Point Cloud Normal Estimation with Graph-Convolutional Neural Networks

POINT CLOUD NORMAL ESTIMATION WITH GRAPH-CONVOLUTIONAL NEURAL NETWORKS

Francesca Pistilli, Giulia Fracastoro, Diego Valsesia, Enrico Magli

Politecnico di Torino, Italy

ABSTRACT

Surface normal estimation is a basic task for many point cloud processing algorithms. However, it can be challenging to capture the local geometry of the data, especially in presence of noise. Recently, deep learning approaches have shown promising results. Nevertheless, applying convolutional neural networks to point clouds is not straightforward, due to the irregular positioning of the points. In this paper, we propose a normal estimation method based on graph-convolutional neural networks to deal with such irregular point cloud domain. The graph-convolutional layers build hierarchies of localized features to solve the estimation problem. We show state-of-the-art performance and robust results even in presence of noise.

Index Terms— Point clouds, normal estimation, graph-convolutional neural networks

1. INTRODUCTION

The increased availability and quality of range-sensing instruments, such as LiDAR, has made point clouds a commonly used data type. A point cloud can be described as an unordered collection of 3D points sampled from an underlying surface. Estimating the normal vector to the surface is a fundamental task that is part of the pipeline of many point cloud processing algorithms addressing a variety of problems. For instance, normals can be used to apply shading effects in computer graphics [1], improve surface reconstruction [2], regularize segmentation [3] and denoising [4]. Therefore, improvements on normal estimation algorithms are significant as they benefit a large number of downstream tasks.

However, processing point clouds can be challenging due to their nature as unordered sets of points. Traditional model-based approaches typically resorted to fitting local geometric models of the surface [5, 6, 7, 8, 9, 10], encountering model difficulties in presence of noise. Recently, an interest in deep learning approaches has grown out of their capability of building more complex representations that can also be robust to noise. Nevertheless, extending deep learning approaches to point clouds is hard due to the lack of a grid-like domain and the permutation-invariance problem, i.e., the irrelevance of the ordering of points. PointNet [11] is one of the early approaches to use deep neural networks on point clouds for the classification problem; it addresses those issues by applying the same weights to all points and then merging the information with a globally-symmetric function (e.g., a max pool). However, this construction is sub-optimal as it does not exploit some of the useful properties that made convolutional neural networks (CNNs) so successful. In particular, it cannot create hierarchies of localized features where the hidden representation of a point is constructed from the features of its neighbors and then is in turn assembled with the other local representations to create higher-level features.

Graph-convolutional neural networks [12] are emerging as a state-of-the-art approach to deal with irregular domains in the form of a general graph. They are, therefore, naturally suited to process point clouds where a graph can be constructed to capture the spatial neighbors of points. The graph convolution operation used in such networks extends properties of traditional convolution such as weight reuse as well as locality and hierarchical compositionality of features to the graph domain. Indeed, graph-convolutional neural networks have already been used successfully to address classification [13], segmentation [14], and generation [15] tasks on point clouds.

In this paper, we present a graph-convolutional neural network to estimate unoriented surface normals from raw point clouds. Thanks to graph convolution, the network can create complex hierarchies of features with a dynamically expanding receptive field. This allows the proposed method to achieve state-of-the-art performance, providing robust estimates even in presence of noise.

2. RELATED WORK

The most well-known method for point cloud normal estimation uses Principal Component Analysis (PCA) [5]. This method selects a patch of a given size around a point and uses PCA regression in order to estimate the tangent plane. More sophisticated surface fitting techniques, such as jet fitting [6], moving least squares [7], and spherical fitting [10], have also been introduced. However, the performance of these methods is very sensitive to the choice of the patch size. If the patch size is too large, it can lead to oversmoothing, especially near sharp edges. On the other hand, if the patch size is too small,

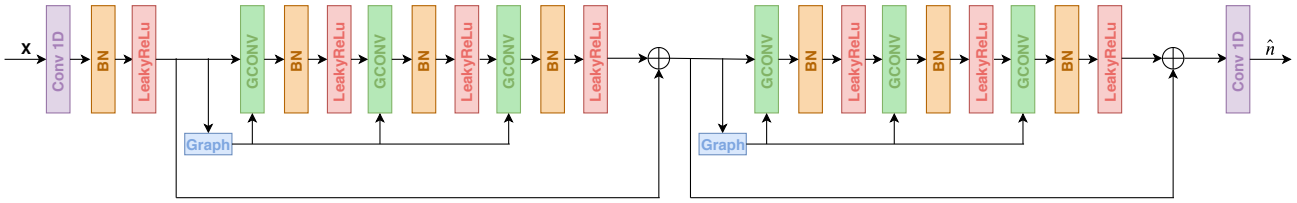


Fig. 1. Proposed architecture for normal estimation.

the method can be less robust to outliers.

