A Comprehensive Survey on Convolutional Neural Networks for Learning from Point Cloud

Jagyan Prasad Mahapatro[1] and Ramy Battrawy[2]

[1] yagya.mhp@gmail.com
[2] ramy.battrawy@dfki.de

Abstract. This report presents different approaches of applying convolutional neural networks to learn from Point Clouds. As point clouds are geometric representations of space, learning from point clouds would provide insight into the objects in the scene. Applying CNNs on images can learn robust features in ways that handcrafted features have not been able to. Similarly, feeding point clouds through various CNNs can learn robust geometric features as well. State-of-the-art approaches have been presented to apply CNN on point clouds. In this survey, we present a few of them to show the effectiveness of CNN-based solutions from point clouds for various tasks such as Object Detection, Semantic Segmentation, Scene Flow Estimation, Registration, Motion Forecasting, etc. We compare the results of these approaches which use different data sets to benchmark their results.

1 Introduction

By increasing the accuracy of 3D sensors such as RGB-D, Time-of-Flight and LiDARs, 3D information is showing effectiveness in many applications such as 3D localization, robot navigation and autonomous driving systems. Point clouds are the simplest representation of 3D shapes and scenes which can be generated directly by LiDARs or indirectly from RGB-D or Time-of-Flight sensors by using the calibration parameters of the camera. Point clouds provide a rich geometric structure and they prove to be advantageous where only visual information is not enough to extract feature information. Hence, the underlying structure of the points provides a better insight into the objects of the scene which can be helpful in many different applications such as autonomous driving, indoor navigation, robotics, etc.

Deep learning and specifically convolutional neural networks (CNNs) show incredible success in image processing and analysis. This motivated experts and researchers to try methods for generalizing deep learning to 3D scenes and shapes. However, point clouds possess certain features which make it difficult to apply traditional convolution networks because they are not ordered, unstructured and very sparse compared to RGB images. Thus, some approaches apply CNN on point clouds after converting them into a fixed structure such as voxel grids. However, this voxelized representation can reduce the geometrical information of 3D data. Therefore, this survey shows different approaches which can consume point clouds in convolutional networks. The results of these approaches are compared with the state-of-the-art in the evaluation section.

2 Approaches

This survey covers different approaches to apply convolutional neural networks on point clouds to perform different tasks. We shall look at five different approaches to perform tasks such as Detection, Classification, Segmentation, etc. After that we shall look at an approach for point cloud registration and another for object tracking and motion forecasting.

2.1 Dynamic Graph CNN for learning from point clouds

DGCNN [19] develops a model for learning from point clouds and shows its performance on 3D recognition tasks. The primary idea is that local features are important for point clouds even after introducing deep learning approaches.

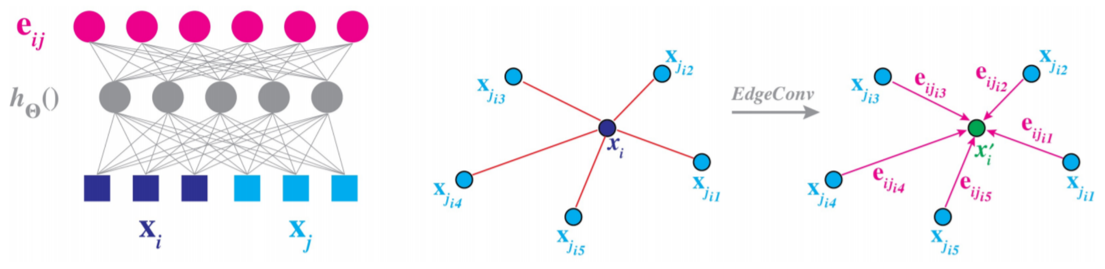

Fig. 1: Left: Computing an edge feature. Right: The EdgeConv operation. The output of EdgeConv is calculated by aggregating the edge features associated with all the edges emanating from each connected vertex [19].

