# Learning Robust Graph-Convolutional Representations for Point Cloud Denoising

# Learning Robust Graph-Convolutional Representations for Point Cloud Denoising

Francesca Pistilli, *Student Member, IEEE*, Giulia Fracastoro, *Member, IEEE*, Diego Valsesia, *Member, IEEE*, Enrico Magli, *Fellow, IEEE*

***Abstract*—Point clouds are an increasingly relevant geometric data type but they are often corrupted by noise and affected by the presence of outliers. We propose a deep learning method that can simultaneously denoise a point cloud and remove outliers in a single model. The core of the proposed method is a graph-convolutional neural network able to efficiently deal with the irregular domain and the permutation invariance problem typical of point clouds. The network is fully-convolutional and can build complex hierarchies of features by dynamically constructing neighborhood graphs from similarity among the high-dimensional feature representations of the points. The proposed approach outperforms state-of-the-art denoising methods showing robust performance in the challenging setup of high noise levels and in presence of structured noise.**

***Index Terms*—Point cloud, denoising, outlier removal, graph neural networks.**

## I. Introduction

Point clouds are an unordered collection of 3D points sampled from 2D surfaces of objects or scenes. They are usually acquired by devices, such as LIDARs, or are the output of reconstruction algorithms. Point clouds are becoming increasingly popular due to their ability to provide a detailed representation of the real world and the interest in exploring geometric representation in challenging applications, such as autonomous driving. However, the use of such type of data is limited by the fact that the acquisition methods typically perturb the data with a non-negligible amount of noise and outliers. Therefore, the development of efficient restoration methods is crucial to improve the performance of various downstream tasks such as shape matching, surface reconstruction, object segmentation and more.

Several approaches to point clouds denoising and outlier removal can be found in literature. Traditional model-based methods typically address the denoising task by fitting a surface to the noisy data or exploiting some geometric features extracted from the noisy observations and detect outliers using statistical methods. These techniques are efficient at low levels of noise, otherwise they usually suffer from oversmoothing. Recently, there has been great interest around learning-based methods, in particular deep neural networks. The main challenges of point cloud processing with neural networks are the irregular domain, i.e., point clouds can not be represented in a grid-like structure, and the permutation-invariance problem, i.e., the fact that any permutation of the order in which points are stored still represents the same point cloud. These two characteristics make extending traditional techniques based on convolutional neural networks (CNNs) to point clouds difficult. Some algorithms tackled these challenges by either approximating the irregular domain with a grid, e.g. by building voxels, or by building a permutation-invariant model as a composition of operations acting on single points (e.g., size-1 convolution) and a globally symmetric function (e.g., a max pool) as done by PointNet [1]. Nevertheless, the former introduces an undesirable approximation, while the latter lacks the expressiveness of CNN where the convolution operation extracts features that are localized as functions of the neighborhood of a pixel and features are assembled in a hierarchical manner by means of multiple layers, progressively expanding the receptive field.

Graph convolution [2] has recently emerged as a valid and elegant method to build operators over irregular domains represented as graphs and it is able to deal with permutation-invariant data. Furthermore, this type of network is able to replicate some of the most desirable characteristic of convolutional neural networks (CNN), such as localization and compositionality of the features as well as efficient weight reuse. Spatial-domain definitions of graph convolution have been recently applied to several problems involving point clouds such as classification [3], segmentation [4], shape completion [5] and generation [6]. However, point clouds denoising and outlier removal have not yet been addressed and present unique challenges such as studying graph convolutional layers that are stable in presence of noise and constructing local feature hierarchies rather than global features as typically required by classification or segmentation problems.

In this paper, we propose a deep graph-convolutional neural network to denoise the point cloud geometry. The proposed network simultaneously removes outliers and denoises the remaining points, learning effective feature representation for both tasks without prior knowledge of the noise variance. The proposed architecture has an elegant fully-convolutional behavior that, by design, can build hierarchies of local or non-local features to effectively regularize the denoising problem. This is in contrast with other methods in the literature that typically work on fixed-size patches or apply global operations [7], [8]. Moreover, dynamic computation of the graph from similarities among the high-dimensional feature-space representations of the points allows to uncover more complex latent correlations than defining neighborhoods in the noisy 3D space, and possibly exploit non-local self-similarity. Extensive experimental results show a significant

The authors are with Politecnico di Torino - Department of Electronics and Telecommunications, Italy. Email:name.surname@polito.it. This material is based upon work supported by Google Cloud.

improvement over state-of-the-art methods, especially in the challenging conditions of high noise levels. The proposed approach is also robust to structured noise distributions such as the ones encountered in real LiDAR acquisitions. Our main contributions with respect to existing literature can therefore be summarized as: i) an improved architecture for the point cloud restoration problem that can handle outlier detection and removal and denoising in a single model; ii) a highly general graph convolutional layer that is stable in presence of noise and with a dynamic update of the graph; iii) state-of-the-art denoising performance. This work is an extension of our conference work [9]. It significantly expands the conference material by considering the problem of simultaneous denoising and outlier detection and removal instead of simple denoising of white Gaussian noise. We include a novel neural network architecture that efficiently creates a shared feature space both for the outlier detection and denoising tasks, an augmented loss function based on weighted cross entropy to enable the detection of sparse outliers, and a blind denoising method that does not require variance-specific training and expanded experimental results.

The paper is organized as follows. Sec. II-A presents some background material on graph convolutional neural networks and discusses how the proposed approach differs from other works in the literature. Sec. III presents our proposed method. Sec. IV and Sec. V present extensive experimental validation and analysis of the properties of the proposed method. Finally, Sec. VI draws some conclusions.

## II. Background and related work

### A. Graph-convolutional neural networks

In the last years, data-driven solutions based on neural networks have shown impressive performance on a variety of problems, including low-level tasks such as image restoration [10], [11]. The workhorse of these methods is the convolutional neural network (CNN), which has been shown to capture highly complex features in images. Despite their success, a major shortcoming of CNNs is that they are unable to process data defined on irregular domains. In particular, one case that is drawing attention is when the data structure can be described by a graph and data are defined as vectors on the graph nodes. We can find such kind of data in many applications, e.g. 3D point clouds [4], [3], computational biology [12], [13], and social networks [14]. However, extending CNNs from data with a regular structure, such as images and video, to graph-structured data is not straightforward.

In order to design effective graph neural networks, one of the major challenges is defining a convolution operator for graph-structured data that has all the desirable properties of the classical convolution. In fact, convolution is one of the main building blocks of the standard CNNs. Its properties of stationarity, locality and compositionality are a good match for many types of data and can allow effective weight reuse. For these reasons, defining a convolutional operation for graph-structured data with similar characteristics is of paramount importance and it has been extensively studied in recent years, even if a universally accepted solution is still missing at the moment. In the literature on this topic, we can identify two different classes of approaches. The first one is the spectral approaches [15], [16], [14], where the convolution operator is defined in the frequency domain through the graph Fourier transform. Polynomial approximations [16] have been proposed in order to reduce the computational cost of this operation. One of the most well-known graph neural network architectures, the Graph Convolutional Network (GCN) [14], uses a degree-1 polynomial approximation for semi-supervised problems. However, a fundamental limitation of the spectral approaches is that the learned filters cannot be generalized to different graph structures. The second class of approaches overcomes this drawback by defining the convolution operator in the spatial domain. In this case, the convolution is defined as a weighted local aggregation combining the signal values of the neighboring nodes. Since such operation is defined at a neighborhood level, it can be effectively applied to data with different graph structures. Several definitions of local aggregations have been proposed [17], [18], [19], [3], [20], [21], [4], [22], [23]. Many of them employ scalar weights [17], [20], [18] or weight matrices that do not depend on the input data [4], [19], [21], [23]. On the other hand, the Edge-Conditioned Convolution (ECC) in [3] proposes to weigh the contributions of the neighbors using edge-dependant matrices. With respect to the other approaches, the use of edge-dependent matrices allows to define a more general convolution operator with an increased representational power. This motivates our choice of employing the ECC in this paper. On top of this, we also use the approximations proposed in [24] in order to address vanishing gradient and over-parameterization issues that typically affect this definition of graph convolution.

### B. Point cloud denoising

3D point cloud denoising has received great attention and the approaches to solve it can be broadly categorized into: local surface fitting methods [25], [26], [27], [28], [29], [30], sparsity-based methods [31], [32], [33], graph-based methods [34], [35], [36], and learning-based methods [7], [37], [8], [38], [39]. Among the methods belonging to the first category, the moving least squares (MLS) approach [25] is one of the most popular and its extensions [26], [27] provide a degree of robustness in presence of outliers. Other surface fitting methods have also been proposed for point cloud denoising, such as jet fitting [30] or parameterization-free local projector operator (LOP) [28], [29]. These methods achieve remarkable performance at low levels of noise, but they suffer from over-smoothing when the noise level is high [40] or if outliers are not robustly removed.

A second class of point cloud denoising methods [31], [32], [33] is based on sparse representations. In this case, the denoising procedure solves two minimization problems with sparsity constraints, where the first one estimates the surface normals and then the second one uses them in order to update the point positions. However, at high levels of noise the normal estimation can be very poor, leading to over-smoothing or over-sharpening [32]. The MRPCA model [33] accounts for the presence of sparse outliers but it can be hard to correctly

tune the desired sparsity in the optimization problem and its convergence is not very robust in presence of high noise and strong outliers.