A different set of approaches [8, 9, 16] employs the Voronoi diagram of the point cloud to estimate the normals of the points. These methods preserve sharp edges and they can provide strong theoretical guarantees on surface approximation and robustness. However, in practice they rely on a very careful fine tuning of the hyperparameters and they need a preprocessing step in presence of strong noise.

Recently, learning-based methods [17, 18, 19, 20, 21], especially the ones based on deep learning, have gained attention thanks to their capability of learning more sophisticated representations. However, since point clouds are unordered sets of points, applying convolutional neural networks to such data type is not straightforward. For this reason, some methods preprocess the point cloud in order to obtain a representation of the data that can be used as input of a standard convolutional neural network. For example, NestiNet [18] employs a multi-scale point statistics representation which encodes the local geometry on a grid, while Boulch et al. [20] use a Hough transform to obtain an image-like representation of the neighborhood of a point. This preprocessing step allows to use standard convolutional neural networks, at the expense of the representation power of the method, which is limited by the handcrafted point cloud features of choice. Instead, PCPNet [17] proposes a different approach, employing a network architecture similar to PointNet [11]. With this approach, each point of the local patch is processed independently and then points are aggregated using a global symmetric function. The main drawback of this approach is that it does not exploit the neighboring points in a hierarchical way like standard CNNs.

3. PROPOSED METHOD

In this section, we present the proposed graph-convolutional network to estimate the normal vector associated to each point of an input point cloud. We first present an overview of the architecture of the proposed network. Then, we describe in detail the graph-convolutional layer, which represents the core of the proposed architecture, and the loss function employed during training.

3.1. Architecture

An overview of the architecture is shown in Fig. 1. The network takes as input a patch of N points of a point cloud and estimates the unoriented normal vector associated to each point. First, the patch is normalized so that its points have zero mean and unit standard deviation. Then, the normalized 3D coordinates of the input point cloud are projected onto an F-dimensional feature space employing a single-point convolutional layer followed by a batch-normalization block and an activation function. The rest of the network can be described as a series of two residual blocks, where a skip connection sums the feature vectors from the input of the block with the feature vectors at its output. Residual connections are well known to reduce vanishing gradient issues and provide improved training convergence. The residual blocks represent the core of the network where the geometrical information is extracted. Each residual block is composed by three graph-convolutional layers, each followed by batch normalization and an activation function. In addition to the feature vectors of the points, the graph-convolutional layer also requires as input a graph, describing connections between points. The graph is computed at the beginning of each residual block as a nearest neighbor graph using Euclidean distances between the feature vectors of the points. Since the graph is updated at every residual block, it is a dynamic graph construction. Such dynamic construction has been seen to promote more powerful feature representations as well as exploiting self-similar patterns [14, 15]. We remark that the graph is shared by all layers inside each residual block to limit computational complexity. Finally, after the two residual blocks, a single-point convolutional layer projects the features of the estimated normals to the 3D space. Notice that, in contrast with other methods such as PCPNet, which estimates only the normal in the central point of the patch, the proposed network estimates a normal for each point in the patch. However, estimates for points at the edges of the patch may suffer from highly skewed neighborhoods.

3.2. Graph-convolutional layer

The graph-convolutional layer is the main building block of the proposed network. Graph convolution generalizes the convolutional operation to data that lie on irregular domains. In the last years many definitions of graph convolution have been proposed. In this paper, we employ a lightweight version of the Edge-Conditioned Convolution (ECC) [13]. The ECC is defined as a weighted aggregation of the node features

Table 1. Unoriented RMS angle error (degrees).

	Proposed	Nesti-Net	PCPNet	PCA	Jet	HoughCNN
Noiseless	**6.47**	6.99	8.49	8.31	7.60	10.02
Low Noise	10.73	**10.11**	11.08	12.00	12.36	11.21
Med Noise	**17.53**	17.63	18.26	18.38	18.33	22.66
High Noise	**22.09**	22.28	22.80	23.50	23.41	33.39

Table 2. Angle error percentiles.