The input of the network is a set of n points with m features. Although usually point clouds are represented just as the x, y and z coordinate, this model can also accept more features for points such as color for the points. The model is inspired by PointNet [13] and convolution operations. Instead of working on the points individually as a separate data point, the approach attempts to exploit the local geometric structures by constructing local neighborhood as a graph representation and applying convolutions on the edges which connect the points. This is similar to applying convolution on patches of pixels in images. The convolution operation in this model is known as EdgeConv as shown in Fig. 1. This model is different from other graph CNNs because the graphs are dynamically updated after each layer of the network thus qualifying the neighbourhood information every time to make it more accurate after each iteration.

The model architectures consists of two branches of segmentation and classification. The classification model has n points as input, after which an edge set of k features are calculated at each EdgeConv Layer, and then the features are aggregated within each set to compute EdgeConv responses for corresponding points. Finally, the output features of the last EdgeConv layer are aggregated to assign class to the whole set. The segmentation model extends the classification model by concatenating the 1D global descriptor and all the EdgeConv outputs (serving as local descriptors) for each point. The output of this model is class value for each point instead of the whole point.

2.2 SPLATNet: Sparse Lattice Networks for Point cloud processing

SPLATNet [17] proposes a generic and flexible neural network to apply CNN on point clouds without transforming point cloud representation to either 2D images or 3D voxel grids. It uses Bilateral Convolution Layers (BCLs) [18] as the building blocks for the network. The paper presents the idea of combining point-based and image-based representations in a network and training it in end-to-end fashion. We shall now look at the structure and the functionality of BCLs.

Bilateral Convolutional Layers accept n input points and f features of each point and then these points are processed in the following steps:

1. Splat: Projects the input features f onto the d dimensional lattice defined by the lattice features.
2. Convolve: Performs d dimensional convolution on the splatted signal with learnable filter kernels.
3. Slice: Filtered signal is then mapped back to the input points.

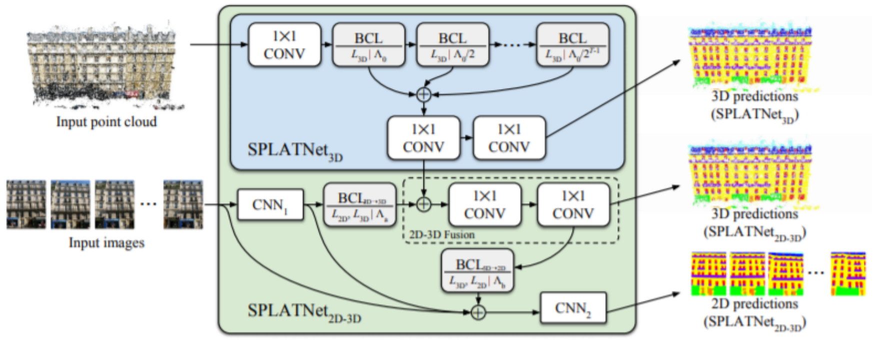
Input point cloud
1×1 CONV
BCL
SPLATNet3D
3D predictions (SPLATNet3D)
Input images
CNN1
2D-3D Fusion
3D predictions (SPLATNet2D-3D)
CNN2
2D predictions (SPLATNet2D-3D)
SPLATNet2D-3D

Fig. 2: Illustration of inputs, outputs and network architectures for SPLATNet$_{3D}$ and SPLATNet$_{2D-3D}$ [17].

Two variations of the network are presented as below:

1. **SPLATNet$_{3D}$**: Input is a 3D point cloud with n points and d $\geq$ 3 input features. Then 1×1 convolution is performed and followed by BCLs. The responses of the BCLs are concatenated and passed through two additional CONV layers as shown in Fig. 2. Finally, a softmax layer produces point-wise class label probabilities. Although the model is mainly experimented with XYZ lattices in this work, BCL allows for other lattice spaces such as position and colour space (XYZRGB) or normal space.