Another approach for point cloud denoising is derived from the theory of graph signal processing [41]. These methods [34], [35], [36] first define a graph whose nodes are the points of the point cloud. Then, graph total variation (GTV)-based regularization methods are applied for denoising. These techniques have proved to achieve very strong performance when the noise level is low. Instead, at high noise levels, the graph construction can become unstable, negatively affecting the denoising performance.

In the last years, learning-based methods [7], [37], [8], [38], especially the ones based on deep learning, have been gaining attention. Extending convolutional neural networks to point cloud data is not straightforward, due to the irregular positioning of the points in the space. In the context of shape classification and segmentation, many methods have recently been proposed specifically to handle point cloud data. PointNet [1] is one of the most relevant works in this field, where each point is processed independently before applying a global aggregation. Recently, a few methods proposed to extend the approach of PointNet to point cloud denoising. PointCleanNet [7] uses an approach similar to PointNet in order to estimate correction vectors for the points in the noisy point cloud. In contrast, the authors in [8] use a neural network similar to PointNet to estimate a reference plane for each noisy point and then they obtain the denoised point cloud by projecting the noisy point onto the corresponding reference plane. Also PointProNet [38] performs point cloud denoising by employing an architecture similar to PointNet in order to estimate the local directions of the surface. However, the main drawback of these techniques based on PointNet is that they work on individual points and then apply a global symmetric aggregation function, but they do not exploit the local structure of the neighborhood. PointCleanNet addresses this issue by taking as input local patches instead of the entire point cloud. However, this solution is still limited by the fact that the network cannot learn hierarchical feature representations, like standard CNNs. Moreover, PointCleanNet does not handle denoising with outliers in a single model, but has two models, one specialized on outlier detection and one on denoising, thus limiting its efficiency.

Graph-convolutional networks have shown promising performance on tasks such as segmentation and classification. In particular, DGCNN [4] first introduced the idea of a dynamic graph update in the hidden layers of a graph-convolutional network in the context of classification and segmentation tasks addressed in [4]. However, the denoising problem is significantly different since it relies more on localized representations rather than global features. In particular, there are several design choices that make DGCNN unsuitable for point cloud denoising: the spatial transformer block is not useful for denoising since it seeks a canonical global representation, whereas denoising is mostly concerned with local representations of point neighborhoods; it also significantly increases the computational complexity for large point clouds; the DGCNN graph convolution operation uses a max operator in the aggregation, which is unstable in presence of noise; the specific graph convolution definition is also less general than the one presented in this paper, which allows to implement adaptive filters where the aggregation weights are dependent on the feature vectors instead of being fixed as in [4], as well as incorporating an edge attention term which is especially important in presence of noise because it promotes a lowpass behavior by penalizing edges with large feature variations.

## III. Proposed method

This section presents the proposed Graph-convolutional Point Outlier removal and Denoising (GPOD) network. The goal of our work is to build a single model that can robustly denoise point clouds affected by outliers as well as geometry noise. We first present a brief system overview and then focus on the main building blocks.

### A. Architecture overview

An overview of the system architecture is presented in Fig. 1. The proposed method combines the outlier removal and denoising tasks in a single model so that a common feature space can be efficiently learned. At a high level, the architecture shows three main building blocks: i) feature extraction; ii) outlier detection and removal; iii) residual denoising.

The feature extraction stage acts as a pre-processor that creates a feature space capturing localized representations of neighborhoods in the point cloud. This feature space is shared between the outlier removal and denoising tasks and its representations are robust to input noise. The feature extraction stage is composed of three convolutional layers with single-point convolutions and a residual block with three graph-convolutional layers.

Outlier detection is performed by using the representations learned by the feature extractor and feeding them to a point-wise binary classifier, implemented as a single-point convolution. Outlier removal is then performed by subsampling the point cloud, removing all those points classified as outliers according to a design decision threshold on the classifier probabilities.

The subsampled point cloud is then processed by the rest of the network, which completes the denoising task by means of two residual blocks with graph-convolutional layers.

We remark that, contrary to other methods in the literature (e.g., PointCleanNet [7]), we propose a single model that performs outlier removal and denoising at the same time. This is advantageous as it increases efficiency by sharing a common parameterization for both tasks. Moreover, early outlier removal is advantageous with respect to returning both outlier probabilities and denoised points as final outputs of the model since the presence of outliers can negatively affect the feature space learned by the network and reduce the performance. In fact, outlier points should be disregarded as they do not convey any information about the signal, and, if not removed, they could be involved in graph-convolution operations and perturb the displacement estimation, especially at high levels of noise.

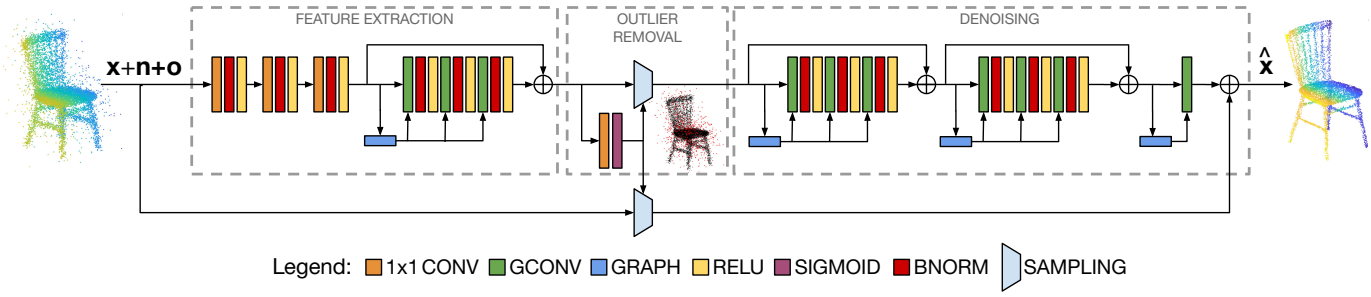


Fig. 1. Proposed Graph-convolutional Point Outlier removal and Denoising (GPOD) architecture.

### B. Graph-convolutional layer

The graph-convolutional layer is the core of the proposed architecture. As described in Section II-A, we use the Lightweight ECC [24], which is a modified version of the ECC presented in [3].

The graph-convolutional layer has two inputs: a matrix $\mathbf{H}^{l+1} \in \mathbb{R}^{F^l \times N}$ representing a feature vector for each of the $N$ points, and a graph describing the connections between points. The output feature vectors $\mathbf{H}^{l+1} \in \mathbb{R}^{F^{l+1} \times N}$ at layer $l$ are computed by performing a weighted local aggregation:

$$\mathbf{H}_i^{l+1} = \mathbf{W}^l \mathbf{H}_i^l + \sum_{j \in \mathcal{N}_i^l} \gamma^{l,j \to i} \frac{\sum_{t=1}^{r} \omega_t^{j \to i} \boldsymbol{\phi}_t^{j \to i} \boldsymbol{\psi}_t^{j \to i^T} \mathbf{H}_j^l}{|\mathcal{N}_i^l|},$$

where $\mathbf{H}_i^l$ is the input feature vector at node $i$ and $\mathcal{N}_i^l$ is the set of its neighbors. The weights include a self-loop matrix $\mathbf{W}^l \in \mathbb{R}^{F^{l+1} \times F^l}$ which is shared among all points. The other weights employed in the aggregation, i.e., vectors $\boldsymbol{\phi}_t^{j \to i} \in \mathbb{R}^{F^{l+1}}$, $\boldsymbol{\psi}_t^{j \to i} \in \mathbb{R}^{F^l}$ and $\omega_t^{j \to i} \in \mathbb{R}$, are computed as functions of the difference between the input feature vectors of point $i$ and point $j$, i.e., $\boldsymbol{\phi}_t^{j \to i}, \boldsymbol{\psi}_t^{j \to i}, \omega_t^{j \to i} = \mathcal{F}\left(\mathbf{h}_i^l - \mathbf{h}_j^l\right)$. This function is implemented as a multilayer perceptron with two layers, where the final fully-connected layer can be approximated by means of a stack of circulant matrices since the number of free parameters would otherwise be very large. The aggregation weight matrix is approximated as $\sum_{t=1}^{r} \omega_t^{j \to i} \boldsymbol{\phi}_t^{j \to i} \boldsymbol{\psi}_t^{j \to i^T}$, where $r$ is a hyperparameter setting the maximum rank; this is done to reduce the number of parameters and memory requirements of the aggregation operation. We refer the reader to [24] for additional details on these approximations. The parameter $\gamma^{l,j \to i} \in \mathbb{R}$ is an edge attention term which exponentially depends on the Euclidean distance between feature vectors of neighboring nodes:

$$\gamma^{l,j \to i} = \exp\left(-\|\mathbf{H}_i^l - \mathbf{H}_j^l\|_2^2 / \delta\right),$$

being $\delta$ a decay hyperparameter. This is particularly useful in stabilizing the operation in presence of noise or imperfect outlier removal.

The graph is dynamically constructed by searching for the $k$-nearest neighbors of each point in terms of Euclidean distance between their feature vectors. In order to reduce the computational cost, a search area of predefined size, centered around the point, is defined, e.g., as a fixed number of neighbors in the noisy 3D space, among which $k$ nearest neighbors are selected for each point (see Fig. 7 for a visual representation of the search area and feature space neighborhoods). This allows to avoid computing pairwise distances between all points in the point cloud at every layer where the graph is recomputed, and only compute distances between the features of a point and those of the points in its search area. The graph is computed at the input of each residual block and then it is shared among the graph-convolutional layers inside the block, again to limit complexity.

As described in Sec. II-A, the definition of graph convolution employed in this paper presents some advantages over alternative definitions such as GraphSAGE [19], FeastNet [20] or DGCNN [4]. In particular, the aggregation weights depend on the input data making the filtering layer adaptive. Moreover, since the function is implemented as an multilayer perceptron, it can be more general than a fixed function with some learnable parameters.