	90 percentile			95 percentile			99 percentile		
	Proposed	Nesti-Net	PCPNet	**Proposed**	Nesti-Net	PCPNet	**Proposed**	Nesti-Net	PCPNet
Noiseless	**8.50**	10.07	12.35	**12.43**	15.13	17.02	**24.04**	25.86	29.66
Low noise	15.61	**15.39**	17.28	22.30	**21.46**	22.91	38.34	**34.73**	36.51
Med noise	27.17	**26.84**	28.67	**35.37**	35.42	37.19	**56.28**	58.92	57.08
High noise	**35.16**	35.17	36.90	**45.57**	45.59	47.33	**65.33**	67.66	66.83

restricted to a neighborhood. We use the lightweight ECC presented in [22], which introduces some approximations in order to reduce the computation complexity and alleviate the risk of vanishing gradient.

The graph-convolutional layer takes as input the feature vectors associated to each point and the graph structure, which describes the connections between the points of the patch. The output feature vector $\mathbf{h}_i^{l+1} \in \mathbb{R}^F$ of point i at layer l is obtained as a weighted aggregation performed over its neighborhood $\mathcal{N}_i^l$:

$$\mathbf{h}_i^{l+1} = \mathbf{W}^l \mathbf{h}_i^l + \sum_{j \in \mathcal{N}_i^l} \gamma^{l,j \to i} \frac{\sum_{s=1}^{r} \kappa_s^{j \to i} \boldsymbol{\theta}_s^{j \to i,L} \boldsymbol{\theta}_s^{j \to i,R^T} \mathbf{h}_j^l}{|\mathcal{N}_i^l|},$$

where $\mathbf{W}^l \in \mathbb{R}^{F \times F}$ is a trainable matrix representing the self-loop contribution, and $\boldsymbol{\theta}_s^{j \to i,R}, \boldsymbol{\theta}_s^{j \to i,L} \in \mathbb{R}^F$ and $\kappa_s^{j \to i} \in \mathbb{R}$ are the outputs of a fully-connected network $\mathcal{F}^l$ whose input is the difference between the feature vectors of point i and point j, i.e.,

$$\boldsymbol{\theta}_s^{j \to i,R}, \boldsymbol{\theta}_s^{j \to i,L}, \kappa_s^{j \to i} = \mathcal{F}^l \left(\mathbf{h}_i^l - \mathbf{h}_j^l \right).$$

The network $\mathcal{F}^l$ is defined as a two-layer multi-layer perceptron, where the second layer is composed of multiple stacked partial circulant matrices in order to reduce the number of parameters. The value r is an hyperparameter that defines the maximum rank of the aggregation weight matrix $\sum_{s=1}^{r} \kappa_s^{j \to i} \boldsymbol{\theta}_s^{j \to i,L} \boldsymbol{\theta}_s^{j \to i,R^T}$. The scalar weight $\gamma^{l,j \to i} \in \mathbb{R}$ is an edge attention term defined as

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

where $\delta \in \mathbb{R}$ is a decay parameter. This term is inspired by the graph attention networks proposed in [23] and it helps to stabilize the network by penalizing the edges that connect nodes with very distant feature representations.

3.3. Loss Function

During training, we consider as loss function the Euclidean distance between the predicted normal vectors $\hat{\mathbf{n}}$ and the ground truth ones $\mathbf{n}$:

$$L = \frac{1}{P} \sum_{i \in \mathcal{S}_P} \min \left(\|\hat{\mathbf{n}}_i - \mathbf{n}_i\|_2^2, \|\hat{\mathbf{n}}_i + \mathbf{n}_i\|_2^2 \right), \qquad (1)$$

