Graphical Models 104 (2019) 101033

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

Deep point-based scene labeling with depth mapping and geometric patch feature encoding

Jun-Xiong Cai [a,1], Tai-Jiang Mu [a,1,*], Yu-Kun Lai [b], Shi-Min Hu [a]

[a] Department of Computer Science and Technology, Tsinghua University, China
[b] School of Computer Science & Informatics, Cardiff University, UK

ARTICLE INFO

Keywords:
Segmentation
Semantics labeling
CNN
Deep learning
Patch
Scene understanding

ABSTRACT

This paper presents a deep CNN approach for point-based semantic scene labeling. This is challenging because 3D point clouds do not have a canonical domain and can have complex geometry and substantial variation of sampling densities. We propose a novel framework where the convolution operator is defined on depth maps around sampled points, which captures characteristics of local surface regions. We introduce Depth Mapping (DM) and Reverse Depth Mapping (RDM) operators to transform between the point domain and the depth map domain. Our depth map based convolution is computationally efficient, robust to scene scales and sampling densities, and can capture rich surface characteristics. We further propose to augment each point with feature encoding of the local geometric patches resulted from multi-method through patch pooling network (PPN). The patch features provide complementary information and are fed into our classification network to achieve semantic segmentation.

1. Introduction

Representing 3D scenes is the basis of many applications, ranging from 3D geometric modeling to semantic understanding and navigation. Unlike 2D images which have a natural regular canonical domain, 3D scenes are often represented using surface-based meshes, point clouds and voxels. The voxel representation has a similar regular topology as images, but due to its third power nature, it is restricted to low resolutions, not sufficient to represent 3D scenes with details. On the other hand, surface meshes and point clouds can represent complex 3D scenes with rich details, but they both have irregular connectivity. Real-world 3D scenes are usually captured through range sensors such as Kinect [1], Matterport cameras [2,3] or LiDAR laser scanners [4]. The resulting data of such scanning technology is usually in the form of unstructured surface point clouds, making it the most popular representation for 3D scenes.

Therefore, point cloud based 3D scene understanding is very important in real-world applications, such as semantic reconstruction [5], autonomous driving [6–8], housekeeping or autonomous navigation robots [9], as well as augmented reality (AR) and virtual reality (VR) applications [10,11]. Compared to 2D image analysis and understanding, 3D point clouds contain much more reliable depth information and geometric features which are insensitive to lighting, and so are essential for mobile robots and autonomous driving applications.

The huge success of convolutional neural networks in image detection and analysis motivates the deep learning attempts on 3D scene data. However, point cloud data of indoor scenes often contain millions of points, more than pixels in typical images for deep learning. Moreover, it also has unstructured point distribution due to non-uniform sampling of 3D space, which makes it difficult for many deep learning methods, especially CNN (convolutional neural networks) based methods. To handle these problems, many existing works [12–15] use the voxel representation which allows the 2D convolution to be extended to the 3D space. Although the voxel domain has similar regular topology as images, the amount of data becomes prohibitively large when the resolution increases even mildly. Therefore, it is only suitable for small-scale 3D models, but clearly insufficient to represent scene level 3D data with acceptable accuracy.

The pioneering work PointNet [16] made the first attempt to directly apply a deep neural network to raw point cloud data. PointNet learns high-level spatial features from grid cells of a certain size. However, its capability of capturing the local context and geometric features is limited. The point-based pooling network is affected by non-uniform distribution of points in the 3D space and does not capture the local neighborhood information fully. More recent point-based neural network architectures [17,18] made various improvements of context learning or large-scale optimization. However, all of these methods require non-convolution operators.

* Corresponding author.
E-mail address: mmmutj@gmail.com (T.-J. Mu).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.gmod.2019.101033
Received 23 February 2019; Accepted 28 May 2019
Available online 30 May 2019