Finally, we remark that the proposed architecture is fully-convolutional thanks to the graph convolution operation. By fully-convolutional we mean that the output feature vector of each point at a given layer is obtained as an aggregation of the feature vectors of neighboring points in the previous layer, thus building complex hierarchies of aggregations. This is in contrast with PointCleanNet [7] which works by processing each patch independently to estimate the denoised version of the central point. That approach is not able to create hierarchies of features obtained by successive multi-point aggregations, as in a classical CNN. Thanks to the graph-convolutional structure, we can replicate this behavior in the graph-based approach, thereby learning more powerful feature spaces.

### C. Outlier detection

The part of the network able to detect outliers is composed of two building blocks: the feature extraction and the outlier removal block, as reported in Fig. 1.

The feature extraction is performed by three single-point convolutions that gradually project the input from the 3D space to an $F$-dimensional feature space, and a residual block. In general, a residual block is a building-block of the proposed network characterized by three graph-convolutional layers followed by batch normalization to stabilize training and an input-output skip connection to reduce vanishing gradient issues. Notice that the graph is shared among the graph convolutional layers inside the residual block to limit computational complexity. This block is able to learn a feature

representation suitable for the addressed tasks. The features are then processed by a binary classifier, which is implemented as a single-point convolution that returns the probability of being an outlier associated to each point. All the points that are considered as outlier according to a specific detection threshold are removed from the feature representation of the original point cloud and the feature vectors of the remaining points are the input to the denoising block.

The loss function considered for the outlier detection task is the weighted cross entropy (WCE):

$$L_{\mathrm{WCE}} = -\sum_i \left[\beta \cdot p_i \log(\hat{p_i}) + (1-p_i)\log(1-\hat{p_i})\right], \quad (1)$$

where $\beta$ is a positive weight, $p_i$ is the true label of the $i$-th point and $\hat{p}_i$ is the predicted probability of being an outlier. The weighted cross entropy is usually exploited for unbalanced classification, as in our scenario where the number of outliers is typically far less compared to the total number of points in the point cloud. The parameter $\beta$ is a weight able to penalize or increment the cost of positive error with respect to negative errors. If $\beta > 1$, the number of false negatives decreases increasing the recall, otherwise the number of false positive decreases increasing the precision. In our network the parameter $\beta$ is set to a value larger than one in order to increase the chance of capturing the outliers, avoiding false negatives that would highly penalize the overall denoising performance.

### D. Denoising with outlier removal

The denoising block reported in Fig. 1 takes as input the feature representations of all the points except those classified as outliers, and returns as output the final denoised point cloud in 3D space without outliers.

At a high level, the block architecture is a residual network that estimates the additive noise component of the input instead of the denoised point cloud, because it has been shown [10] that predicting the residual is easier than directly cleaning the data. A cascade of two residual blocks is responsible for the noise estimation in the feature space; the noise is later projected to the 3D space by a single graph-convolutional layer. Finally the estimated noise is removed from the noisy point cloud. At the beginning of each residual block the graph is computed by selecting the $k$ nearest neighbors to each point in terms of Euclidean distances in the feature space. The dynamic graph construction, consisting in updating the graph after each residual block, has been shown to induce more powerful feature representations [4], [6] and in the context of a residual denoising network it progressively uncovers the latent correlations that have not been eliminated yet.

The denoising loss function is a combination of the mean squared error (MSE) and a regularization term that takes into account the surface proximity (SP), computed as the distance between each denoised point and the closest ground truth:

$$L_{\mathrm{MSE-SP}} = \frac{1}{N}\sum_{i=1}^{N}\left[\|\hat{\mathbf{x}}_i^{'} - \mathbf{x}_i\|_2^2 + \lambda \min_j \|\hat{\mathbf{x}}_i^{'} - \mathbf{x}_j\|_2^2\right], \quad (2)$$

where $N$ is the number of points in the original point cloud, $\hat{\mathbf{x}}^{'}$ is the denoised point cloud without the outliers, $\mathbf{x}$ is the original point cloud without additive noise and outliers, and $\lambda$ is a regularization hyperparameter. The possible remaining outliers, which are not detected by the outlier classifier, are not considered in the loss function since its purpose is to constrain the denoising performance in presence of additive Gaussian noise. Notice that the MSE cost alone does not account on the fact that the points may lie on a surface and therefore the tangential component of the noise is not as relevant as the normal component. This property is incorporated by means of the SP regularizer, which approximates the distance of the denoised point from the ground truth surface by computing the distance between the denoised point and the closest ground truth point. Other works also considered proximity to surface in the loss function. Notably, PointCleanNet [7] uses a loss that combines the proximity to surface with a dual term measuring the distance between a ground truth point and the closest denoised point. This is done to ensure that the denoised points do not collapse into filament structures. We found that using the MSE to enforce this property provides better results. Indeed, combining this dual term and surface proximity one obtains the popular Chamfer measure (C2C), but to the degerate behavior promoted by the dual term, using C2C as loss function has, in general, worse performance than the proposed loss.

### E. Training procedure

The final loss for denoising with outlier removal combines the losses for the two tasks as:

$$L_{\mathrm{OR-DN}} = L_{\mathrm{MSE-SP}} + \alpha L_{\mathrm{WCE}}, \quad (3)$$

where $\alpha$ is a regularization parameter.

The training procedure follows three steps: i) optimize only the parameters of the feature extraction and the outlier removal block using Eq. (1) as loss; ii) then optimize only the parameters of the denoising block using Eq. (2) as loss; iii) finally, finetune all the parameters of the proposed network using Eq. (3) as loss.

This training process allows each module to correctly specialize for its own tasks, limiting the overall computational complexity, while at the same time providing the benefits of end-to-end optimization in the final finetuning step.

## IV. Experimental results

### A. Experimental setting

The training, validation and test sets are collections of post-processed point clouds from the Shapenet [43] repository. This database is composed of 3D models of 55 object categories, each described as a collection of meshes. We first sample 30720 uniformly distributed points for each selected model, then we scale the obtained point clouds normalizing their diameter in order to ensure that data are at the same scale. The training set is a collection of more than 100000 patches of 1024 points, randomly selected from point clouds from all the categories of the repository except those reserved for the test set and the outlier validation set. Each patch is created by selecting a point from a point cloud and collecting its 1023

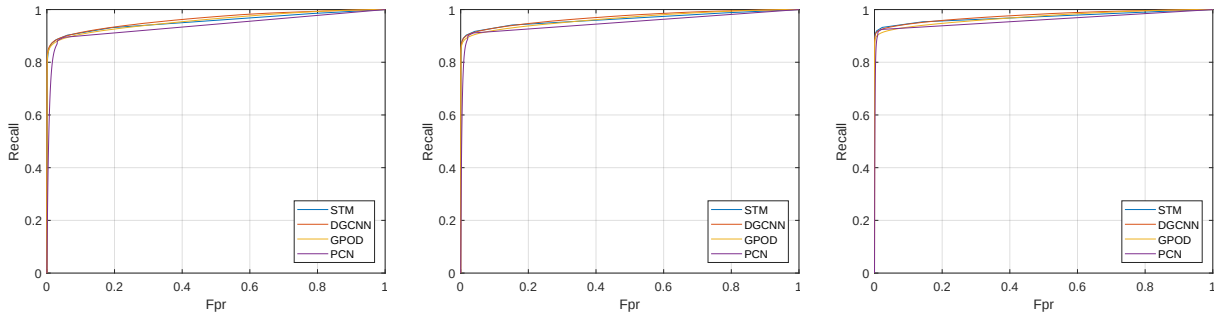


Fig. 2. ROC curves. Left $\sigma = 0.02$, center $\sigma = 0.015$, right $\sigma = 0.01$.

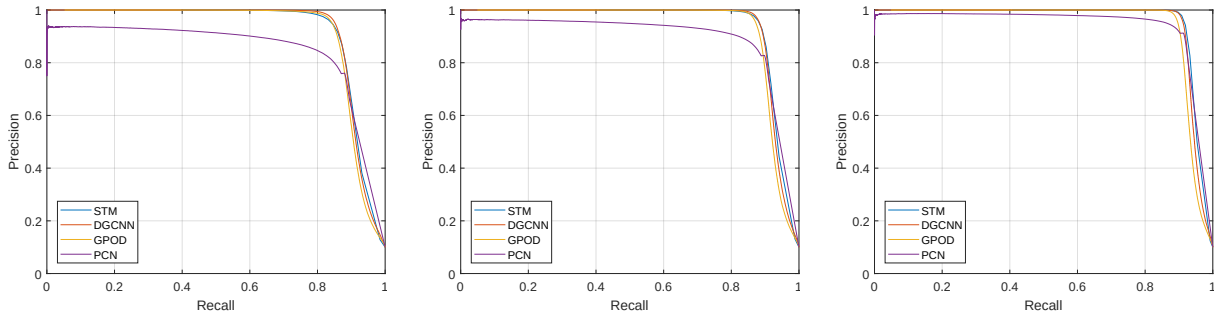


Fig. 3. Precision-Recall curves. Left $\sigma = 0.02$, center $\sigma = 0.015$, right $\sigma = 0.01$.