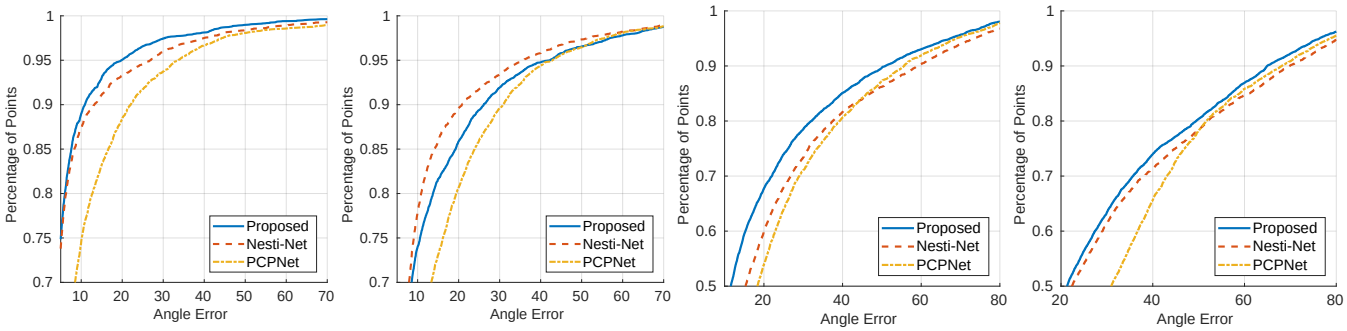
where $\mathcal{S}_P$ is the set containing the P closest nodes to the central point of the patch. This is due to the fact that the nodes far from the center of the patch can suffer from border effects, due to highly skewed receptive fields. Therefore, even if the proposed method estimates the normals of all the points of the patch, we consider only the P nodes closest to the patch center. At the same time, we use more points than just the central point, in contrast with the approach by PCPNet, in order to improve training efficiency and convergence. Moreover, since the main goal of the proposed method is to estimate the unoriented normals associated to the points, the minimum function in Eq. (1) selects for each point the normal orientation that provides the minimum error.

4. EXPERIMENTAL RESULTS

In this section we present a set of experiments aimed at evaluating the performance of the proposed method with respect to model-based baselines and state-of-the-art deep learning approaches. In particular, we consider PCA [5], jet fitting [6], HoughCNN [20], PCPNet [17] and Nesti-Net [18].

4.1. Training and testing details

The training and testing datasets are the same used by PCPNet [17] and Nesti-Net [18] in order to ensure a fair comparison. The training data are composed by 8 point clouds with 100000



(a) Noiseless (b) Low noise (c) Medium noise (d) High noise

Fig. 2. Cumulative distribution of angle errors for point cloud "star sharp".

points partitioned into 10000 patches of $N = 800$ points each. During training, a batch of 16 patches is provided as input to the network and a normal vector is estimated for each of the points in the patch. However, as explained in the previous section, only the normals of the $P = 250$ points closest to the central point of the patch are considered in the loss function. A variance-specific model is independently trained for both our model and PCPNet. White Gaussian noise at three noise standard deviations is considered to test the robustness of the proposed method. The same standard deviation levels used in earlier works are adopted, i.e. 0.012, 0.006, 0.00125 with respect to the bounding box. We remark that the graph construction in the proposed method uses 15 nearest neighbors at noiseless and low noise levels and 35 nearest neighbors at medium-high noise levels. The number of neighbors has been cross-validated on a validation dataset but not optimized for the specific standard deviation; rather it has been discretized into two configurations: one for low noise and one for high noise. The edge attention hyperparameter is set to $\delta = 10$. Leaky ReLUs are used as activation functions. The proposed network is trained for approximately 100 epochs with an initial learning rate equal to 10^{-4}, then decreased to 10^{-5} after 60 epochs.

The testing data are 19 point clouds with 100000 points. Following the protocol of earlier works, a subset of 5000 points per point cloud is chosen for evaluation of the error metric. For each of these points a patch composed of the nearest 800 points is provided as input to the network.

4.2. Quantitative results

In order to quantitatively assess the performance of the proposed method, we measure the root mean squared (RMS) angle error for unoriented normal estimation on the 5000 points per point cloud in the test set. The RMS angle error (in radians) for unoriented estimation is defined as

$$E = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\arccos\left(1-\frac{1}{2}\min(\|\hat{\mathbf{n}}_i-\mathbf{n}_i\|_2^2, \|\hat{\mathbf{n}}_i+\mathbf{n}_i\|_2^2)\right)\right]^2},$$

being $\mathbf{n}_i$ the ground-truth normal vector at point i, $\hat{\mathbf{n}}_i$ the corresponding estimated normal vector and N the number of points.

