POLITECNICO DI TORINO
Repository ISTITUZIONALE

Learning Graph-Convolutional Representations for Point Cloud Denoising

Original
Learning Graph-Convolutional Representations for Point Cloud Denoising / Pistilli, F., Fracastoro, G., Valsesia, D., Magli, E.. - ELETTRONICO. - 12365:(2020), pp. 103-118. (European Conference on Computer Vision ECCV 2020 Glasgow (UK) August 23–28, 2020) [10.1007/978-3-030-58565-5_7].

Availability:
This version is available at: 11583/2844374 since: 2021-01-21T11:13:51Z

Publisher:
Springer

Published
DOI:10.1007/978-3-030-58565-5_7

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Springer postprint/Author's Accepted Manuscript

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature’s AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/978-3-030-58565-5_7

(Article begins on next page)

11 August 2026

Learning Graph-Convolutional Representations for Point Cloud Denoising

Francesca Pistilli[0000−0001−9372−032X], Giulia Fracastoro[0000−0001−8495−1097], Diego Valsesia[0000−0003−1997−2910], and Enrico Magli[0000−0002−0901−0251]

Politecnico di Torino, Italy
francesca.pistilli@polito.it

Abstract. Point clouds are an increasingly relevant data type but they are often corrupted by noise. We propose a deep neural network based on graph-convolutional layers that can elegantly deal with the permutation-invariance problem encountered by learning-based point cloud processing methods. 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. When coupled with a loss promoting proximity to the ideal surface, the proposed approach significantly outperforms state-of-the-art methods on a variety of metrics. In particular, it is able to improve in terms of Chamfer measure and of quality of the surface normals that can be estimated from the denoised data. We also show that it is especially robust both at high noise levels and in presence of structured noise such as the one encountered in real LiDAR scans.

Keywords: Point cloud, denoising, graph neural network

1 Introduction

A point cloud is a geometric data type consisting in an unordered collection of 3D points representing samples of 2D surfaces of physical objects or entire scenes. Point clouds are becoming increasingly popular due to the availability of instruments such as LiDARs and the interest in exploiting the richness of the geometric representation in challenging applications such as autonomous driving. However, the acquisition process is imperfect and a significant amount of noise typically affects the raw point clouds. Therefore, point cloud denoising methods are of paramount importance to improve the performance of various downstream tasks such as shape matching, surface reconstruction, object segmentation and more.

Traditional model-based techniques [1–6] have typically focused on fitting a surface to the noisy data. Such techniques work well in low-noise settings but they usually suffer from oversmoothing, especially in presence of high amounts of noise or geometries with sharp edges. Given the success of learning-based methods, in particular those exploiting deep neural networks, in a wide variety of tasks,

including image denoising and restoration problems [7–9], a few works have recently started exploring point cloud denoising with deep neural networks. The most challenging problems in processing point clouds are the lack of a regular domain, such as a grid, and the fact that a point cloud is just a set of points and any permutation of them still represents the same data. Any learning-based method must therefore learn a *permutation-invariant* function that can deal with data defined on an irregular domain. This is a significant challenge that point cloud processing algorithms tackled by either approximating the irregular domain with a grid, e.g. by building voxels, or building a permutation-invariant function 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 [10]. Neither of these solutions is completely satisfactory. The former introduces an undesirable approximation, while the latter lacks the expressiveness of convolutional neural networks (CNN) where the convolution operation extracts features that are localized as functions of the neighborhood of a pixel and features of features are assembled in a hierarchical manner by means of multiple layers, progressively expanding the receptive field. Recently, graph convolution [11] has emerged as an elegant way to build operators that are intrinsically permutation-invariant and defined on irregular domains, while also exploiting some of the useful properties of traditional convolution, such as localization and compositionality of the features as well as efficient weight reuse. In particular, spatial-domain definitions of graph convolution have been recently applied in several problems involving point clouds such as classification [12], segmentation [13], shape completion [14] and generation [15]. Notably, the point cloud denoising problem has yet to be addressed with graph-convolutional neural networks.

In this paper, we propose a deep graph-convolutional neural network for denoising of point cloud geometry. 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 [16, 17]. 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. Extensive experimental results show a significant 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.

2 Related work

The literature on 3D point cloud denoising is vast and it can be subdivided into four categories: local surface fitting methods [1–6], sparsity-based methods [18–20], graph-based methods [21–23], and learning-based methods [16, 24, 17, 25]. Among the methods belonging to the first category, the moving least squares

(MLS) approach [1] and its robust extensions [2, 3] are the most widely used. Other surface fitting methods have also been proposed for point cloud denoising, such as jet fitting [6] or parameterization-free local projector operator (LOP) [4, 5]. These methods achieve remarkable performance at low levels of noise, but they suffer from over-smoothing when the noise level is high [26].