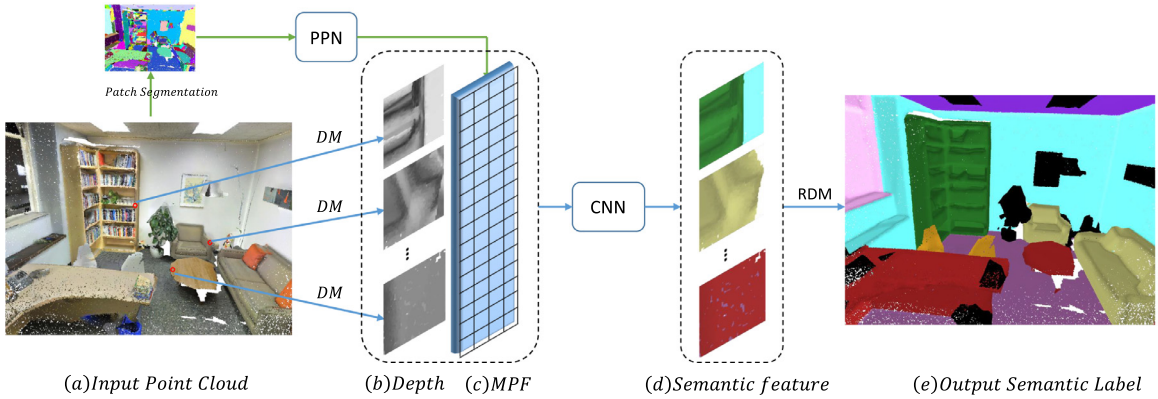


Fig. 1. Our point-based semantic labeling pipeline. Given an input scene represented as an unstructured point cloud (a), our algorithm automatically assigns a semantic label (object category) to each point in the scene. We extract MPF (multi-method patch feature) (c) through training a patch segmentation pooling network (PPN). Our classification network is a Convolutional Neural Network defined in the depth maps of sampled points. To achieve this, we introduce Depth Mapping (DM) and Reverse Depth Mapping (RDM) operators to transform between the point cloud and depth maps. The DM operator transforms the point cloud into depth maps (b) of sampled points, which allows us to apply 2D CNN on (b) and (c) to learn local semantic labels (d). Finally, we compute the semantic label for each point in the scene through the RDM operator (e). Unlike existing methods, our method copes with the entire scene without unnecessary block partitioning, and thus better captures global context.

In this paper, we propose a novel convolutional neural network architecture for point cloud data. We focus on geometric feature extraction and scene level context learning for point cloud data, and learn a network for point-based semantic labeling, i.e. assigning an object class to each point in the input point cloud. Individual points carry rather limited geometric information, so we propose to augment each point with additional information of the local geometric patches it belongs to. However, consistently segmenting point cloud data into patches is not trivial. To address this, we first perform multi-method segmentation to obtain patches through different methods, and design a pooling network to learn reliable patch-based geometric features. Given points with augmented patch features, we further develop a novel CNN architecture for point cloud data. We start by obtaining a set of sampled points using farthest point sampling to provide a good coverage of the point cloud. For each sampled point, we build a 2D depth map to represent its neighborhood, on which 2D convolutions are performed. To achieve this, we introduce DM (Depth Mapping) and Reverse Depth Mapping (RDM) operators which map neighboring points to the depth map and vice versa. This is computationally efficient and captures rich geometric details.

The key contributions of our work are summarized as follows:

- We propose the first deep CNN architecture that can be directly applied to point clouds with a flexible number of points, which is also insensitive to the sampling density, thanks to the fully convolutional architecture.
- We design a multi-method patch segmentation pooling network to learn to extract reliable geometric patch information which complements point information.
- We design two operators: DM and RDM in order to bridge between the point cloud and 2D depth maps around sampled points. Then, we perform 2D convolutions on depth maps to analyze point cloud data, which is both efficient and informative.

An example demonstrating the pipeline of our method for point-based semantic labeling of scenes is shown in Fig. 1. We apply our method to two main large-scale 3D point cloud scene datasets, and show that our method outperforms state-of-the-art methods. We first review the most related work to ours in Section 2. We present our network architecture and algorithm details in Section 3. Experimental results are presented in Section 4 and finally conclusions are drawn in Section 5.

2. Related work

2.1. Obtaining unstructured surface point cloud data

Unstructured point cloud data can be captured using a variety of methods, including: (1) Dense RGB-D SLAM (simultaneous localization and mapping) methods on RGB-D images captured by Kinect-like sensors, such as KinectFusion [19], ElasticFusion [20], BundleFusion [21] etc. For such methods, point cloud data is fused from RGB-D scans. Typical datasets captured in this way include ScanNet [22], SceneNN [23] and Washington RGBD dataset [24]. This approach is of low cost, but the obtained point cloud data is noisier and is more likely to have missing points than other scanning technology, due to the sensor limitations. (2) Using a Matterport sensor capable of capturing panoramic RGB-D images, which are then fused to form the complete point cloud. Typical datasets captured using Matterport includes Full 2D-3D-S Dataset [2] and Stanford 3D Indoor Spaces (S3DIS) [3]. (3) Using a 3D laser scanner, e.g. the dataset in [4]. This approach is more time-consuming, but the obtained point cloud is more accurate than alternative methods. Regardless of the scanning technology, the obtained point clouds can be treated as a set of unstructured 3D points usually along with color information.

2.2. Traditional methods for point cloud semantic labeling

This work addresses semantic labeling of unstructured point cloud data. From the geometric analysis perspective, few methods have been proposed to address this problem. Koppula et al. [25] present a method that performs SVM (support vector machine) classification on pre-segmented patches based on a set of manually specified patch features. Lai et al. [26] learn a hierarchical sparse coding feature called HMP3D for scene labeling trained with CAD model datasets. Vosselman [27] proposes an approach that combines multiple segmentation and post-processing methods for semantic segmentation of urban point cloud scenes with high density. Compared with deep learning methods, such traditional methods have limited learning capabilities, and often need to make some assumptions for the input point cloud data (e.g. quality, density, etc.) to work effectively.