2. **Joint 2D-3D Processing with SPLATNet$_{2D-3D}$**: Given multi-view 2D images are processed using a 2D segmentation CNN. The network then passes it through a BCL and concatenates the output of SPLATNet$_{3D}$ before the last convolution and add two more layers of convolution for classifying points in 3D predictions. For the 2D predictions, the network concatenates the output of the 2D convolutions, reverses BCL on the output of the 1×1 convolution and then performs another 2D convolution for the 2D predictions as shown in Fig. 2. The joint processing by this way can result in better predictions for both 2D images and 3D point clouds.

SPLATNet$_{3D}$ achieves better IoU over all point-based approaches and SPLATNet$_{2D-3D}$ outperforms the state-of-the-art in semantic segmentation. It also boosts the performance of multi-view image labelling.

2.3 PointConv: Deep Convolutional Networks on 3D Point Clouds

PointConv [20] presents an approach to perform convolutions on 3D point clouds with non-uniform sampling. PointConv is a density re-weighted convolution which can fully approximate the 3D convolution on any set of 3D points. The input is n points with d dimensions, including xyz, color, normals, etc.

In the network, the points are sampled around a point as shown in Fig. 3(a). Then they are passed into an MLP to calculate the weights. For each input point, the weights for the MLP are computed using its relative coordinate being passed through a Kernel Density Estimate (KDE) [20]. KDE is a non-parametric method of estimating the probability density function (PDF) [20] of a continuous random variable. Then, element-wise data is aggregated using summation and product. As the structure of the network is hierarchical, an interpolation is used to propagate coarse features from previous layers. Point deconvolution is then used until the input points have been propagated back to the original resolution as shown in Fig. 3 (b).

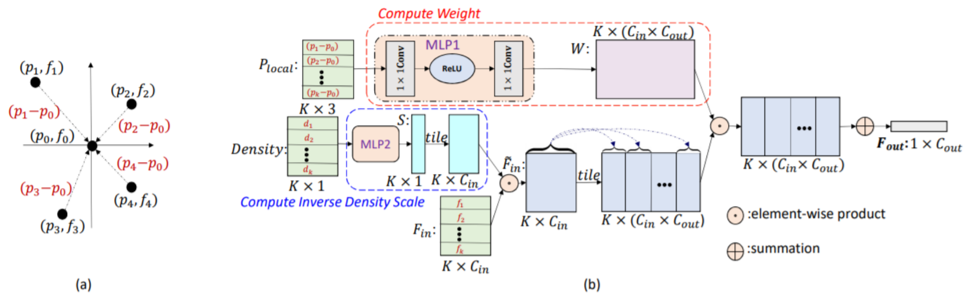

Fig. 3: PointConv [20]. (a) A local region with the coordinates of points transformed from global into local coordinates, p is the coordinates of points, and f is the corresponding feature; (b) The process of conducting PointConv on one local region centered around one point (p_0, f_0). The input features come from the K nearest neighbors centered at (p_0, f_0), and the output feature is F_{out} at p_0.

2.4 Point Convolutional Neural Networks by Extension Operators

PCNN [1] is a methodology to define a way of applying convolutions on point clouds by using extension and restriction operators. Extension operator extends point clouds to a continuous volumetric function over the ambient space. Radial Basis Function (RBF) [3] is chosen as the Extension Operator because it is invariant to point order. A radial basis function (RBF) is a real-valued function whose value depends only on the distance between the input and some fixed points, such as origin or some other points. In addition, different sampling techniques extend to the same volumetric representation. Then a continuous volumetric convolution is applied over the ambient space. Finally, the restriction operator does the inverse of the extension operator by restricting the volumetric representation to the point cloud. The representation of this can be seen in Fig. 4.

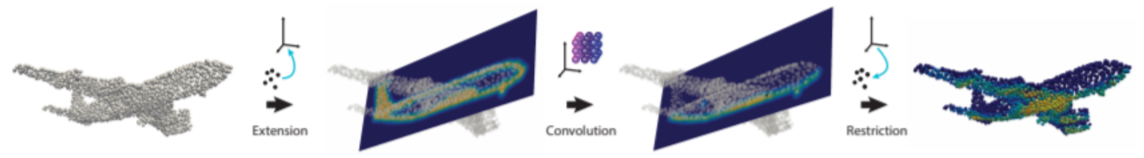

Fig. 4: PCNN [1]: First, a function over the point cloud (in this case the constant one) is extended to a continuous volumetric function over the ambient space; second, a continuous volumetric convolution is applied to this function (without any discretization or approximation); and lastly, the result is restricted back to the point cloud.