A second class of point cloud denoising methods [18–20] is based on sparse representations. In this case, the denoising procedure is composed of 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 [19].

Another approach for point cloud denoising is derived from the theory of graph signal processing [27]. These methods [21–23] 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 [16, 24, 17, 25], 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. However, in the context of shape classification and segmentation, many methods have recently been proposed specifically to handle point cloud data. PointNet [10] 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 [16] uses an approach similar to PointNet in order to estimate correction vectors for the points in the noisy point cloud. Instead, in [17] the authors 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 [25] 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.

Graph-convolutional networks have shown promising performance on tasks such as segmentation and classification. In particular, DGCNN [13] first introduced the idea of a dynamic graph update in the hidden layers of a graph-convolutional network. However, the denoising problem is significantly different from the classification and segmentation tasks addressed in [13], that rely more on global features instead of localized representations. 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 and also significantly increases the computational complexity for large point clouds; the 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 [13], 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.

3 Proposed method

In this section we present the proposed Graph-convolutional Point Denoising Network (GPDNet), i.e., a deep neural network architecture to denoise the geometry of point clouds based on graph-convolutional layers. The focus of the paper is to investigate the potential of graph convolution as a simple and elegant way of dealing with the permutation invariance problem encountered when processing point clouds. For this reason, we focus on analyzing the network in a discriminative learning setting where a clean reference is available and it is perturbed with white Gaussian noise. We refer the reader to [24] for a technique to train any point cloud denoising network in an unsupervised fashion only using noisy data.

3.1 Architecture

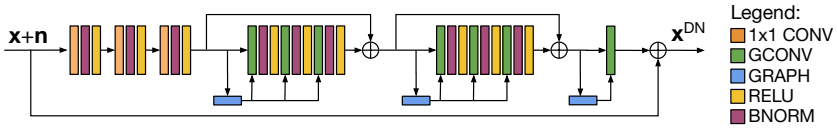

Fig. 1. GPDNet: graph-convolutional point cloud denoising network.

An overview of the architecture of GPDNet is shown in Fig. 1. At a high level, it is a residual network that estimates the noise component of the input point cloud, which has been shown [7] to be easier than directly cleaning the data. A first block is composed of three single-point convolutions that gradually transform the 3D space into an F-dimensional feature space. Then a cascade of two residual blocks is used, having an input-output skip connection to reduce vanishing gradient issues. Each residual block is composed of three graph-convolutional layers. The graph is computed by selecting the k nearest neighbors

to each point in terms of Euclidean distances in the feature space. Notice that the graph construction is dynamic, i.e., it is updated after every residual block but shared among the graph convolutional layers inside the block to limit computational complexity. Dynamic construction of a similarity graph has been shown to induce more powerful feature representations [13, 15] and, in the context of a residual denoising network, it progressively uncovers the latent correlations that have not yet been eliminated. Intuitively, dynamic graph construction is preferable over building the graph in the noisy 3D space as neighborhoods might be strongly perturbed at high noise variances, leading to unstable or sub-optimal neighbor assignments. All layers are interleaved with batch normalization which stabilizes training, especially in presence of Gaussian noise. Finally, the last graph-convolutional layer projects the features back to the 3D space.

3.2 Graph-convolutional layer

The core of the proposed architecture is the graph-convolutional layer. Graph convolution is a generalization of convolution to data that are defined over the nodes of a general graph rather than a grid. Multiple definitions of graph convolution have been proposed to capture salient properties of classical convolution, notably localization and weight reuse. In this paper, we use a modified version of the Edge-Conditioned Convolution (ECC) [12] to address vanishing gradient and over-parameterization. In particular, we use some of the approximations introduced in [28] in the context of image denoising.

The 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. The output feature vector $\mathbf{h}_i^{l+1} \in \mathbb{R}^{F^l}$ of point i at layer l is computed by performing a weighted aggregation over its neighborhood $\mathcal{N}_i^l$ as defined by the graph:

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

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 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 scalar $\omega_t^{j \to i}$ are computed as functions of the difference between the feature vector 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 (MLP) 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 value r is a hyperparameter setting the maximum rank of the aggregation weight matrix obtained by explicitly computing $\sum_{t=1}^{r} \omega_t^{j \to i} \boldsymbol{\phi}_t^{j \to i} \boldsymbol{\psi}_t^{j \to i^T}$, again to reduce the number of parameters and memory requirements of the aggregation operation. The parameter $\gamma^{l,j \to i}$ is a scalar edge attention term which exponentially depends on the Euclidean distance between feature vectors across

an edge:

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

being δ a decay hyperparameter.