2.3. Deep learning on 3D voxels

To generalize image-based deep learning frameworks to 3D data, a direct approach is to represent 3D shapes as structured volumetric

grids on which 3D convolutions can be applied [12–14]. However, the voxel representation has high memory and computational costs, and can only represent coarse geometric shapes. [15,28] use Octree or KD-Tree structures to help reduce voxel storage and accelerate convolution computation. To deal with point cloud data using the voxel representation, [29] presents a PCNN (point CNN) framework with two operators: extension and restriction, which achieve transformations between point clouds and the voxel representation and perform convolutions on the voxel data for analyzing point cloud data. The work [30] voxelizes a point cloud into 3D voxels and applies 3D FCNN (fully connected neural networks). Even with spatial partitioning techniques, the resolution of voxel data that can be effectively handled is still insufficient to cope with 3D scenes with details, so none of these methods have been applied to scene level point labeling.

2.4. Deep learning on point clouds

Recently, PointNet [16] made a great attempt to use neural networks to learn directly on raw point cloud data. They design a mini-network (T-Net) to work out and apply an appropriate transformation to point coordinates for transformation normalization, and further use max pooling and fully connected layers for feature extraction within certain blocks (typically $1.5\,\mathrm{m} \times 1.5\,\mathrm{m}$ in the $x - y$ plane and over the entire range in the z direction). Tatarchenko et al. [31] present a tangent convolution that learning surface geometry feature from projected virtual tangent image. PointNet++ [17] upgrades PointNet by adding sampling layers and grouping layers for geometric feature computing. Engelmann et al. [18] strengthen point feature learning with sampled neighbors in the 2D Euclidean space and make an attempt to learn larger-scale spatial context above the grid block level. PointCNN [32] introduced an X-transformation for 1-D convolution kernels to make the point set ordered, which makes it possible for multi-level convolutions to learn high-level features. The X-transformation is a trained operator to determine the order of convolution input. However, mapping from a 3D point cloud to an ordered 1D point array can be unreliable for data occlusion and sensitive to local point cloud density. SPG [33] partitions the scanned scene into homogeneous elements and sets up a superpoint graph structure to learn the contextual relationships between object parts. Pan et al. [34] extend CNNs in the regular domains to curved 2D manifolds using parallel frames. This method requires 3D dense mesh data as input which is not usually available for 3D scenes. We propose a new point-based deep learning architecture that augments points with local geometric patch information and a CNN with convolutions defined in 2D depth maps of the neighborhoods of sampled points. These two sources provide complementary information and help improve performance. While the local depth maps provide detailed information of the local geometry, 3D scenes often contain a large number of planar surfaces where local depth maps are not informative. Geometric patch features are particularly useful in such cases. Our method is able to handle the point cloud without the need of partitioning points into blocks of specific sizes and extracts useful geometric information to improve semantic labeling performance.

3. Depth mapping point-based CNN

In this section, we introduce our framework in detail. We propose a CNN approach that takes an unstructured scene point cloud as input, and directly outputs its semantic labeling result (i.e. the object category for each point). Denote by N and M the number of points in the input point cloud and the number of object categories (e.g. tables, chairs, etc.), respectively.

A point cloud is represented as a set of 3D points $\{P_i | i = 1, 2, ., N\}$. Each point P_i is represented using a 10-dimensional vector, i.e. the point descriptor

$$\tilde{\mathbf{P}}_i = (x, y, z, n_x, n_y, n_z, r, g, b, c),$$

where (x, y, z) is the 3D coordinates of the point, (n_x, n_y, n_z) is the point normal estimated using PCA (principal component analysis) of the local neighborhood, (r, g, b) is the color information and c is the curvature calculated using PCL library as the minimum eigenvalue of the local PCA.

Our model is constructed with two main networks. The first one is a patch pooling network (PPN) that learns to extract useful geometric features. For this network, we first perform multi-method patch segmentation and extract patch-based features using different segmentation methods at different scales (Section 3.1). These multi-method patch features are fed into the pooling network along with point features to produce an aggregated feature associated with each point (Section 3.2). The second network is a classification network that takes the 3D point cloud as input and outputs labeling results for points with an embedded CNN for hierarchical feature learning (Section 3.3).