TABLE I
OUTLIER DETECTION PERFORMANCE (%).

| $\sigma$ | F1 STM [42] | F1 DGCNN [4] | F1 PCN [7] | F1 **GPOD** | Recall STM [42] | Recall DGCNN [4] | Recall PCN [7] | Recall **GPOD** | Precision STM [42] | Precision DGCNN [4] | Precision PCN [7] | Precision **GPOD** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.02 | 89.58 ± 3.07 | 89.69 ± 3.57 | 81.01 ± 1.36 | 89.43 ± 3.29 | 84.61 ± 4.76 | 85.36 ± 4.33 | 88.27 ± 4.75 | 83.88 ± 4.48 | 94.55 ± 1.45 | 94.54 ± 2.97 | 75.06 ± 2.17 | 95.87 ± 2.31 |
| 0.015 | 92.21 ± 2.51 | 92.12 ± 2.80 | 85.67 ± 1.65 | 91.41 ± 2.68 | 86.67 ± 4.27 | 87.09 ± 4.08 | 90.05 ± 4.20 | 85.29 ± 4.23 | 97.73 ± 0.82 | 97.83 ± 1.60 | 81.85 ± 2.15 | 98.59 ± 0.99 |
| 0.01 | 94.10 ± 2.07 | 93.81 ± 2.26 | 90.99 ± 2.08 | 92.50 ± 2.34 | 88.54 ± 3.98 | 88.76 ± 3.85 | 91.53 ± 3.77 | 86.30 ± 4.02 | 99.66 ± 0.18 | 99.56 ± 0.36 | 90.57 ± 2.12 | 99.76 ± 0.22 |

closest points. The test set consists of 100 point clouds taken from ten different categories: airplane, bench, car, chair, lamp, pillow, rifle, sofa, speaker, and table. Two test sets are created from the clean collection of point clouds, one corrupted only by Gaussian noise and the other one with additional outliers. The validation set is a collection of 10 point clouds, belonging to five different categories: bath, clock, laptop, tower and train. The validation set is employed after the first stage of the training to set the threshold for outlier detection. The threshold is identified as the value that allows to achieve the maximum F1 score over the validation set and the selected value is exploited in the following two stages of the training. In our experiments the threshold is set to 0.03.

The obtained training, validation and test sets are artificially corrupted to model noisy observations. Additive Gaussian noise with a range of standard deviations $\sigma_n \in [0.01, 0.02]$ is added to the original data to simulate different levels of noise. The proposed method is blind, therefore data corrupted by all the standard deviations in the aforementioned range are considered for the training procedure. For the denoising with outlier removal task, a mix of outliers and noisy points are considered. Outliers are modeled as points corrupted by additive Gaussian noise with a standard deviation 10 times higher than the one of the higher level of Gaussian noise, i.e., $\sigma_o = 0.2$, which is added to 10% of the noiseless data points. Finally, an outlier is resampled if its realization is closer than one noise standard deviation to the original position.

The different stages of the recommended training procedure are trained for approximately 50000, 80000 and 100000 iterations, respectively. A batch size of 16 patches is always used. The number of features used for all the layers is 99, except for the first three single-point convolutional layers of the feature extraction block, where the number of features is gradually increased from 33 to 66 and finally to 99. The Adam optimizer has been employed with a fixed learning rate equal to $10^{-5}$ for the first two stages and equal to $10^{-6}$ for the last one. Concerning the graph-convolutional implementation, the rank $r$ for the low-rank approximation is set to 11 and 3 circulant rows are considered for the construction of the circulant matrix. During testing, the network takes as input the whole point cloud and a search area is associated to each point of the point cloud, wherein the neighbors are searched and identified. Unless otherwise stated, 16 nearest neighbors are used for graph construction.

### B. Outlier detection performance

In this section the outlier detection performance of the GPOD network is analyzed and compared with other techniques. We consider a statistical method (STM) based on [42] and two learning-based methods, PointCleanNet [7] and

DGCNN [4]. PointCleanNet (PCN) is one of the most recent methods that specifically address both denoising and outlier removal tasks and its code is publicly available. The method is divided into two different networks, separately trained and utilized for the two tasks; in this section we only consider the performance of the network in charge of the outlier detection. The DGCNN architecture, which originally addresses segmentation and classification of point clouds, is modified in order to adapt the segmentation architecture for the outlier detection task. The purpose of this experiment is to verify whether the small subnetwork dedicated to outlier removal is competitive against state-of-the-art models specialized for outlier removal. Indeed, we do not seek to attain state-of-the-art performance but to be close to it with a small module that can be easily integrated in a larger architecture and still retain that performance after also training the full architecture for the denoising task, For this reason, we present PointCleanNet and DGCNN trained to only perform outlier detection, contrary to GPOD which is trained for both outlier detection and denoising.

To evaluate the performance the Receiver Operating Characteristic (ROC) and the precision/recall curves obtained over the test set are analyzed and reported in Fig. 2 and 3. Moreover, average values of F1-score, recall and precision of the ten categories of the test set are reported in Table I. We ensure a fair comparison on the reported values by selecting the detection threshold as the one that maximizes the F1 score over the validation set for all the methods. In particular for PointCleanNet the threshold is equal to 0.8, for DGCNN to $4.4 \times 10^{-6}$ and for STM to 0.61.

The results suggest that all methods roughly exhibit the same performance, which is an interesting result since GPOD is not specifically specialized on outlier removal. It also has to be noted that PCN tends, by design, to have a high recall at the cost of a lower precision.

TABLE II
CHAMFER MEASURE ($\times 10^{-6}$), $\sigma = 0.02$. DENOISING PERFORMANCE.

| Class | Noisy | DGCNN [4] | TD [37] | APSS [27] | RIMLS [26] | AWLOP [29] | MRPCA [33] | GLR [34] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|---|---|---|---|---|
| Airplane | 161.79 | 131.11 | 73.08 | 175.68 | 186.24 | 145.94 | 123.71 | 90.55 | 74.17 | **53.65** |
| Bench | 161.52 | 122.55 | 69.05 | 166.85 | 182.42 | 157.29 | 127.51 | 83.99 | 90.34 | **48.26** |
| Car | 148.74 | 137.25 | 92.36 | 141.69 | 167.78 | 145.51 | 109.49 | 77.56 | 160.08 | **74.50** |
| Chair | 163.75 | 159.69 | 86.34 | 160.01 | 155.38 | 158.12 | 122.70 | 79.85 | 145.56 | **66.52** |
| Lamp | 204.05 | 273.24 | 93.84 | 178.08 | 198.22 | 187.31 | 146.41 | 109.24 | 85.31 | **63.72** |
| Pillow | 215.58 | 198.95 | 83.43 | 164.83 | 196.53 | 206.14 | 150.65 | 85.86 | 92.84 | **64.84** |
| Rifle | 144.18 | 86.67 | 52.44 | 195.68 | 176.07 | 144.22 | 105.87 | 89.19 | 71.57 | **35.60** |
| Sofa | 184.11 | 155.88 | 84.64 | 166.34 | 190.91 | 178.93 | 133.98 | 89.31 | 144.72 | **78.79** |
| Speaker | 186.01 | 172.84 | 92.29 | 138.80 | 162.34 | 180.45 | 126.17 | 84.37 | 160.26 | **74.00** |
| Table | 168.32 | 144.88 | 71.96 | 171.25 | 179.81 | 162.36 | 125.72 | 78.06 | 102.17 | **53.47** |
| Average | 173.80 | 158.31 | 79.95 | 165.92 | 179.57 | 166.63 | 127.22 | 86.80 | 112.70 | **61.33** |

### C. Denoising performance without outliers

In this section, we analyze the pure denoising performance of the methods, i.e., when the test point clouds are only corrupted by additive Gaussian noise without outliers.

Several classes of point cloud denoising methods are available in literature, as reported in Section II-B and we consider at least one algorithm from each one. APSS [27] and RIMLS [26] are well-known MLS-based surface fitting methods and they were tested using the MeshLab software [44]. AWLOP [29] is another surface fitting method and it was tested using the software released by the authors. MRPCA [33] is a sparsity-based method and it was tested using the code provided by the authors. GLR [34] is one of the most promising works belonging to the graph signal processing category and it was tested using the code provided by the authors. PointCleanNet (PCN) [7] is considered as the state-of-the-art learning-based method. We also consider Total Denoising (TD) [37] as a more recent learning-based method. Furthermore, a modified version of the popular DGCNN [4] is taken into account as an additional baseline. This modified version replaces the output $1 \times 1$ convolutional layer to regress the point displacement and disregards the category embedding as all methods are blind to class labels.