This definition of graph convolution has some advantages over alternative definitions such as GraphSAGE [29], FeastNet [30] or DGCNN [13]. In particular, the aggregation weights are functions of feature differences making the filtering operation performed by the graph-convolutional layer *adaptive*. Moreover, since the function is implemented as an MLP, it can be more general than a fixed function with some learnable parameters.

The graph is constructed by searching for the k-nearest neighbors of each point in terms of Euclidean distance between their feature vectors. To limit complexity, 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 (see Fig. 4 for a visual representation of the search area and feature space neighborhoods).

We remark that GPDNet 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 a multi-point aggregation of the feature vectors of neighboring points in the previous layer, thus building complex hierarchies of aggregations. This is in contrast with PointCleanNet [16] which works by processing each patch independently to estimate the denoised version of the central point. That approach does not create hierarchies of features obtained by successive multi-point aggregations, as in a classical CNN. The graph-convolutional structure recovers this behavior and can learn more powerful feature spaces.

3.3 Loss functions

We consider two loss functions to train the proposed method in a supervised setting. The first one is the mean squared error (MSE) between the denoised point cloud $\hat{\mathbf{x}}$ and its noiseless ground truth $\mathbf{x}$, i.e.:

$$L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 \tag{3}$$

being N the number of points in the point cloud. This is the most natural choice in presence of Gaussian noise. However, it does not exploit prior knowledge about the distribution of points. In fact, it does not use 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 can be incorporated by regularizing the MSE loss with a term measuring the distance of the denoised point from the ground truth surface. Such measure can be approximated by the proximity to surface metric which computes the distance between the denoised point and the closest ground truth

point. The loss function (MSE-SP) then becomes:

$$L_{\text{MSE}-\text{SP}} = \frac{1}{N}\sum_{i=1}^{N}\left[\|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 + \lambda \min_j \|\hat{\mathbf{x}}_i - \mathbf{x}_j\|_2^2\right] \tag{4}$$

for a regularization hyperparameter λ. Other works also considered proximity to surface in the loss function. Notably, PointCleanNet [16] 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.

4 Experimental results

In this section an experimental evaluation against state-of-the-art approaches as well as an analysis of the proposed technique is performed. Code is available[1].

4.1 Experimental setting

The training and test set are created selecting post-processed subsets of ShapeNet [31] repository. This database is composed by 3D models of 55 object categories, each one described by a collection of meshes. Before utilization, the data has to be sampled and normalized. First we sample 30720 uniformly distributed points for each model, then we rescale the obtained point clouds normalizing their diameter in order to ensure that data are at the same scale. More than 100000 patches of 1024 points each are randomly selected from the point clouds to create the training set, taking point clouds from all the categories except 10 reserved for the test set. Each patch is created by randomly selecting a point from a point cloud and collecting its 1023 closest points. The test set is constituted by 100 point clouds taken from ten different categories: airplane, bench, car, chair, lamp, pillow, rifle, sofa, speaker, table. We randomly select ten models from each category and sample 30720 uniformly distributed points from each model.

GPDNet is trained for a fixed noise variance for approximately 700000 iterations, each one characterized by a batch size of 16. The number of features used for all the layers is 99, except for the first three single-point convolutional layers 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^{-4}. Concerning the graph-convolutional implementation, the rank r for the low-rank approximation is set to 11, 3 circulant rows are considered for the construction of the circulant matrix, and $\delta = 10$. During testing, GPDNet 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 in terms of Euclidean distances are used for graph construction.

[1] https://github.com/diegovalsesia/GPDNet

4.2 Comparisons with state-of-the-art

In this section the proposed method is compared with state-of-the-art methods for point cloud denoising. As described in Section 2, different categories of point cloud denoising methods are present in the literature. In the experiments, we take into account at least one algorithm from each category. APSS [3] and RIMLS [2] are well-known MLS-based surface fitting methods and they were tested using the MeshLab software [32]. AWLOP [5] is another surface fitting method and it was implemented using the software released by the authors. MRPCA [20] is a sparsity-based method and it was implemented using the code provided by the authors. GLR [21] is one of the most promising works belonging to the graph-based category and it was implemented using the code provided by the authors. PointCleanNet (PCN) [16] is one of the most recent learning-based methods and its code is publicly available. In order to ensure a fair comparison, PointCleanNet was retrained with additive Gaussian noise at a specific standard deviation, instead of using the blind model released by the authors. We also include a modified version of DGCNN [13] as an additional baseline. This modified version replaces the segmentation head with a single-point convolution to regress the point displacement.