3.1. Patch segmentation and multi-method feature extraction

Since each point contains limited information, we extract local geometric patches to provide richer information, similar to superpixels in image analysis. Patch segmentation is a good way to extract geometric shape features for surface point clouds. There are many methods for point cloud segmentation. Koppula et al. [25] use a traditional region growing method from the PCL library for patch segmentation. Silberman et al. [1] treat RGBD images as point clouds and design a semantic segmentation method to categorize points into 4 limited classes (ground, furniture, prop, structure). Mattausch et al. [4] propose a lightweight region growing method for fused point cloud data that is more sensitive to depth. In general, none of these geometry patch segmentation methods are perfect, due to the existence of noise, occlusion and discontinuity in typical point cloud data.

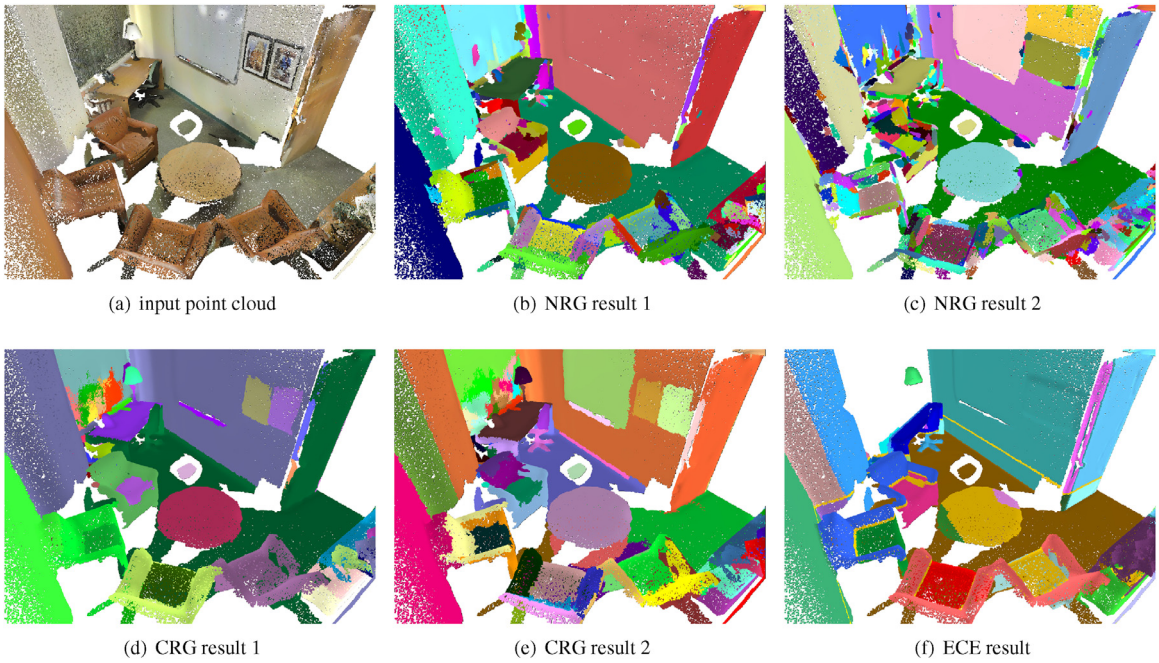
Our semantic labeling is performed at the point level and patch segmentation is only used to provide additional geometric information. Our method is thus not sensitive to the patch segmentation method used. Moreover, each method works better in certain situations than other methods, and most methods have adjustable parameters corresponding the scale (coarser or finer) of segmentation. We therefore take an approach that utilizes multiple segmentation methods at different scales. We consider the following methods: 1) Normal-based Region Growing (NRG) [4], 2) Color-based Region Growing (CRG) from PCL [35], and 3) Euclidean Cluster Extraction (ECE) [36]. Brief introduction of these methods is given below.

3.1.1. Normal-based Region Growing (NRG)

Mattausch et al. [4] grows a patch starting from a seed point $\mathbf{s}$. Given a new point $\mathbf{p}$ belonging to the k-nearest neighbors of an existing point in the current patch, we add $\mathbf{p}$ to the patch if the following conditions are satisfied:

$$\|\mathbf{n_p} \cdot \mathbf{n_s}\| > t_1, \|(\mathbf{p} - \mathbf{s}) \cdot \mathbf{n_s}\| < t_2, \quad (1)$$

which states that $\mathbf{p}$ should have a consistent normal as $\mathbf{s}$ and within a small distance to the tangential plane of $\mathbf{s}$. The segmentation results are highly dependent on the parameters t_1, t_2 that control the planarity of segmentation. In practice, different settings may work better for different objects, such as a board and a sofa with substantial variation of planarity. Inappropriate settings of these parameters may result in over-segmentation or under-segmentation. By altering these parameters, we are able to obtain segmentation results at different scales. An example is shown in Fig. 2(b-c), where two settings are used. Since it is hard or impossible to determine segmentation parameters that work well for different data sources or different object classes in one scene, we perform multi-scale segmentation by using a set of parameters of different scales. In our experiments, the multi-scale segmentation parameters are: $\{t_1 = 0.5, t_2 = 0.04\}, \{t_1 = 0.5, t_2 = 0.04\}, \{t_1 = 0.6, t_2 = 0.03\}, \{t_1 = 0.7, t_2 = 0.02\}, \{t_1 = 0.8, t_2 = 0.01\}, \{t_1 = 0.95, t_2 = 0.002\}$.