Table 1 shows the RMS angle error achieved by the various methods. For methods having multi-scale variants, the scale achieving the best result is selected for each standard deviation. It can be noticed that the proposed method is close to or improves state-of-the-art results, achieving lower average errors. In order to gain a deeper insight into the distribution of errors, Table 2 shows a few percentiles of the angle error distribution, averaged over the whole test set with a one-standard-deviation confidence interval. For instance, the value of 24.04 degrees for the 99 percentile means that 99% of the points have a normal estimation error lower than 24.04 degrees. The angle error (in radians) at point i is computed as

$$E_i = \arccos\left(1-\frac{1}{2}\min(\|\hat{\mathbf{n}}_i-\mathbf{n}_i\|_2^2, \|\hat{\mathbf{n}}_i+\mathbf{n}_i\|_2^2)\right).$$

It can be noticed that the proposed method achieves lower values, meaning that it has a shorter tail of the error distribution and consequently fewer points exhibiting high normal estimation errors. We argue that having a lower proportion of points with high estimation errors is even more desirable than lower average error as outliers with high errors can significantly affect the performance of algorithms relying on high quality normal estimation. Fig. 2 shows the high-error tail of the cumulative error distribution for point cloud "star sharp" for the proposed method, Nesti-Net and PCPNet.

4.3. Qualitative results

Fig. 3 visually shows the per-point angle error of the estimated normals with respect to the ground truth for the proposed method, Nesti-Net, PCPNet and PCA at all noise standard deviations. It can be noticed that the proposed method shows a lower proportion of points with high estimation errors. It is also noticeable how the proposed method achieves lower errors in the challenging region constituted by the central junction of the star shape. This is an area where PCPNet in particular suffers from higher errors.