TABLE III
CHAMFER MEASURE ($\times 10^{-6}$), $\sigma = 0.015$. DENOISING PERFORMANCE.

| Class | Noisy | DGCNN [4] | TD [37] | APSS [27] | RIMLS [26] | AWLOP [29] | MRPCA [33] | GLR [34] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|---|---|---|---|---|
| Airplane | 97.78 | 77.18 | 41.29 | 86.42 | 106.33 | 73.32 | 67.39 | 36.76 | 35.27 | **29.14** |
| Bench | 94.82 | 70.17 | 44.93 | 75.51 | 91.93 | 82.04 | 70.05 | 32.19 | 30.10 | **27.62** |
| Car | 102.23 | 94.82 | 66.71 | 72.56 | 103.52 | 93.38 | 69.88 | 55.92 | 92.23 | **54.53** |
| Chair | 105.16 | 100.93 | 67.63 | 81.47 | 104.38 | 92.47 | 73.45 | 48.62 | 69.18 | **47.07** |
| Lamp | 120.65 | 173.83 | 43.30 | 65.79 | 82.40 | 88.78 | 77.09 | 39.93 | 30.59 | **28.57** |
| Pillow | 132.57 | 120.43 | 35.11 | 22.74 | 42.54 | 112.54 | 73.67 | 31.38 | 29.02 | **27.27** |
| Rifle | 80.40 | 45.26 | 56.46 | 92.14 | 110.51 | 69.35 | 55.65 | 31.81 | **21.45** | 23.30 |
| Sofa | 121.02 | 101.30 | 46.09 | 42.80 | 69.92 | 107.58 | 72.62 | 51.12 | 61.15 | **43.91** |
| Speaker | 123.27 | 114.86 | 67.25 | 46.45 | 58.28 | 110.29 | 77.95 | 53.75 | 87.68 | **51.12** |
| Table | 103.50 | 87.84 | 54.56 | 62.64 | 78.21 | 89.33 | 70.87 | 37.94 | 43.88 | **35.30** |
| Average | 108.14 | 98.66 | 52.33 | 64.85 | 84.80 | 91.91 | 70.86 | 41.94 | 50.05 | **36.78** |

TABLE IV
CHAMFER MEASURE ($\times 10^{-6}$), $\sigma = 0.01$. DENOISING PERFORMANCE.

| Class | Noisy | DGCNN [4] | TD [37] | APSS [27] | RIMLS [26] | AWLOP [29] | MRPCA [33] | GLR [34] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|---|---|---|---|---|
| Airplane | 50.32 | 39.80 | 42.13 | 28.22 | 39.73 | 31.27 | 28.19 | **19.56** | 26.36 | 31.64 |
| Bench | 48.71 | 36.88 | 40.92 | 26.97 | 32.76 | 34.08 | 32.93 | **20.43** | 27.64 | 33.03 |
| Car | 64.34 | 60.77 | 65.78 | 47.73 | 55.56 | 54.21 | 44.33 | **42.22** | 75.34 | 54.06 |
| Chair | 60.78 | 58.00 | 77.03 | 37.31 | 45.65 | 47.91 | 38.41 | **34.98** | 55.10 | 50.75 |
| Lamp | 59.73 | 112.13 | 44.76 | 24.57 | 34.02 | 35.23 | 31.51 | **19.67** | 20.58 | 31.79 |
| Pillow | 69.79 | 62.97 | 24.56 | **15.64** | 21.23 | 46.36 | 23.95 | 17.59 | 21.07 | 29.21 |
| Rifle | 38.97 | 22.20 | 90.50 | 36.01 | 49.37 | 27.79 | 23.49 | **15.84** | 15.09 | 36.38 |
| Sofa | 69.63 | 57.87 | 41.22 | **22.27** | 28.04 | 53.08 | 32.14 | 30.88 | 43.36 | 40.99 |
| Speaker | 73.50 | 69.39 | 68.59 | **26.50** | 30.19 | 58.92 | 47.57 | 40.78 | 76.09 | 56.03 |
| Table | 56.21 | 47.44 | 54.98 | 27.45 | 32.63 | 41.26 | 34.78 | **27.12** | 43.02 | 40.38 |
| Average | 59.20 | 56.74 | 55.05 | 29.27 | 36.92 | 43.011 | 33.73 | **26.91** | 40.36 | 40.43 |

In order to compare the denoising performance, the Chamfer measure, also called Cloud-to-Cloud (C2C) distance is computed. This metric is widely utilized in point cloud denoising, because it computes an average distance of the denoised points from the original surface. First, the mean distance between each denoised point and its closest ground truth point is computed, then the one between each ground truth point and its closest denoised point. The Chamfer measure is taken as their average:

$$\text{C2C} = \frac{1}{2}\left[\frac{1}{N_1}\sum_{j=1}^{N_1}\min_i \|\hat{\mathbf{x}}_i - \mathbf{x}_j\|_2^2 + \frac{1}{N_2}\sum_{i=1}^{N_2}\min_j \|\hat{\mathbf{x}}_i - \mathbf{x}_j\|_2^2\right], \tag{4}$$

where $N_1$ and $N_2$ are respectively the number of points in the original and in the denoised point cloud, $\hat{\mathbf{x}}$ the denoised points and $\mathbf{x}$ the original points. We compute the average C2C

TABLE V
MEAN CHAMFER MEASURE ($\times 10^{-6}$), $\sigma = 0.01$. VARIANCE-SPECIFIC TRAINED NETWORK. DENOISING PERFORMANCE.

| Class | Noisy | DGCNN [4] | TD [37] | APSS [27] | RIMLS [26] | AWLOP [29] | MRPCA [33] | GLR [34] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|---|---|---|---|---|
| Average | 59.20 | 53.14 | 55.05 | 29.27 | 36.92 | 43.011 | 33.73 | 26.91 | 40.36 | **24.68** |

over all the models of each category in the test set and the results of the experiments at different noise levels are reported in Table II, III and IV. It can be observed that the GPOD network significantly outperforms state-of-the-art methods at medium and high levels of noise, as shown in Table II with $\sigma = 0.02$ and Table III with $\sigma = 0.015$. Instead, traditional model-based methods become very competitive for the easier task at low noise variance ($\sigma = 0.01$). This behaviour can be explained by the fact that most of the other methods involves surface reconstruction or normal estimation, operations that are computed with reduced accuracy at high levels of noise. Instead, GPOD and other learning-based methods can leverage more complex features which are more robust at high noise levels. It has to be recalled that the proposed method is trained blind, without using any information about the amount of noise that is inserted in the point clouds. Therefore, it is reasonable to accept high performance at higher level of noise, which is the most critical scenario, and average results at lower level of noise. Furthermore, if more information about the noise variance is available, it is possible to significantly increase performance, especially at low level of noise, by training a non-blind model, as shown in Table V. In this case our network and the other learning-based methods, DGCNN and PCN, are trained with point clouds corrupted by additive Gaussian noise at a specific standard deviation ($\sigma = 0.01$) and it is clearly visible that the proposed method outperforms the others.

Fig. 4 shows qualitative results at a medium noise level by presenting the denoised point cloud for each method. The surface distance of each point is visualized in the figure to understand the position of the denoised points with respect to the ground truth. The root mean square value of the surface distance (RMSD) can be computed as:

$$\mathrm{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\min_{j}\|\hat{\mathbf{x}}_i - \mathbf{x}_j\|_2^2}. \tag{5}$$

We can observe that on average GPOD provides a lower point-surface distance and the shape of the reconstructed point cloud is more similar to the original one compared to the other methods.

### D. Denoising performance with outlier removal

Finally, the denoising performance in presence of Gaussian noise and outliers is analyzed. The methods taken into account for comparison are denoising methods that include outlier removal. For the learning-based methods, we consider PointCleanNet [7] and DGCNN [4]. For both PointCleanNet and DGCNN the experiments are performed using two models: first given the noisy observations the outlier detection network is applied to identify and remove the outliers, then the remaining points are fed into the denoising network, obtaining the final denoised results. The same thresholds exploited for the computation of the outlier removal performance in Sec. IV-B are applied. For what concerns traditional model-based approaches, we use STM as first step for the outlier removal task and then the resulting point clouds are processed by GLR (STM + GLR).

TABLE VI
CHAMFER MEASURE ($\times 10^{-6}$), $\sigma = 0.02$. DENOISING WITH OUTLIER REMOVAL PERFORMANCE.

| Class | Noisy | STM + GLR [42] [34] | DGCNN [4] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|
| Airplane | 2888.50 | 95.99 | 132.06 | **55.52** | 59.80 |
| Bench | 2947.12 | 89.6 | 122.16 | 55.53 | **50.73** |
| Car | 2508.31 | 83.35 | 137.46 | 133.41 | **81.53** |
| Chair | 2381.04 | 92.90 | 162.18 | 108.63 | **75.07** |
| Lamp | 2958.91 | 120.40 | 194.73 | **59.56** | 66.29 |
| Pillow | 2816.90 | 91.48 | 198.14 | **56.05** | 74.42 |
| Rifle | 3958.87 | 102.18 | 81.72 | 49.73 | **38.92** |
| Sofa | 2527.52 | 97.00 | 162.54 | 122.23 | **82.63** |
| Speaker | 2328.51 | 87.21 | 155.96 | 134.34 | **77.36** |
| Table | 2720.44 | 85.77 | 144.75 | 81.58 | **52.10** |
| Average | 2803.61 | 94.59 | 149.17 | 85.66 | **65.88** |

We again adopt the Chamfer measure to evaluate the denoising performance in presence of outliers. In this case, the number of points in the denoised and original point clouds, $N_2$ and $N_1$ in Eq. (4), will differ depending on the number of the detected outliers. The median C2C values for the categories in the test set for different levels of standard deviation are reported in Tables VI, VII and VIII.

It can be seen that at medium and high levels of noise our method is able to outperform the state-of-the-art, as shown in Table VI and VII. Once again, at low level of noise the model-based method (STM+GLR) becomes more effective and achieves the best performance. Nevertheless, GPOD is the best among the learning-based methods despite being a single model. The insights on this behavior are similar to the pure denoising experiments reported in section IV-C. Also in this case, if more information about the noise is provided it is possible to perform a variance-specific training which significantly increase the performance. In particular, at low level of noise our method achieves the best results outperforming STM+GLR, as reported in Table IX. Tables X and XI report the results using the Cloud to Plane metric [45] and show a similar behavior to the aforementioned one.

Finally, qualitative results are reported in Fig. 5 for the denoising with outlier removal task. In the figure the denoised point clouds without outliers are shown and the surface distance of each point is reported. To measure the point-surface distance, we use the RMSD metric previously introduced for Fig. 4. GPOD provides on average a point-surface distance lower with respect to the other methods. Furthermore, our method is able to better reconstruct the original shape and surfaces of the point cloud without loosing important details or thinning the shape.