(a) input point cloud (b) NRG result 1 (c) NRG result 2

(d) CRG result 1 (e) CRG result 2 (f) ECE result

Fig. 2. Given a point cloud (a), we obtain different segmentation results using different methods with different parameter settings. NRG result 1 (b): $t_1 = 0.5, t_2 = 0.04$, NRG result 2 (c): $t_1 = 0.7, t_2 = 0.02$, CRG result 1 (d): $t_3 = 6, t_4 = 5$, CRG result 2 (e): $t_3 = 4, t_4 = 3$ and ECE result (f): $t_5 = 0.02$.

These parameters do not need tuning because the aim is to provide a good coverage of patches at different scales.

3.1.2. Color-based region growing (CRG) from PCL

Rusu and Cousins [35] uses color instead of normals as the constraint. Moreover, it uses a merging algorithm for over-segmentation and under-segmentation control by making attempts to merge patches with close colors and further merge tiny clusters with closest neighboring patches. In our experiments, we consider three scales and set the point color threshold t_3 (for region growing use) and region color threshold t_4 (for region merging use) as $\{t_3 = 3, t_4 = 3\}, \{t_3 = 6, t_4 = 5\}, \{t_3 = 9, t_4 = 7\}$. An example of segmentation results with CRG at two different scales is shown in Fig. 2(d-e).

3.1.3. Euclidean Cluster Extraction (ECE)

Yan-Xiong et al. [36] segments a point cloud into clusters based on the Euclidean distance. Firstly it designs a plane filter to remove large planes in the scene to allow segmentation of point clusters corresponding to individual objects lying on the large plane (desk, ground, wall etc.). In our experiments, the cluster tolerance distance t_5 is set as $0.01m$ and $0.02m$ for two scales. An example of ECE segmentation result is given in Fig. 2(f).

Each point belongs to patches with different shapes in different segmentation methods and scales. For each patch, we extract a 10-dimensional feature descriptor $\mathbf{F} = \{f_1, f_2, \ldots, f_8\}$ (Note that f_2 is three-dimensional). The detailed definition for patch descriptors is listed in Table 1.

3.2. PPN: patch pooling network

In the previous step, for each point P_i, we extract L patch segmentation results by using different segmentation methods at different scales, leading to patch features $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_L$, along with point descriptor $\tilde{\mathbf{P}}_i$. Not all the segmentation results are meaningful for patches around a given point. Feeding all the information is not only redundant, but can also be misleading by poor segmentation results with specific methods at certain levels.

Table 1
Feature descriptors of a patch. The PCA normal is treated as 3 channels. n in f_8 is the number of points in the patch.

F	Descriptors
f_1	Mean height
f_2	PCA normal of the points in the patch
f_3	Area of the concave hull: a
f_4	width of fitting rectangle: w
f_5	length of fitting rectangle: l
f_6	Ratio between length and width: w/l
f_7	Ratio between fitting rectangle area and area f_3: $w \times l/a$
f_8	Point density: n/a

In order to obtain a concise and informative feature representation, we propose a novel pooling neural network. The detailed network architecture is shown in Fig. 3. We first take all the patch features $\mathbf{F}_j$ $(j = 1, 2, \ldots, L)$ as input to the network. We then perform two stages of MLPs (multi-layer perceptrons) to independently transform patch features at the same level, leading to an $L \times 252$ dimensional feature representation. To extract the most informative features, we propose to use max pooling to reduce the dimension to 252 as aggregated patch feature $\tilde{\mathbf{F}}$. These are further combined with the point descriptor $\tilde{\mathbf{P}}_i$, and fed into another MLP stage to obtain 64-dimensional point/patch feature, denoted as $\hat{\mathbf{P}}_i$. To train this network, we use $\hat{\mathbf{P}}_i$ to classify the point into M categories, with the last layer containing M neurons. In the training process, for each point in the training set, the input to the network includes the point descriptor and patch features, and the network is optimized to predict the label (object category) of the point as an M-dimensional one-hot vector.

For our semantic labeling method, the pooling network is used to obtain point/patch feature $\hat{\mathbf{P}}_i$ for each point P_i. The trained pooling network is also able to predict the label of each point, although only using its local information, without higher-level context. We take the result of PPN as our baseline.