As metric to evaluate the performance of the proposed method, we compute the Chamfer measure, also called Cloud-to-Cloud (C2C) distance. 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 then their average:

$$\text{C2C} = \frac{1}{2N}\left[\sum_{i=1}^{N} \min_{j} \|\hat{\mathbf{x}}_i - \mathbf{x}_j\|_2^2 + \sum_{j=1}^{N} \min_{i} \|\hat{\mathbf{x}}_i - \mathbf{x}_j\|_2^2\right]. \tag{5}$$

The results of the experiments at different noise levels are reported in Table 1. As described in Sec. 3.3, in the proposed network we consider two different loss functions obtaining two versions of the proposed method, namely GPDNet MSE and GPDNet MSE-SP. It is clearly visible that both versions of the proposed method significantly outperform state-of-the-art methods, especially at medium and high levels of noise, as shown in Table 1 with $\sigma = 0.015$ and $\sigma = 0.02$. Instead, at low noise level the other algorithms become more competitive and the performance gap decreases, but the proposed method still obtains the best results in the majority of the categories. This can be explained by the fact that most of the other methods involves surface reconstruction or normal estimation, operations that cannot be computed with sufficient accuracy at high levels of noise. Instead, the proposed method directly estimates the denoised point cloud. In addition, it can be observed from Table 1 that the GPDNet MSE-SP version is particularly effective at high levels of noise, outperforming GPDNet MSE in almost all the categories. This behavior can be explained by the regularizing effect of the surface distance component of the loss, which is especially useful at

Table 1. Chamfer measure ($\times 10^{-6}$), 16-NN, N=Noisy.

Class	N	DGCNN [13]	APSS [3]	RIMLS [2]	AWLOP [5]	MRPCA [20]	GLR [21]	PCN [16]	**GPD MSE**	**GPD MSE-SP**
$\sigma = 0.01$										
airp.	50	44.82	28.22	39.73	31.27	28.19	19.56	26.36	**17.22**	17.58
bench	49	38.70	26.97	32.76	34.08	32.93	20.43	27.64	**19.33**	19.80
car	64	60.47	47.73	55.56	54.21	44.33	42.22	75.34	**38.09**	38.14
chair	61	59.69	37.31	45.65	47.91	38.41	34.98	55.10	**29.50**	29.69
lamp	60	52.54	24.57	34.02	35.23	31.51	19.67	20.58	**16.17**	17.15
pill.	70	64.28	**15.64**	21.23	46.36	23.95	17.59	21.07	17.11	19.04
rifle	39	26.99	36.01	49.37	27.79	23.49	15.84	15.09	14.45	**14.00**
sofa	70	65.05	**22.27**	28.04	53.08	32.14	30.88	43.36	25.87	27.21
speak.	74	68.72	**26.50**	30.19	58.92	47.57	40.78	76.09	34.87	35.81
table	56	50.17	27.45	32.63	41.26	34.78	27.12	43.02	**24.27**	24.64
$\sigma = 0.015$										
airp.	98	84.40	86.42	106.33	73.32	67.39	36.76	35.27	28.47	**27.62**
bench	95	64.76	75.51	91.93	82.04	70.05	32.19	30.10	28.72	**26.96**
car	102	93.43	72.56	103.52	93.38	69.88	55.92	92.23	52.92	**51.77**
chair	105	94.4 5	81.47	104.38	92.47	73.45	48.62	69.18	46.28	**43.73**
lamp	121	112.06	65.79	82.40	88.78	77.09	39.93	30.59	**27.37**	28.60
pill.	133	113.32	22.74	42.54	112.54	73.67	31.38	29.02	**23.32**	27.25
rifle	80	61.04	92.14	110.51	69.35	55.65	31.81	**21.45**	28.43	22.48
sofa	121	99.63	42.80	69.92	107.58	72.62	51.12	61.15	**40.10**	42.04
speak.	123	114.12	46.45	58.28	110.29	77.95	53.75	87.68	**49.20**	49.57
table	104	84.95	62.64	78.21	89.33	70.87	37.94	43.88	36.06	**33.89**
$\sigma = 0.02$										
airp.	162	127.44	175.68	186.24	145.94	123.71	90.55	74.17	45.96	**42.30**
bench	162	99.36	166.85	182.42	157.29	127.51	83.99	90.34	41.24	**36.77**
car	149	113.94	141.69	167.78	145.51	109.49	77.56	160.08	72.06	**67.43**
chair	163	132.91	160.01	155.38	158.12	122.70	79.85	145.56	67.91	**60.16**
lamp	204	153.02	178.08	198.22	187.31	146.41	109.24	85.31	45.21	**44.60**
pill.	216	190.32	164.83	196.53	206.14	150.65	85.86	92.84	**34.47**	38.58
rifle	144	131.91	195.68	176.07	144.22	105.87	89.19	71.57	43.07	**29.55**
sofa	184	155.51	166.34	190.91	178.93	133.98	89.31	144.72	**62.58**	65.06
speak.	186	136.72	138.80	162.34	180.45	126.17	84.37	160.26	66.57	**63.40**
table	168	115.00	171.25	179.81	162.36	125.72	78.06	102.17	50.47	**44.80**