### E. Real noise removal

In order to check if the proposed architecture can generalize beyond white Gaussian noise, we test our model on two realistic denoising settings.

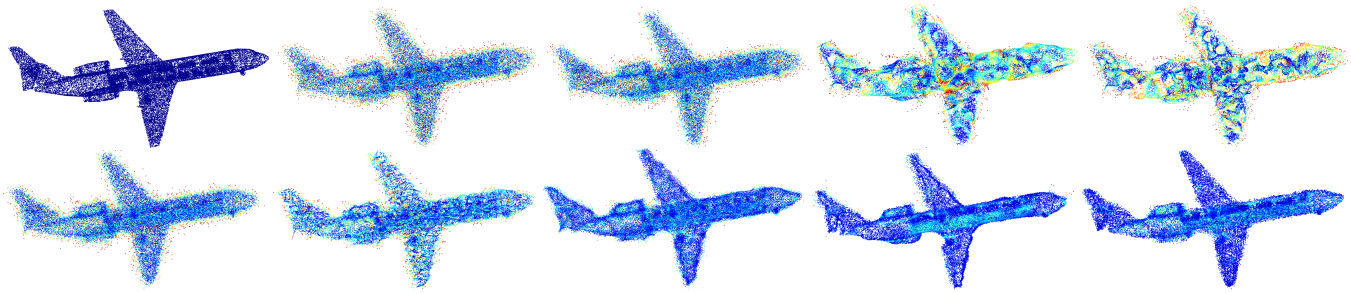

Fig. 4. Denoising results for $\sigma = 0.015$. Color represents distance to surface (red is high, blue is low). Top left to bottom right: clean point cloud, noisy point cloud, DGCNN (RMSD $= 0.0091$), APSS (0.0123), RIMLS (0.0127), AWLOP (0.0106), MRPCA (0.0096), GLR (0.0070), PointCleanNet (0.0065), GPOD (0.0062).

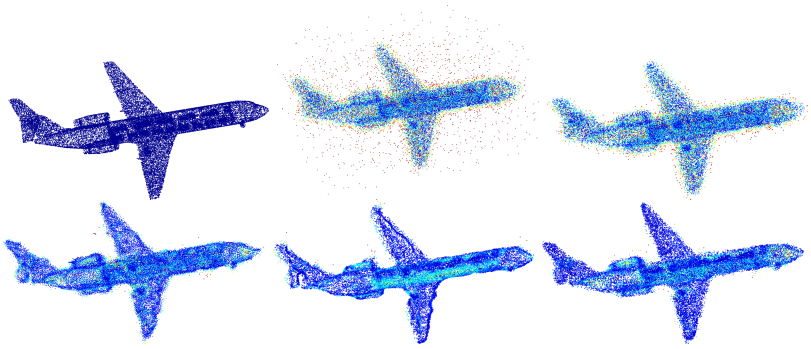

Fig. 5. Denoising results for $\sigma = 0.015$ and outliers. Color represents distance to surface (red is high, blue is low). Top left to bottom right: clean point cloud, noisy point cloud, DGCNN (RMSD $= 0.011$), STM + GLR (0.0077), PointCleanNet (0.0063), GPOD (0.0064).

TABLE VII
CHAMFER MEASURE ($\times 10^{-6}$), $\sigma$ = 0.015. DENOISING WITH OUTLIER REMOVAL PERFORMANCE.

| Class | Noisy | STM + GLR [42] [34] | DGCNN [4] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|
| Airplane | 2849.51 | 40.05 | 81.18 | 39.54 | **29.60** |
| Bench | 2865.19 | 37.90 | 70.71 | 33.73 | **28.01** |
| Car | 2476.96 | 64.21 | 95.72 | 107.50 | **59.25** |
| Chair | 2367.17 | 56.78 | 110.92 | 69.48 | **48.68** |
| Lamp | 2818.64 | 45.77 | 129.65 | 35.78 | **32.36** |
| Pillow | 2799.90 | 33.37 | 124.37 | 51.33 | **30.04** |
| Rifle | 3968.17 | 40.52 | 44.02 | 27.99 | **24.40** |
| Sofa | 2408.57 | 60.29 | 108.48 | 74.87 | **49.91** |
| Speaker | 2323.95 | 50.08 | 120.01 | 85.31 | **43.16** |
| Table | 2580.49 | 42.66 | 88.69 | 55.13 | **35.81** |
| Average | 2745.86 | 47.16 | 97.37 | 58.07 | **38.12** |

TABLE VIII
CHAMFER MEASURE ($\times 10^{-6}$), $\sigma$ = 0.01. DENOISING WITH OUTLIER REMOVAL PERFORMANCE.

| Class | Noisy | STM + GLR [42] [34] | DGCNN [4] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|
| Airplane | 2765.50 | **26.38** | 41.50 | 28.51 | 33.28 |
| Bench | 2790.02 | **26.36** | 36.32 | 27.53 | 34.80 |
| Car | 2425.01 | **50.87** | 63.01 | 74.52 | 54.56 |
| Chair | 2358.60 | **43.66** | 63.19 | 57.44 | 52.01 |
| Lamp | 2830.85 | **24.98** | 67.12 | 36.52 | 33.21 |
| Pillow | 2692.34 | **19.76** | 67.69 | 24.59 | 28.85 |
| Rifle | 3845.30 | 23.91 | 19.92 | **17.01** | 39.33 |
| Sofa | 2431.10 | **38.04** | 60.89 | 48.030 | 42.04 |
| Speaker | 2330.73 | **37.17** | 75.36 | 51.30 | 41.54 |
| Table | 2610.38 | **36.42** | 47.88 | 41.56 | 39.78 |
| Average | 2707.98 | **32.75** | 54.29 | 40.70 | 39.94 |

TABLE IX
MEAN CHAMFER MEASURE ($\times 10^{-6}$), $\sigma$ = 0.01. VARIANCE-SPECIFIC TRAINING. DENOISING WITH OUTLIER REMOVAL PERFORMANCE.

| Class | Noisy | STM + GLR [42] [34] | DGCNN [4] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|
| Average | 2707.98 | 32.75 | 62.24 | 48.46 | **29.77** |

TABLE X
MEAN POINT TO PLANE DISTANCE ($\times 10^{-6}$). DENOISING WITH OUTLIER REMOVAL PERFORMANCE.

| $\sigma$ | Noisy | STM + GLR [42] [34] | DGCNN [4] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|
| 0.02 | 2110.32 | 61.86 | 110.2 | **41.58** | 41.93 |
| 0.015 | 2058.11 | 24.41 | 68.39 | 25.09 | **17.32** |
| 0.01 | 2050.00 | **14.39** | 35.61 | 19.80 | 19.98 |

First, we train and evaluate our network on a simulated LiDAR dataset. We simulate scanning the Shapenet objects with a Velodyne HDL-64E scanner using the Blensor software [46]. Two sources of noise are considered for the acquisition process: a laser distance bias with Gaussian distribution and a per-ray Gaussian noise. We set both distributions to be zero-mean and with a standard deviation equal to 1% of the longest side of the object bounding box. We also retrained PointCleanNet on the simulated data for comparison with a

TABLE XI
MEAN POINT TO PLANE DISTANCE ($\times 10^{-6}$), $\sigma$ = 0.01. VARIANCE-SPECIFIC TRAINING. DENOISING WITH OUTLIER REMOVAL PERFORMANCE.

| | Noisy | STM + GLR [42] [34] | DGCNN [4] | PCN [7] | **GPOD** |
|---|---|---|---|---|---|
| Average | 2050.00 | 14.39 | 39.27 | 22.65 | **11.30** |

TABLE XII
VELODYNE SCAN STRUCTURED NOISE, RMSD, 8-NN.

| Noisy | PCN | **GPOD** |
|---|---|---|
| 0.1447 | 0.0966 | **0.0602** |

state-of-the-art model. Table XII shows that the results follow those on white Gaussian noise, with the proposed method improving over PointCleanNet. Note that RMSD is used as metric in place of the Chamfer measure since it is better suited to the case when points are not uniformly distributed.

The second experiment involves real point clouds generated by image-based 3D reconstruction techniques. Point clouds obtained by these methods are in general highly affected by noise and a large amount of outliers due to image imperfections. We considered the multiple-view plane-sweep algorithm (PS) [47] as image-based reconstruction method and one of the generated point clouds provided by the algorithm implementation in [48]. The point cloud shown in Fig. 6 on the left is the noisy reconstruction produced by the algorithm and is the input to the denoiser. We report qualitative results of our method and PointCleanNet in Fig. 6 to have a benchmark since the ground truth is not available. We exploited pre-trained models provided by the authors to test PointCleanNet and finetuned our network with data affected by a variable proportion of outliers. GPOD provides a denoised point cloud with fewer diffused outliers with respect to PointCleanNet, as visible from the bottom-right portion of the point clouds in Fig. 6 where PointCleanNet presents a cluster of points outside of the main shape of the torch that cannot be easily removed without compromising the entire shape. In general our method is able to reconstruct sharper object details, as in the body of the torches.

## V. METHOD ANALYSIS

### A. Low-rank approximation analysis

In section III-B the description of the graph-convolution layer is reported with a particular focus on the implementation of the aggregation weight matrix. A low-rank approximation of maximum rank $r$ of the weight matrix is enforced in order to limit the memory occupation and computation complexity as well as reduce vanishing gradient effects. We performed several tests to study the behaviour of the network as a function of the chosen maximum rank. Tables XIII and XIV report the results on the denoising with outlier removal and pure denoising tasks. We compared the value chosen for the experiments in the previous section ($r = 11$) with a low rank value (lowest complexity) and the highest rank that fit the GPU memory. It is possible to notice that it is not true that higher rank corresponds to an increment of the performance. On the contrary, we verify that $r = 11$ provides not only a desirable working point in terms of complexity but also achieves the best performance.