3.3. Classification network using depth mapping CNN

CNN is an outstanding approach for machine learning on structured data. However, scanned scene point cloud data is usually unstructured

Fig. 3. Patch feature pooling network (PPN). L is the number of patch segmentation results. The network processes one point at a time, where the input includes patch features $\mathbf{F}_j$ $(j = 1, 2, \ldots, L)$ and point descriptor $\tilde{\mathbf{P}}_i$. The output is classification scores for M classes, and the point is assigned to the class with the highest score. MLP stands for multi-layer perceptrons, and numbers in brackets are layer sizes. FC is a fully connected layer.

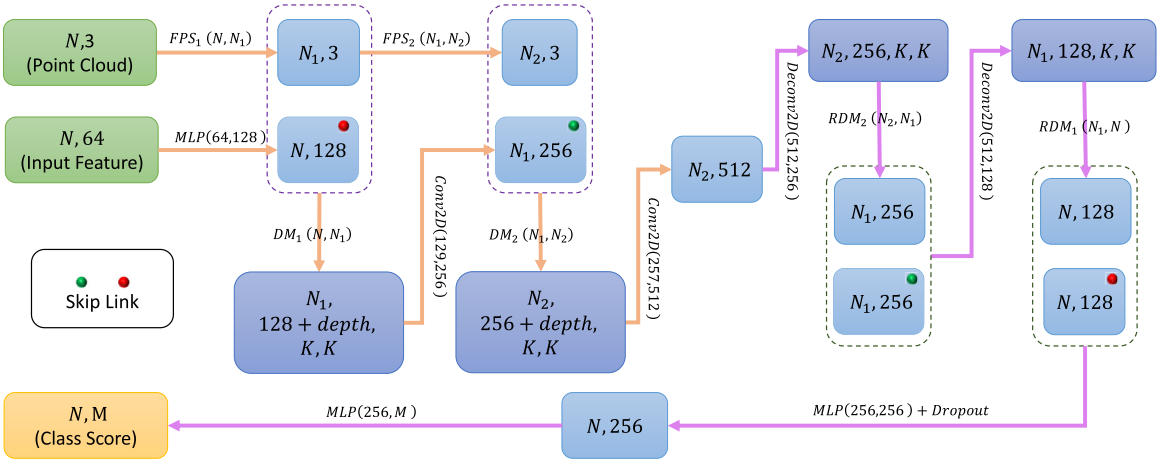


Fig. 4. Our Depth Mapping CNN architecture for point-based semantic labeling. The network takes N points as input. For the i^{th} vertex, the point feature $\hat{\mathbf{P}}_i$ is extracted using PPN in Fig. 3. The output of the network is classification scores of M classes for each of N points. $MLP(\ldots)$ stands for multi-layer perceptrons, where numbers in bracket are layer sizes. $FPS(x, y)$ denotes the farthest point sampling operator, similar to Qi et al. [17], where x and y mean the sizes of point cloud before and after sampling. $DM(x, y)$ refers to the Depth Mapping operator that generates a depth image around each sampled point by projecting neighboring points in the Euclidean space from the upper-layer point cloud to the tangent plane of the sampled point. x and y are the numbers of points and sampled points, respectively. The result is a $K \times K$ ($K = 16$ in our experiments) depth image for each sampled point. The depth information is added to the point features as an additional input channel. $Conv2d(x, y)$ and $DeConv2d(x, y)$ refer to 2D convolution/Deconvolution operators on depth images, where x and y denote the channel sizes of input and output. $RDM(x, y)$ is the reverse mapping that maps the depth images onto upper-layer point cloud. Skip Link layers are added after $DeConv2d$.

on which traditional CNN methods are difficult to apply. Therefore, we design new DM and RDM operators to bridge between CNN and point cloud data. The DM operator maps the point cloud into depth maps on which 2D CNN can be easily applied, and the RDM operator reverses the procedure of DM and remaps the sampled depth image back onto upper layer point cloud with deconvoluted features. Our full classification network architecture is summarized in Fig. 4, where we build a multi-layer convolutional network with point cloud down-sampling and up-sampling.

3.3.1. DM/RDM operators

The DM operator $DM(p, q)$ takes the input point cloud with p points $P_1, P_2, \ldots, P_p$ and a subset of q sampled points $P_{s_1}, P_{s_2}, \ldots, P_{s_q}$ as input and generates a depth image for each sampled point. For each sampled point P_{s_j}, we search the neighboring points with k-nearest neighbor (kNN) and project them onto the tangent plane (orthogonal to the normal of the sampled point) of P_{s_j}. To improve learning, we orient the depth map consistently such that it is orthogonal to the gravity direction of P_{s_j}. We then sample in the tangent plane a depth image of resolution $K \times K$ and add the depth (absolute distance between neighboring points and the tangent plane) as an extra feature channel. Similarly, point features of projected neighboring points are also used to generate 2D feature images. This operator aims at learning rich local structure features for points. $K = 16$ is used in our experiments. Since points are discrete, we interpolate between projected neighboring points to obtain complete depth/feature images. Examples of depth images are shown in Figs. 1 and 5. As can be seen, such depth images are insensitive to sampling densities and provide compact and rich geometric information for learning. After passing through convolutional layers, we introduce the RDM operator $RDM(q, p)$ which takes semantic features from the CNN in depth images as input and outputs the point cloud with semantic label information $P_1, P_2, \ldots, P_p$. The feature channel of P_i is extracted from the depth image associated with P_{s_k} that is nearest to P_i. DM and RDM and inverse operators, both required to set up an end-to-end network structure.