The target of this paper is to achieve efficiency, invariance to the order of points, robust to sampling techniques and translation invariant which is achieved through same properties of RBF. PCNN [1] shows imperative results in classification, segmentation and normal estimation.

2.5 3D Point Capsule Networks

3DPointCapsNet [23] presents a network based on a well-known network 2D capsule networks [15]. It is an auto-encoder for learning from unstructured 3D data, i.e., point clouds. The architecture is detailed as the following:

1. **Encoder**: The input is N=2048 points and d=3 features, i.e., the x,y and z coordinate. These points are put through a point-wise MLP, like PointNet [13] for retrieving individual local features into feature maps. These feature maps are then passed through multiple independent convolution layers which are then combined using max-pooling to obtain global latent representation. The representations are then concatenated to a set of vectors called local primary point capsules. Finally, dynamic routing algorithm is used to embed these local capsules to higher level latent capsules.
2. **Decoder**: The latent capsules from encoder are treated as feature map and MLP is used to reconstruct a patch of points. Each capsule is then replicated m times. Each replication is then appended to a unique randomly synthesized grid to a local area [23]. Then these are passed though a final MLP and the patches formed are glued together to form the final image.

2.6 DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration

DeepVCP [10] presents a method to align two different point clouds using deep learning. It achieves comparable registration accuracy and better runtime efficiency to the state-of-the-art geometry-based methods such as ICP [2] and G-ICP [16]. It also shows higher robustness to inaccuracy in initial poses. The method consists of the following components:

1. **Deep Feature Extraction**: Both the source and target point clouds are passed through PointNet++ [14] to extract feature descriptors. Then, those descriptors are selected by a method which have a certain semantic meaning and dynamic objects are avoided by the following section.
2. **Point Weighting**: Here, higher weight is assigned to distinct and invariant static objects, hence ensuring dynamic objects being assigned lower weight and hence not selected as the representative points for the point cloud.
3. **Deep Feature Embedding**: This module takes the selected feature descriptors combined with the input features as input and outputs a 32-dimensional vector of the same length.
4. **Corresponding Point Generation**: Here comes the main feature of the network, i.e., generation of corresponding target points by accepting the feature descriptors which are the centers of voxels formed around the points extracted in the previous step. Then, 3D CNN is used to learn from the similarity distance metric between the source and the target features. A bidirectional matching is also conducted during inference to improve registration accuracy.

2.7 Fast and Furious: Real time End-to-End 3D Detection, Tracking and Motion forecasting with a Single Convolutional Net

Fast and Furious [12] presents a fully convolution based approach that performs simultaneous 3D detection, tracking and motion forecasting. It exploits the spatio-temporal information captured by a 3D sensor which produces point clouds such as LiDAR. Fast and Furious Network performs 3D convolutions across space and time over a bird's eye view representation of the 3D world in a very memory and computationally efficient way. The primary idea is that tracking and predicting can help object detection and false positives can be reduced as well as occluded objects can also be detected by considering evidence across time. As all of these tasks are done simultaneously, the network is very fast. This makes the model perfectly suitable for autonomous driving task.

The input of this model is 4D tensors which encompass XYZ information and temporal information. To apply convolution, the 3D space is represented as a voxel grid and it is filled by checking if a voxel contains a point. As point clouds are mostly sparse, instead of applying 3D convolutions, 2D convolutions are applied using height as the number of channels. As the model works with a voxelized representation, it would also work for denser point clouds. To avoid losing complex geometric features, the voxel size can be reduced. Temporal information is then added into it by taking past n frames and then representing them in the current vehicle coordinate system. Then, multiple frames are appended along a new temporal dimension. The temporal information can be fused either by using early fusion or late fusion. Early fusion fuses the temporal information in the very first layer which will be fast but loses the complex temporal features. Late fusion fuses the temporal data gradually which will be slower in comparison but can capture high level motion features. Finally, tracklets are formed around detected object and as we have past frames, occluded objects will be detected in past frames and new objects will appear in the current frame.