### B. Feature analysis

We analyze the characteristics of the receptive field, i.e., the set of points whose feature vectors influence the features

TABLE XIII
MAXIMUM RANK. DENOISING WITH OUTLIER REMOVAL PERFORMANCE.

| | **F1** | | | **C2C** ($\times 10^{-6}$) | | |
|---|---|---|---|---|---|---|
| $r$ | 0.02 | 0.015 | 0.01 | 0.02 | 0.015 | 0.01 |
| 3 | $89.01^{\pm 0.05}$ | $90.30^{\pm 0.06}$ | $91.05^{\pm 0.05}$ | $69.75^{\pm 0.09}$ | $41.56^{\pm 0.31}$ | $43.91^{\pm 0.13}$ |
| 11 | $\mathbf{89.43^{\pm 0.04}}$ | $\mathbf{91.37^{\pm 0.05}}$ | $\mathbf{92.56^{\pm 0.04}}$ | $\mathbf{65.96^{\pm 0.11}}$ | $\mathbf{38.20^{\pm 0.09}}$ | $\mathbf{39.97^{\pm 0.14}}$ |
| 18 | $89.22^{\pm 0.04}$ | $90.69^{\pm 0.05}$ | $91.65^{\pm 0.05}$ | $72.03^{\pm 0.30}$ | $39.74^{\pm 0.19}$ | $40.05^{\pm 0.14}$ |

TABLE XIV
MAXIMUM RANK. DENOISING PERFORMANCE.

| | **C2C** ($\times 10^{-6}$) | | |
|---|---|---|---|
| $r$ | 0.02 | 0.015 | 0.01 |
| 3 | $61.75^{\pm 0.05}$ | $37.35^{\pm 0.04}$ | $41.46^{\pm 0.08}$ |
| 11 | $\mathbf{61.48^{\pm 0.13}}$ | $\mathbf{36.88^{\pm 0.05}}$ | $41.47^{\pm 0.04}$ |
| 18 | $64.21^{\pm 0.07}$ | $37.11^{\pm 0.07}$ | $\mathbf{39.51^{\pm 0.06}}$ |

of a specific point, induced by the graph convolutional layers. In Fig. 7 we show an example of the receptive field of a single point for the output of the graph convolutional layers of a residual block with respect to the input of the residual block. The visualization is on the denoised point cloud. We observe that the receptive field is quite localized in the 3D space and its size increases as the number of layers increases. It is interesting to note that, since the graph is dynamically constructed in the feature space, the points of the receptive field are not just the spatially closest ones but they are also among the ones with similar shape characteristics. For example, in Fig. 7 the considered point is on the lower side of the chair stretcher and all the points of the receptive field belong to the same part of the surface.

In order to better analyze this non-local property of the receptive field we measure its radius in the 3D space and compare it to a fixed graph construction where the neighbors are determined by proximity in the noisy 3D space. Fig. 8 shows the radius of the receptive field of each point at the output of a residual block in the denoising module with respect to the input of the residual block. The radius is evaluated as the 90 percentile Euclidean distance in the 3D space on the clean point cloud (90 percentile is used since the maximum might be an unstable metric). It can be noticed that, when using the dynamic graph construction, the radius is only slightly larger in the first residual block but can be significantly larger in the second one. This can be interpreted as the feature space building and exploiting more and more non-local features with patterns similar to those in Fig. 7.

### C. Dynamic graph

An interesting study concerns the impact of the graph computation. The dynamic graph construction, as reported in Fig. 1, where the graph is computed in the feature space, is compared to a fixed one, where the neighbors are identified in the noisy 3D space. For the fixed graph construction the neighbors are computed from the original noisy input for the feature extraction and outlier removal blocks and from the noisy input without the detected outliers for the denoising block and the graph is used for all graph-convolutional layers inside the corresponding block. Table XV demonstrates that

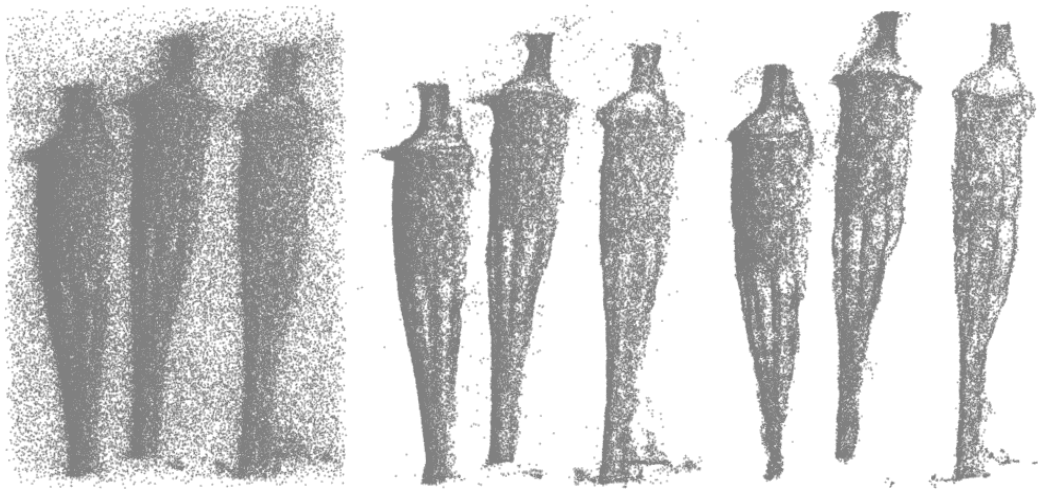

Fig. 6. Denoising results for real noise. Noisy point cloud (left), PointCleanNet (center), GPOD (right).

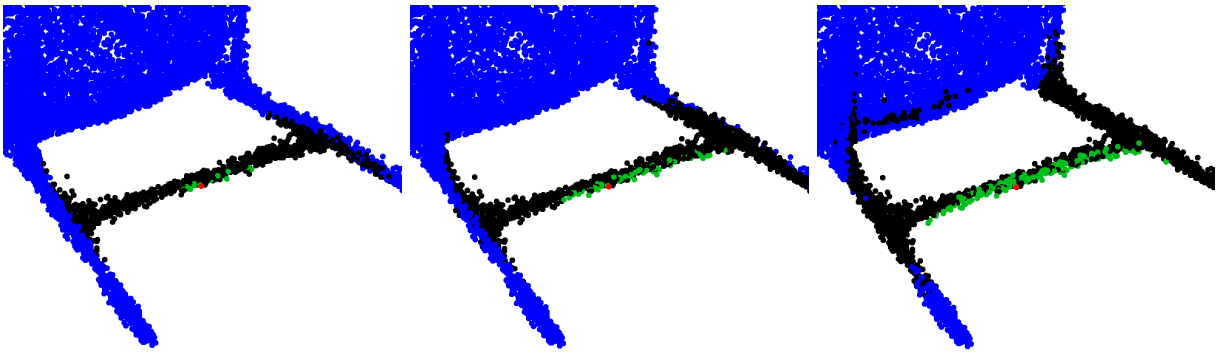

Fig. 7. Receptive field (green) and search area (black) of a point (red) for the output of the three graph-convolutional layers of the second residual block of the network with respect to the input of the first graph-convolutional layer in the block. Effective receptive field size: 16, 65, 189 points.

TABLE XV
FIXED VS. DYNAMIC GRAPH, $\sigma = 0.015$

| | Denoising with outlier removal | | Denoising | |
|---|---|---|---|---|
| | Dynamic | Fixed | Dynamic | Fixed |
| C2C ($\times 10^{-6}$) | 38.12 | 53.23 | 36.78 | 49.08 |

the use of a dynamic graph update improves the performance thanks to the possibility of finding and exploiting latent feature correlations.

### D. Neighborhood size

Another interesting study is the effect of neighborhood size. Selecting a larger number of neighbors for the graph-convolutional layer increases the size of the receptive field and can help denoise smooth areas in the point cloud by capturing more context, at the price of losing some localization and increased computational complexity. This is related to results on image denoising [49], where it is known that the optimal size of the receptive field depends on the noise variance. From Tables XVI and XVII it is possible to observe that increasing the number of neighbors is beneficial, up to a saturation point, and that the optimal number is neighbors actually depends on the noise variance. A blind method such as GPOD, therefore, encounters a tradeoff between good performance at high or low variance.

TABLE XVI
NUMBER OF NEIGHBORS (NN). DENOISING WITH OUTLIER REMOVAL PERFORMANCE.

| | **F1** | | | **C2C** ($\times 10^{-6}$) | | |
|---|---|---|---|---|---|---|
| NN | 0.02 | 0.015 | 0.01 | 0.02 | 0.015 | 0.01 |
| 4 | 86.72 | 91.08 | 93.68 | 81.59 | 49.12 | 35.00 |
| 8 | 89.85 | 91.47 | 92.45 | 69.40 | 41.16 | 35.62 |
| 16 | 89.43 | 91.41 | 92.50 | 65.88 | 38.12 | 39.94 |
| 24 | 89.28 | 90.98 | 92.06 | 61.95 | 39.09 | 43.52 |

TABLE XVII
NUMBER OF NEIGHBORS (NN). DENOISING PERFORMANCE.

| | **C2C** ($\times 10^{-6}$) | | |
|---|---|---|---|
| NN | 0.02 | 0.015 | 0.01 |
| 4 | 80.18 | 49.33 | 34.87 |
| 8 | 65.93 | 39.80 | 35.40 |
| 16 | 61.33 | 36.78 | 40.43 |
| 24 | 57.27 | 37.68 | 44.12 |

## VI. CONCLUSION

In this paper, we have presented the GPOD network, a novel graph-convolutional architecture able to efficiently remove outliers and denoise point clouds in a single blind model.