3.3.2. Classification network architecture

The overall classification network that assigns a semantic label to each point in the input point cloud is summarized in Fig. 4. We perform two stages of farthest point sampling, with N_1 and N_2 sampled points respectively. In our experiments, we set $N_1 = N/32$, and $N_2 = N_1/32$. The DM operator is used to generate depth images around sampled points, which are then combined with point features. The regular structure of depth images means 2D convolution and deconvolution operators can be applied to extract useful information. We then apply the RDM operator to map filtered information in the depth domain back to the point cloud. We introduce skip links between dots with the same color to pass detailed information to the later stages of the network, avoiding information loss. The final output is $N \times M$ scores, i.e. for each point, we obtain M scores corresponding to the M object categories, and we simply assign each point to the class with the highest score.

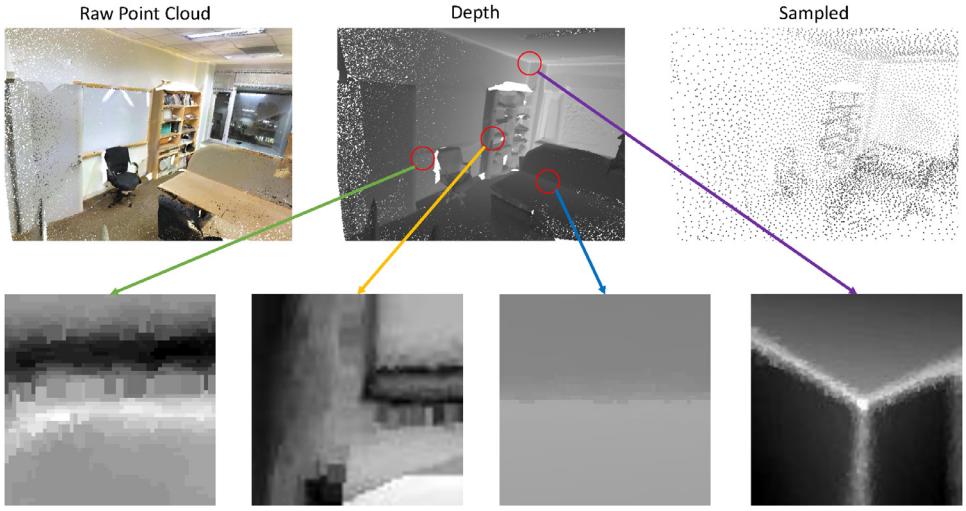


Fig. 5. Examples of depth images generated by the DM operator.

Table 2
Comparison of accuracy and IoU (intersection of union) performance on the S3DIS dataset (per semantic class and overall). OA means overall accuracy, mAcc means mean accuracy across classes and mIoU means mean IoU across classes. Our method outperforms state-of-the-art methods.

Method	OA	mAcc	mIoU	Ceiling	Floor	Wall	Beam	Column	Window	Door	Table	Chair	Sofa	Bookcase	Board	Clutter
PointNet [16]	78.5	66.2	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
SEGCloud [30]	–	57.35	48.9	90.1	96.1	69.9	00.0	18.4	38.4	23.1	**75.9**	70.4	**58.4**	40.9	13.0	41.6
Engelmann [18]	81.1	66.4	49.7	90.3	92.1	67.9	44.7	24.2	52.3	51.2	58.1	47.4	06.9	39.0	30.0	41.9
SPG [33]	85.5	73.0	62.1	89.9	95.1	76.4	62.8	**47.1**	55.3	**68.4**	69.2	**73.5**	45.9	**63.2**	8.7	**52.9**
Ours method	**88.1**	**75.2**	**63.7**	**98.7**	**98.3**	**76.6**	**77.7**	42.6	**63.9**	65.2	66.2	72.3	28.3	45.8	**47.3**	39.2

4. Experiments

We use two main large-scale benchmarks S3DIS and ScanNet for evaluating semantic labeling performance. We train our network on a computer with an Intel i7-8700K CPU, 64GB memory and a GTX 1080Ti GPU with 11GB on-board memory. Cross-entropy loss is optimized during training. When training the PPN, we set the batch size to 256 and set the learning rate to 10^{-4}. For the classification network training, we set the batch size to 1 and the learning rate to 10^{-2}. The training process terminates when the loss converges (i.e. the overall loss does not reduce in three epochs).

