28051)</td>
<td>45.127</td>
<td>44.3338</td>
<td>42.1241</td>
<td>45.3046</td>
<td>45.0015</td>
<td>45.6441</td>
<td>45.6852</td>
</tr>
<tr>
<td>smooth-feature(24871)</td>
<td>44.6808</td>
<td>43.4985</td>
<td>44.7482</td>
<td>44.6553</td>
<td>43.3647</td>
<td>44.7917</td>
<td>44.9090</td>
</tr>
<tr>
<td>sphere(16386)</td>
<td>46.2519</td>
<td>44.6893</td>
<td>42.1789</td>
<td>45.6329</td>
<td>44.8021</td>
<td>46.2177</td>
<td>46.2686</td>
</tr>
<tr>
<td>star(18146)</td>
<td>48.2372</td>
<td>46.8414</td>
<td>41.4964</td>
<td>48.1927</td>
<td>46.8238</td>
<td>48.3193</td>
<td>48.0599</td>
</tr>
<tr>
<td>trim-star(24467)</td>
<td>48.1751</td>
<td>47.5566</td>
<td>46.2349</td>
<td>48.5228</td>
<td>47.6599</td>
<td>49.0233</td>
<td>48.7306</td>
</tr>
</tbody>
</table>
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


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