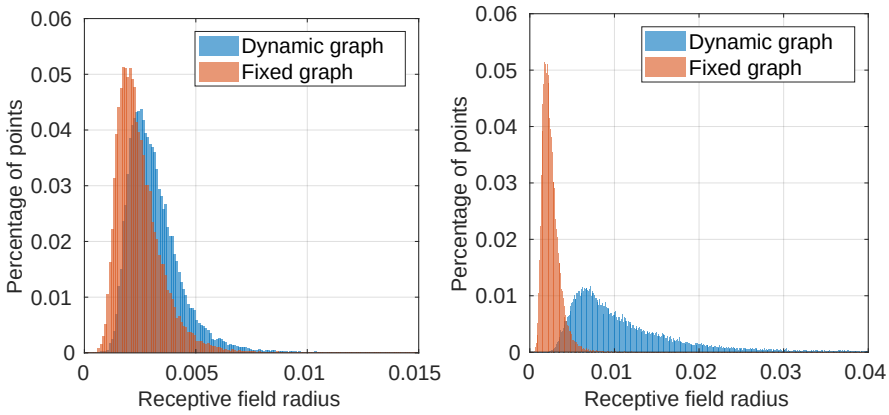


Fig. 8. Radius of receptive field of points at the output of residual block with respect to its input. Left: first residual block. Right: second residual block. Neighbor selection in the noisy 3D space for fixed graph and in the feature space for dynamic graph. Radius is measured as the 90 percentile Euclidean distance to the points in the receptive field on the clean 3D point cloud.

The core of the network is the graph-convolutional layer, that makes the proposed architecture fully convolutional and paves the way to learn hierarchies of features, emulating a classic CNN behaviour. The experimental results show promising performance at all levels of noise and especially significant improvements over state-of-the-art techniques at high levels of noise. Furthermore, it has been demonstrated that if additional noise information is available, the network performance can be further improved.

## References


[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[2] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[3] M. Simonovsky and N. Komodakis, "Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 29–38.

[4] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, p. 146, 2019.

[5] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia, "Deformable shape completion with graph convolutional autoencoders," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1886–1895.

[6] D. Valsesia, G. Fracastoro, and E. Magli, "Learning Localized Generative Models for 3D Point Clouds via Graph Convolution," in *International Conference on Learning Representations (ICLR) 2019*, 2019.

[7] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov, "POINTCLEANNET: Learning to Denoise and Remove Outliers from Dense Point Clouds," in *Computer Graphics Forum*. Wiley Online Library, 2019.

[8] C. Duan, S. Chen, and J. Kovacevic, "3D Point Cloud Denoising via Deep Neural Network Based Local Surface Estimation," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 8553–8557.

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

[10] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, July 2017.

[11] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," in *Advances in Neural Information Processing Systems*, 2018, pp. 1673–1682.

[12] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of dna-and rna-binding proteins by deep learning," *Nature biotechnology*, vol. 33, no. 8, pp. 831–838, 2015.

[13] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, 2015, pp. 2224–2232.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[15] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[16] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[17] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[19] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[20] N. Verma, E. Boyer, and J. Verbeek, "Feastnet: Feature-steered graph convolutions for 3d shape analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2598–2606.

[21] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2018.

[22] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 296–10 305.

[23] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković, "Principal neighbourhood aggregation for graph nets," *arXiv preprint arXiv:2004.05718*, 2020.

[24] D. Valsesia, G. Fracastoro, and E. Magli, "Deep Graph-Convolutional Image Denoising," *arXiv preprint arXiv:1907.08448*, 2019.

[25] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on visualization and computer graphics*, vol. 9, no. 1, pp. 3–15, 2003.

[26] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 493–501.

[27] G. Guennebaud and M. Gross, "Algebraic point set surfaces," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 23.

[28] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 22.

[29] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang, "Edge-aware point set resampling," *ACM transactions on graphics (TOG)*, vol. 32, no. 1, p. 9, 2013.

[30] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Computer Aided Geometric Design*, vol. 22, no. 2, pp. 121–146, 2005.
[31] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, "l1-sparse reconstruction of sharp point set surfaces," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 5, p. 135, 2010.
[32] Y. Sun, S. Schaefer, and W. Wang, "Denoising point sets via l0 minimization," *Computer Aided Geometric Design*, vol. 35, pp. 2–15, 2015.
[33] E. Mattei and A. Castrodad, "Point cloud denoising via moving RPCA," in *Computer Graphics Forum*, vol. 36, no. 8. Wiley Online Library, 2017, pp. 123–137.
[34] J. Zeng, G. Cheung, M. Ng, J. Pang, and C. Yang, "3D point cloud denoising using graph Laplacian regularization of a low dimensional manifold model," *arXiv preprint arXiv:1803.07252*, 2018.
[35] C. Dinesh, G. Cheung, and I. V. Bajic, "3D Point Cloud Denoising via Bipartite Graph Approximation and Reweighted Graph Laplacian," *arXiv preprint arXiv:1812.07711*, 2018.
[36] Y. Schoenenberger, J. Paratte, and P. Vandergheynst, "Graph-based denoising for time-varying point clouds," in *2015 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*. IEEE, 2015, pp. 1–4.
[37] P. Hermosilla, T. Ritschel, and T. Ropinski, "Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning," *arXiv preprint arXiv:1904.07615*, 2019.
[38] R. Roveri, A. C. Öztireli, I. Pandele, and M. Gross, "Pointpronets: Consolidation of point clouds with convolutional neural networks," in *Computer Graphics Forum*, vol. 37, no. 2. Wiley Online Library, 2018, pp. 87–99.
[39] S. Luo and W. Hu, "Differentiable manifold reconstruction for point cloud denoising," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1330–1338. [Online]. Available: https://doi.org/10.1145/3394171.3413727
[40] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, and L. Xiao, "A review of algorithms for filtering the 3d point cloud," *Signal Processing: Image Communication*, vol. 57, pp. 103–112, 2017.
[41] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.
[42] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D Point Cloud Based Object Maps for Household Environments," *Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge in Robotics)*, vol. 56, no. 11, pp. 927–941, 30 November 2008.
[43] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.
[44] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, Eds. The Eurographics Association, 2008.
[45] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3460–3464.
[46] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "BlenSor: Blender Sensor Simulation Toolbox," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, S. Wang, K. Kyungnam, B. Benes, K. Moreland, C. Borst, S. DiVerdi, C. Yi-Jen, and J. Ming, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 199–208.
[47] R. T. Collins, "A space-sweep approach to true multi-image matching." in *CVPR*. IEEE Computer Society, 1996, pp. 358–363.
[48] K. Wolff, C. Kim, H. Zimmer, C. Schroers, M. Botsch, O. Sorkine-Hornung, and A. Sorkine-Hornung, "Point cloud noise and outlier removal for image-based 3d reconstruction," in *Proceedings of International Conference on 3D Vision (3DV)*, October 2016.
[49] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 2392–2399.

**Francesca Pistilli** (IEEE Student Member) received the M.Sc. degrees in Electronic Engineering from Politecnico di Torino and Electrical and Computer Engineering from the University of Illinois at Chicago, Chicago, IL, in 2019 and 2020 respectively. She is currently a Ph.D. student at the Image Processing and Learning group (IPL) at Politecnico di Torino. Her current research interests include deep learning applied to image and point cloud processing.

**Giulia Fracastoro** (IEEE Member) Giulia Fracastoro received the Ph.D. degree in Electronic and Telecommunications Engineering from Politecnico di Torino, Turin, Italy, in 2017. During 2016, she was a visiting student at the Signal Processing Laboratory at EPFL, working on graph learning for image compression. She is currently an Assistant Professor with the Department of Electronics and Telecommunications (DET), Politecnico di Torino. Her research interests include graph signal processing, image processing, and deep learning.

**Diego Valsesia** (IEEE Member) received the M.Sc. degree in telecommunications engineering from the Politecnico di Torino, Turin, Italy, in 2012, the M.Sc. degree in electrical and computer engineering from the University of Illinois at Chicago, Chicago, IL, in 2013, and the Ph.D. degree in electronic and communication engineering from the Politecnico di Torino, in 2016. He is currently an Assistant Professor with the Department of Electronics and Telecommunications (DET), Politecnico di Torino. His main research interests include processing of remote sensing images, compressed sensing, and deep learning.

**Enrico Magli** (IEEE Fellow) received the M.Sc. and Ph.D. degrees from the Politecnico di Torino, Torino, Italy, in 1997 and 2001, respectively. He is currently a Full Professor with Politecnico di Torino, Torino, Italy. His research interests include compressive sensing, image and video coding, and vision. He is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and the EURASIP Journal on Image and Video Processing, and a former Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA. He is a Fellow of the IEEE, and has been an IEEE Distinguished Lecturer from 2015 to 2016. He was the recipient of the IEEE Geoscience and Remote Sensing Society 2011 Transactions Prize Paper Award, the IEEE ICIP 2015 Best Student Paper Award (as senior author), and the 2010 and 2014 Best Associate Editor Award of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.