Table 2. Unoriented normal angle error (degrees), 16-NN.

σ	DGCNN [13]	APSS [3]	RIMLS [2]	AWLOP [5]	MRPCA [20]	GLR [21]	PCN [16]	**GPDNet MSE**	**GPDNet MSE-SP**
0.01	30.83	22.60	24.52	29.79	31.40	21.90	26.85	**20.11**	22.33
0.015	32.52	31.83	37.35	32.17	39.97	25.99	27.54	**21.16**	24.46
0.02	32.31	42.42	45.86	33.41	42.45	31.30	28.65	**22.78**	27.06

high noise variance due to the fact that it can incorporate more prior knowledge about the data. The performance difference between the two variants decreases at low noise levels. It is worth noting that DGCNN shows poor performance for the reasons explained in Sec. 2, being originally designed for classification or segmentation. This is in line with the results presented in the PointCleanNet paper [16] where the authors also show the poor denoising performance of DGCNN, and highlights the importance of the design in this paper, which is tailored to the denosing task.

We also consider another metric for a quantitative assessment of the denoised point clouds. In particular, we assess whether an off-the-shelf algorithm for surface normal estimation can produce more accurate normals when provided with point clouds denoised by the proposed method. Since surface normals are widely used in many applications, we believe that measuring their quality when extracted from the denoised data is a relevant metric for the characterization of a denoiser. In this experiment we consider a different test set, composed of 5 well-known point clouds: Armadillo, Bunny, Column, Galera and Tortuga. The change of dataset is motivated by the availability of ground truth normals for these point clouds. For every denoising method considered in the comparison, we compute the unoriented normal vector of each point in the denoised point cloud. The standard algorithm employed for the normal estimation is the built-in MATLAB function, which is based on principal component analysis. We compute the unoriented normal angle error (UNAE) as

$$\text{UNAE} = \frac{1}{N}\sum_{i=1}^{N} \arccos\left[1 - \frac{1}{2}\min\left(\|\hat{\mathbf{n}}_{i^*} - \mathbf{n}_i\|_2^2, \|\hat{\mathbf{n}}_{i^*} + \mathbf{n}_i\|_2^2\right)\right], \tag{6}$$

where $\mathbf{n}_i$ is the groud-truth normal vector at $\mathbf{x}_i$ and $\hat{\mathbf{n}}_{i^*}$ is the estimated normal vector at the denoised point closest to $\mathbf{x}_i$. Table 2 reports the average error across the five test point clouds. A minimum error of 6.44 degrees is measured since the MATLAB algorithm introduces a non-zero estimation error in the computation of the normals, as can be seen from the first column of Table 2. The error on the noisy data is 31.13, 32.77 and 33.77 degrees respectively. It can be observed that the proposed denoising method, in particular the version with only MSE as loss function, increases the accuracy of the normal estimation, outperforming the state-of-the-art at each noise level considered. It is also interesting to notice that learning-based methods are more stable to noise than model-based methods as their performance degrades more gracefully for increasing noise variance.

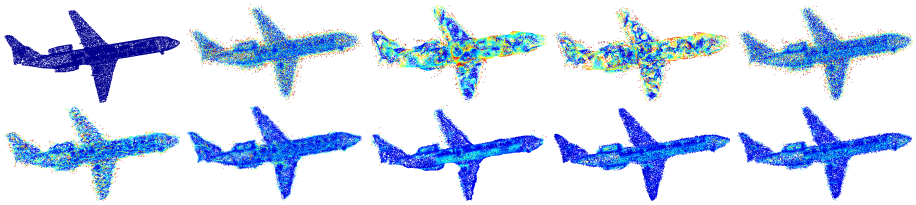

Fig. 2. Denosing results for $\sigma = 0.015$. Color represents distance to surface (red is high, blue is low). Top left to bottom right: clean point cloud, DGCNN (RMSD = 0.0091), APSS (0.0123), RIMLS (0.0127), AWLOP (0.0106), MRPCA (0.0096), GLR (0.0070), PointCleanNet (0.0065), GPDNet MSE (0.0060), GPDNet MSE-SP (0.0062).