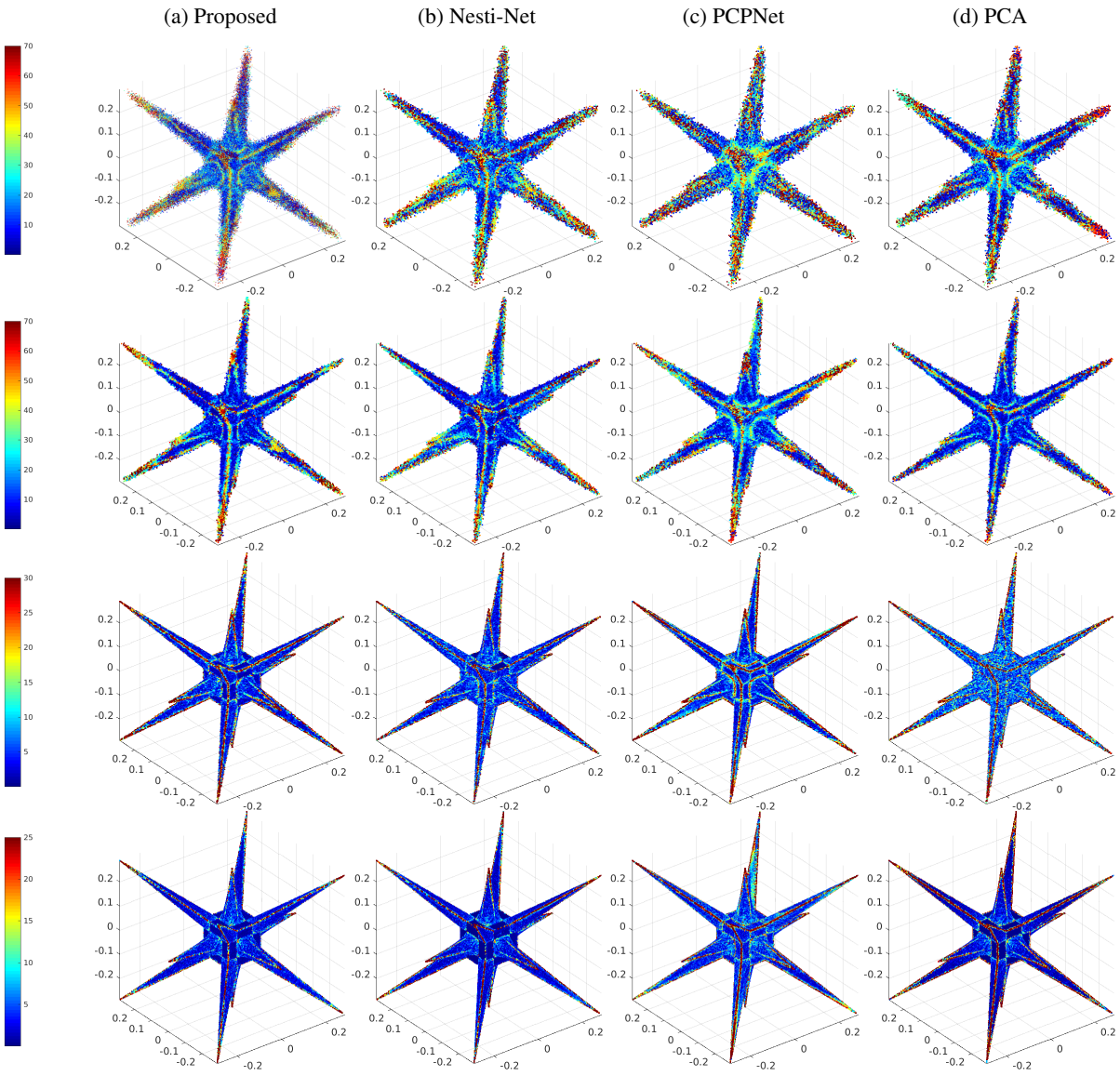


Fig. 3. Angle errors for point cloud "star sharp" at different level of noise: from a high level of noise (top) to noiseless (bottom).

5. CONCLUSIONS

We proposed a method to estimate unoriented surface normals from point clouds using a graph-convolutional neural network. Improvements over state-of-the-art techniques show that the method can achieve robust estimation even in presence of noise.

6. REFERENCES

[1] Henri Gouraud, "Continuous shading of curved surfaces," *IEEE transactions on computers*, vol. 100, no. 6, pp. 623–629, 1971.

[2] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva, "A survey of surface reconstruction from point clouds," in *Computer Graphics Forum*. Wiley Online Library, 2017, vol. 36, pp. 301–329.

[3] Alexander JB Trevor, Suat Gedikli, Radu B Rusu, and Henrik I Christensen, "Efficient organized point cloud segmentation with connected components," *Semantic Perception Mapping and Exploration (SPME)*, 2013.

[4] Chinthaka Dinesh, Gene Cheung, and Ivan V Bajic, "3D Point Cloud Denoising via Bipartite Graph Approximation and Reweighted Graph Laplacian," *arXiv preprint arXiv:1812.07711*, 2018.

[5] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, *Surface reconstruction from unorganized points*, vol. 26, ACM, 1992.

[6] Frédéric Cazals and Marc Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Computer Aided Geometric Design*, vol. 22, no. 2, pp. 121–146, 2005.

[7] David Levin, "The approximation power of moving least-squares," *Mathematics of computation*, vol. 67, no. 224, pp. 1517–1531, 1998.

[8] Nina Amenta and Marshall Bern, "Surface reconstruction by Voronoi filtering," *Discrete & Computational Geometry*, vol. 22, no. 4, pp. 481–504, 1999.

[9] Quentin Mérigot, Maks Ovsjanikov, and Leonidas J Guibas, "Voronoi-based curvature and feature estimation from point clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 743–756, 2010.

[10] Gaël Guennebaud and Markus Gross, "Algebraic point set surfaces," in *ACM Transactions on Graphics (TOG)*. ACM, 2007, vol. 26, p. 23.

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

[12] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst, "Geometric deep learning: going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[13] Martin Simonovsky and Nikos 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.

[14] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, pp. 146, 2019.

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

[16] Tamal K Dey and Samrat Goswami, "Provable surface reconstruction from noisy samples," *Computational Geometry*, vol. 35, no. 1-2, pp. 124–141, 2006.

[17] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra, "PCPNet learning local shape properties from raw point clouds," in *Computer Graphics Forum*. Wiley Online Library, 2018, vol. 37, pp. 75–85.

[18] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer, "Nesti-Net: Normal Estimation for Unstructured 3D Point Clouds using Convolutional Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10112–10120.

[19] Janghun Hyeon, Weonsuk Lee, Joo Hyung Kim, and Nakju Doh, "NormNet: Point-wise normal estimation network for three-dimensional point cloud data," *International Journal of Advanced Robotic Systems*, vol. 16, no. 4, pp. 1729881419857532, 2019.

[20] Alexandre Boulch and Renaud Marlet, "Deep learning for robust normal estimation in unstructured point clouds," in *Computer Graphics Forum*. Wiley Online Library, 2016, vol. 35, pp. 281–290.

[21] Alexandre Boulch and Renaud Marlet, "Fast and robust normal estimation for point clouds with sharp features," in *Computer graphics forum*. Wiley Online Library, 2012, vol. 31, pp. 1765–1774.

[22] Diego Valsesia, Giulia Fracastoro, and Enrico Magli, "Deep Graph-Convolutional Image Denoising," *arXiv preprint arXiv:1907.08448*, 2019.

[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.