3 Evaluation

The aforementioned methods are compared to various approaches. We will show in this section some results based on their applications.

3.1 Classification

To compare the classification, three models use ModelNet40 [21] as the benchmark for classification. The results are represented in Table 1 which combines the results of the models. The other 2 models, i.e., SPLATNet [17] and 3DPointCapsNet [23] also perform classification but as they are tested on different data sets or in different conditions, their results could only be compared using Shapenet [4].

3.2 Segmentation

Segmentation is one of the tasks which is used to evaluate the models. Four models use ShapeNet data set [4] for part segmentation. These comparative results could be seen in Table 2. The 3DPointCapsNet [23] can not be compared to the remaining models because even if they use the ShapeNet data set, their study is not on the same conditions.

Table 1: Classification results on ModelNet40 [21].

Method	Input	Accuracy(%)
3DShapeNet [21]	1K Pts	84.7
PointNet [13]	1K Pts	89.2
PointNet++ [14]	1K Pts	90.2
Kd-Network [8]	1K Pts	91.8
DGCNN [19]	**1K Pts**	**92.9**
DGCNN [19]	**2K Pts**	**93.5**
PointConv [20]	**1K Pts + Normal**	**92.5**
PCNN [1]	**1K Pts**	**92.3**

Table 2: ShapeNet [4] Segmentation results.

Method	Input	mIoU(%)
PointNet [13]	2K Pts	83.7
PointNet++ [14]	2K Pts + Normals	85.1
Kd-Network [8]	4K Pts	82.3
DGCNN [19]	**2K Pts**	**85.2**
PointConv [20]	**1K Pts + Normal**	**85.7**
PCNN [1]	**2K Pts**	**85.1**
SPLATNet$_{3D}$ [17]	-	**84.6**
SPLATNet$_{2D-3D}$ [17]	-	**85.4**

3.3 Point Cloud Registration

DeepVCP [10] is evaluated for Point Cloud registration on KITTI [5] and Apollo-SouthBay data set [11] and the results compared to some state of the art methods presented in Table 3. The much lower maximum errors demonstrate good robustness.

Table 3: Comparison using KITTI data set [5] and Apollo-SouthBay data set [11].

Method	KITTI data set				Apollo-SouthBay data set			
	Angular Error		Translation Error		Angular Error		Translation Error	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max
ICP-Po2PI [2]	0.084	1.693	0.065	2.050	0.026	0.543	0.024	4.448
G-ICP [16]	0.067	0.375	0.065	2.045	0.025	0.562	0.014	1.540
3DFeat-Net [22]	0.199	2.428	0.116	4.972	0.076	1.180	0.061	6.492
DeepVCP-Base [10]	0.195	1.700	0.073	0.482	0.135	1.882	0.024	**0.875**
DeepVCP-Duplication [10]	0.164	1.212	0.071	**0.482**	0.056	0.875	0.018	0.932

3.4 3D Detection

There is no publicly available data set which evaluates 3D detection, tracking and motion forecasting. Thus for FaF [12], a very large scale data set was collected to benchmark it. This data set is 2 orders of magnitude bigger than data sets such as KITTI [5]. In Table 4 it can be seen that FaF [12] gives better mIoU than all other models while giving a very short detection time.

Table 4: Detection performance on 114 X 80 metres region, with object having $\geq$ 3 number 3D points.