Fig. 2 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:

$$\text{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \min_{j} \|\hat{\mathbf{x}}_i - \mathbf{x}_j\|_2^2}. \tag{7}$$

It can be seen that on average both versions of our method provide lower points-surface distance and that the shape of the reconstructed point cloud is more similar to the original one. Fig. 3 shows another qualitative comparison, displaying the unoriented normal estimation error for each denoised point. It can be seen that GPDNet, especially the MSE variant, provides lower normal estimation errors, highlighting the higher quality of the denoised point cloud.

4.3 Ablation studies

Table 3. Fixed vs. Dynamic graph, $\sigma = 0.015$, 8-NN.

	GPDNet MSE		**GPDNet MSE-SP**	
	Dynamic	Fixed	Dynamic	Fixed
C2C ($\times 10^{-6}$)	**35.68**	37.00	**36.99**	38.45
UNAE (degrees)	**23.56**	23.75	**26.29**	26.65

We study the behavior of GPDNet in terms of a few design choices. In particular, we first investigate the impact of dynamic graph computation, i.e., updating the graph from the hidden feature space as in Fig. 1, as opposed to a fixed graph construction where neighbors are identified in the noisy 3D space and used for all graph-convolutional layers. Table 3 shows that dynamic graph update provides improved performance thanks to refined neighbor selection.

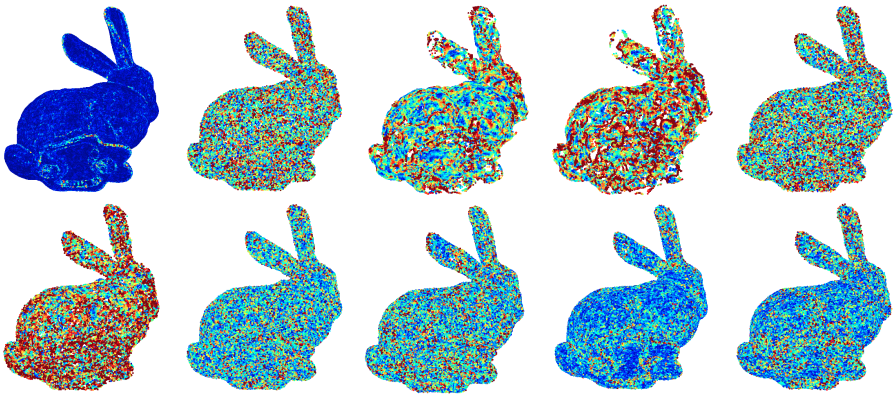

Fig. 3. Denosing results for $\sigma = 0.015$. Color is per-point UNAE (red is high, blue is low). Top left to bottom right: clean point cloud (UNAE = 3.75°), DGCNN (29.73°), APSS (26.29°), RIMLS (33.63°), AWLOP (29.18°), MRPCA (37.50°), GLR (22.08°), PointCleanNet (23.63°), GPDNet MSE (16.62°), GPDNet MSE-SP (23.11°).

We also study the impact of neighborhood size on the overall performance. Selecting a larger number of neighbors for graph convolution 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 [33], where it is known that the optimal size of the receptive field depends on the noise variance. Table 4 shows that increasing the number of neighbors is beneficial, up to a saturation point. We also see that the impact of a larger receptive field is more significant for the GPDNet MSE-SP variant.

Table 4. Number of neighbors.

			4-NN	8-NN	16-NN	24-NN
C2C ($\times 10^{-6}$)	$\sigma = 0.01$	GPDNet MSE	28.27	24.43	**23.69**	23.84
		GPDNet MSE-SP	30.38	25.54	24.31	24.44
	$\sigma = 0.015$	GPDNet MSE	40.46	35.68	36.09	36.67
		GPDNet MSE-SP	46.05	36.99	**35.39**	35.80
	$\sigma = 0.02$	GPDNet MSE	58.88	50.34	52.96	55.45
		GPDNet MSE-SP	64.63	51.82	**49.26**	50.43
UNAE (degrees)	$\sigma = 0.01$	GPDNet MSE	27.22	22.51	**20.11**	20.89
		GPDNet MSE-SP	29.04	24.10	22.33	22.16
	$\sigma = 0.015$	GPDNet MSE	28.03	23.56	**21.16**	21.18
		GPDNet MSE-SP	31.31	26.29	24.46	23.80
	$\sigma = 0.02$	GPDNet MSE	31.09	25.67	**22.78**	22.98
		GPDNet MSE-SP	32.00	28.81	27.06	26.92

4.4 Feature analysis

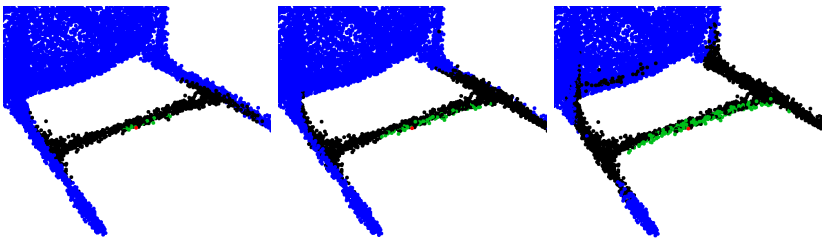

Fig. 4. Receptive field (green) and search area (black) of a point (red) for the output of the three graph-convolutional layers of the second residual block of the network with respect to the input of the first graph-convolutional layer in the block. Effective receptive field size: 16, 65, 189 points.

We analyze the characteristics of the receptive field, i.e., the set of points whose feature vectors influence the features of a specific point, induced by the graph convolutional layers. In Fig. 4 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. 4 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. 5 shows the radius of the receptive field of each point at the output of a residual block 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. 4.

4.5 Structured noise

In order to check if the proposed architecture can generalize beyond white Gaussian noise, we train it on a simulated LiDAR dataset. We simulate scanning the

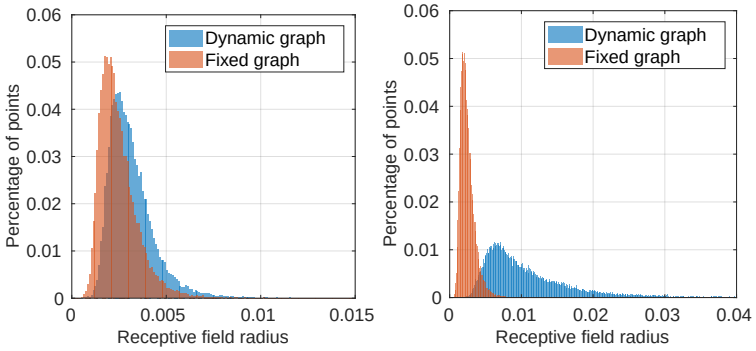

Fig. 5. Radius of receptive field of points at the output of residual block with respect to its input. Left: first residual block. Right: second residual block. Neighbor selection in the noisy 3D space for fixed graph and in the feature space for dynamic graph. Radius is measured as the 90 percentile Euclidean distance to the points in the receptive field on the clean 3D point cloud.

Table 5. Velodyne scan structured noise, RMSD, 8-NN.

Noisy	PointCleanNet	**GPDNet MSE**	**GPDNet MSE-SP**
0.1447	0.0966	0.0664	**0.0602**

Shapenet objects with a Velodyne HDL-64E scanner using the Blensor software [34]. 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 state-of-the-art model. Table 5 shows that the results follow those on white Gaussian noise, with the proposed method improving over PointCleanNet. RMSD is used as metric instead of the Chamfer measure since it is better suited to when points are not uniformly distributed.

5 Conclusions

In this paper, we have presented a new graph-convolutional neural network targeted for point cloud denoising. Thanks to the graph-convolutional layers, the proposed architecture is fully convolutional and can learn hierarchies of features, showing a behaviour similar to standard CNNs. The experimental results show that the proposed method provides a significant improvement over state-of-the-art techniques. In particular, the proposed method is robust to high level of noise and structured noise distributions, such as those observed in real LiDAR scans.