Model	Intersection over Union					
	0.5	0.6	0.7	0.8	0.9	Time[ms]
SqueezeNet_v1.1 [7]	85.80	81.06	69.97	43.20	3.70	9
SSD [9]	90.23	86.76	77.92	52.39	5.87	23
MobileNet [6]	90.56	87.05	78.39	52.10	5.64	65
Fast and Furious Net [12]	**93.24**	**90.54**	**83.10**	**61.61**	**11.83**	30

4 Conclusion

Point clouds provide a good representation of a 3D scene. Many modern 3D sensors are generating point clouds. Therefore, there is potential in learning from point clouds to perform various tasks such as Object Detection, Classification, Segmentation, Motion Forecasting and more. Moreover, we have seen in the past years deep learning showing high quality results for these tasks on 2D images through Convolutional Neural Networks. In this survey, we saw various approaches that could be used for this task on point clouds and compared results of experiments performed on them.

After looking at the different models, it can be seen that DGCNN [19] gives the best results in Classification and the second best results in segmentation. This method seems to be the most suitable because it can give best result only with point clouds without taking any other input; the way SPLATNet$_{2D-3D}$ [17] does.

Now coming to other tasks, DeepVCP [10] could be used as a good alternative to geometric based method for point cloud registration because of its robustness to initial errors in poses. Fast and Furious [12] gives a pretty fast and accurate way of object detection and motion forecasting. However, it relies on a few features of point cloud such as its sparsity. With advent of better sensors this will not be a factor anymore, reducing the chances of detecting complex features by voxelizing the point cloud. Thus, it would give good results for sparse point clouds but for denser point clouds, there would be a trade off between speed and accuracy.

References

1. Matan Atzmon, Haggai Maron, and Yaron Lipman. “Point Convolutional Neural Networks by Extension Operators”. In *the IEEE conference on computer vision and pattern recognition*, 2018.
2. Paul J Besl and Neil D McKay. “Method for registration of 3-D shapes”. In *Sensor fusion IV: control paradigms and data structures*, 1992.
3. Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. “Reconstruction and representation of 3D objects with radial basis functions”. In *the 28th annual conference on Computer graphics and interactive techniques*, 2001.
4. Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. “Shapenet: An information-rich 3d model repository”. In *the IEEE conference on Graphics and Media*, 2015.
5. Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for autonomous driving? the kitti vision benchmark suite”. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

6. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In *the IEEE conference on computer vision and pattern recognition*, 2017.
7. Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size". In *the IEEE conference on computer vision and pattern recognition*, 2016.
8. Roman Klokov and Victor Lempitsky. "Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models". In *the IEEE conference on computer vision and pattern recognition*, 2017.
9. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single Shot MultiBox Detector". In *the 14th European Conference on Computer Vision*, 2016.
10. Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. "DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration". In *the IEEE International Conference on Computer Vision (ICCV)*, 2019.
11. Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. "L3-net: Towards learning based lidar localization for autonomous driving". In *the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
12. Wenjie Luo, Bin Yang, and Raquel Urtasun. "Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net". In *the IEEE conference on computer vision and pattern recognition*, 2018.
13. Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In *the IEEE conference on computer vision and pattern recognition*, 2016.
14. Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In *the IEEE conference on computer vision and pattern recognition*, 2017.
15. Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. "Dynamic Routing Between Capsules". In *the IEEE conference on computer vision and pattern recognition*, 2017.
16. Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. "Generalized-icp.". In *Robotics: science and systems*, 2009.
17. Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. "SPLATNet: Sparse Lattice Networks for Point Cloud Processing". In *the IEEE conference on computer vision and pattern recognition*, 2018.
18. Carlo Tomasi and Roberto Manduchi. "Bilateral filtering for gray and color images". In *sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, 1998.
19. Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. "Dynamic Graph CNN for Learning on Point Clouds". In *the IEEE conference on computer vision and pattern recognition*, 2018.
20. Wenxuan Wu, Zhongang Qi, and Li Fuxin. "PointConv: Deep Convolutional Networks on 3D Point Clouds". In *the IEEE conference on computer vision and pattern recognition*, 2018.
21. Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. "3d shapenets: A deep representation for volumetric shapes". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.
22. Zi Jian Yew and Gim Hee Lee. "3DFeat-Net: Weakly supervised local 3D features for point cloud registration". In *the European Conference on Computer Vision*, 2018.
23. Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. "3D Point Capsule Networks". In *the IEEE conference on computer vision and pattern recognition*, 2018.