References

1. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Computing and rendering point set surfaces. IEEE Transactions on visualization and computer graphics **9**(1) (2003) 3–15
2. Öztireli, A.C., Guennebaud, G., Gross, M.: Feature preserving point set surfaces based on non-linear kernel regression. In: Computer Graphics Forum. Volume 28., Wiley Online Library (2009) 493–501
3. Guennebaud, G., Gross, M.: Algebraic point set surfaces. In: ACM Transactions on Graphics (TOG). Volume 26., ACM (2007) 23
4. Lipman, Y., Cohen-Or, D., Levin, D., Tal-Ezer, H.: Parameterization-free projection for geometry reconstruction. In: ACM Transactions on Graphics (TOG). Volume 26., ACM (2007) 22
5. Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U., Zhang, H.R.: Edge-aware point set resampling. ACM transactions on graphics (TOG) **32**(1) (2013) 9
6. Cazals, F., Pouget, M.: Estimating differential quantities using polynomial fitting of osculating jets. Computer Aided Geometric Design **22**(2) (2005) 121–146
7. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. IEEE Transactions on Image Processing **26**(7) (July 2017) 3142–3155
8. Liu, D., Wen, B., Fan, Y., Loy, C.C., Huang, T.S.: Non-local recurrent network for image restoration. In: Advances in Neural Information Processing Systems. (2018) 1673–1682
9. Valsesia, D., Fracastoro, G., Magli, E.: Image denoising with graph-convolutional neural networks. In: 2019 IEEE International Conference on Image Processing (ICIP). (Sep. 2019) 2399–2403
10. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 652–660
11. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4) (2017) 18–42
12. Simonovsky, M., Komodakis, N.: Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (July 2017) 29–38
13. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) **38**(5) (2019) 146
14. Litany, O., Bronstein, A., Bronstein, M., Makadia, A.: Deformable shape completion with graph convolutional autoencoders. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 1886–1895
15. Valsesia, D., Fracastoro, G., Magli, E.: Learning Localized Generative Models for 3D Point Clouds via Graph Convolution. In: International Conference on Learning Representations (ICLR) 2019. (2019)
16. Rakotosaona, M.J., La Barbera, V., Guerrero, P., Mitra, N.J., Ovsjanikov, M.: POINTCLEANNET: Learning to Denoise and Remove Outliers from Dense Point Clouds. In: Computer Graphics Forum, Wiley Online Library (2019)
17. Duan, C., Chen, S., Kovacevic, J.: 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) 8553–8557

18. Avron, H., Sharf, A., Greif, C., Cohen-Or, D.: l1-sparse reconstruction of sharp point set surfaces. ACM Transactions on Graphics (TOG) **29**(5) (2010) 135
19. Sun, Y., Schaefer, S., Wang, W.: Denoising point sets via l0 minimization. Computer Aided Geometric Design **35** (2015) 2–15
20. Mattei, E., Castrodad, A.: Point cloud denoising via moving RPCA. In: Computer Graphics Forum. Volume 36., Wiley Online Library (2017) 123–137
21. Zeng, J., Cheung, G., Ng, M., Pang, J., Yang, C.: 3D point cloud denoising using graph Laplacian regularization of a low dimensional manifold model. arXiv preprint arXiv:1803.07252 (2018)
22. Dinesh, C., Cheung, G., Bajic, I.V.: 3D Point Cloud Denoising via Bipartite Graph Approximation and Reweighted Graph Laplacian. arXiv preprint arXiv:1812.07711 (2018)
23. Schoenenberger, Y., Paratte, J., Vandergheynst, P.: 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) 1–4
24. Hermosilla, P., Ritschel, T., Ropinski, T.: Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning. arXiv preprint arXiv:1904.07615 (2019)
25. Roveri, R., Öztireli, A.C., Pandele, I., Gross, M.: Pointpronets: Consolidation of point clouds with convolutional neural networks. In: Computer Graphics Forum. Volume 37., Wiley Online Library (2018) 87–99
26. Han, X.F., Jin, J.S., Wang, M.J., Jiang, W., Gao, L., Xiao, L.: A review of algorithms for filtering the 3d point cloud. Signal Processing: Image Communication **57** (2017) 103–112
27. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE signal processing magazine **30**(3) (2013) 83–98
28. Valsesia, D., Fracastoro, G., Magli, E.: Deep Graph-Convolutional Image Denoising. arXiv preprint arXiv:1907.08448 (2019)
29. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems. (2017) 1024–1034
30. Verma, N., Boyer, E., Verbeek, J.: Feastnet: Feature-steered graph convolutions for 3d shape analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 2598–2606
31. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)
32. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: MeshLab: an Open-Source Mesh Processing Tool. In Scarano, V., Chiara, R.D., Erra, U., eds.: Eurographics Italian Chapter Conference, The Eurographics Association (2008)
33. Burger, H.C., Schuler, C.J., Harmeling, S.: Image denoising: Can plain neural networks compete with BM3D? In: 2012 IEEE conference on computer vision and pattern recognition, IEEE (2012) 2392–2399
34. Gschwandtner, M., Kwitt, R., Uhl, A., Pree, W.: BlenSor: Blender Sensor Simulation Toolbox. In Bebis, G., Boyle, R., Parvin, B., Koracin, D., Wang, S., Kyungnam, K., Benes, B., Moreland, K., Borst, C., DiVerdi, S., Yi-Jen, C., Ming, J., eds.: Advances in Visual Computing, Berlin, Heidelberg, Springer Berlin Heidelberg (2011) 199–208