4.1. S3DIS dataset

The S3DIS dataset [3] includes 6 areas and 271 rooms captured by Matterport cameras, with ground truth point-level labeling of 13 classes, including chairs, tables, etc. We perform 6-fold cross validation and take the micro-averaged metrics over all 6 folds as our evaluation result. We compare our method with state-of-the-art deep methods [16,18,30,33] where the performance is either reported or code available from the authors. We use the accuracy and Intersection over Union (IoU) measures to evaluate the performance. As shown in Table 2, our method achieves significantly better result than [16,18,30,33] overall and perform best for most classes. Fig. 6 shows some results of PointNet and our method. It can be seen clearly that our method works significantly better for the classes of windows, boards and columns.

4.2. ScanNet dataset

The ScanNet dataset [22] contains 1513 scanned and reconstructed indoor scenes with 21 object classes. We follow the experiment setting in [22] and use 1201 scenes for training, and the remaining 312 scenes for testing.We compare our work with state-of-the-art methods either based on their reported performance or using code provided by authors. The detailed point-level accuracy results are reported in Table 3, which shows that our method performs better than state-of-the-art methods. Our baseline is the result of only using the PPN, which gives decent performance already (comparable to PointNet), showing the effectiveness of the pooling network. Our Deep Mapping network is a light network for local feature training, which aims at directly applying on point cloud data without pre-processing of block segmentation and provides good point structure for convolutional networks. The performance of using it alone is reasonable but still limited. However, when combined this with the PPN, our method achieves better performance.

Table 3
Comparison of per-point classification accuracy between our method and state-of-the-art deep learning methods on the ScanNet dataset. PPN only is the performance of only using the PPN, which is related to the segmentation result while insensitive to sampling density. Depth Mapping CNN Only is the performance of classification network that only uses the position and color as the input channel without the PPN feature.

Method	Accuracy
PointNet [16]	73.9
PointNet++ [17]	84.5
Tangent CNN [31]	80.9
PointCNN [32]	85.1
PPN Only	73.4
Depth Mapping CNN Only	75.2
Our Method	85.3

4.3. Evaluation of multi-method patch segmentation

We evaluate how the number of patch segmentation methods and segmentation levels affects the trained features. Since different segmen-

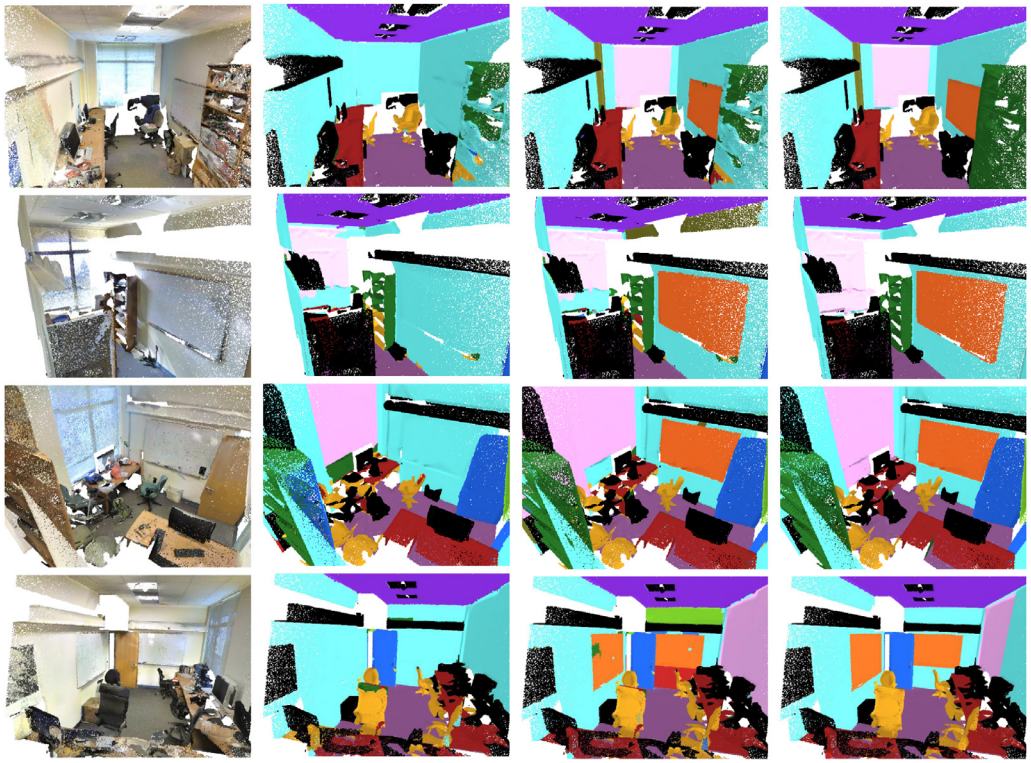
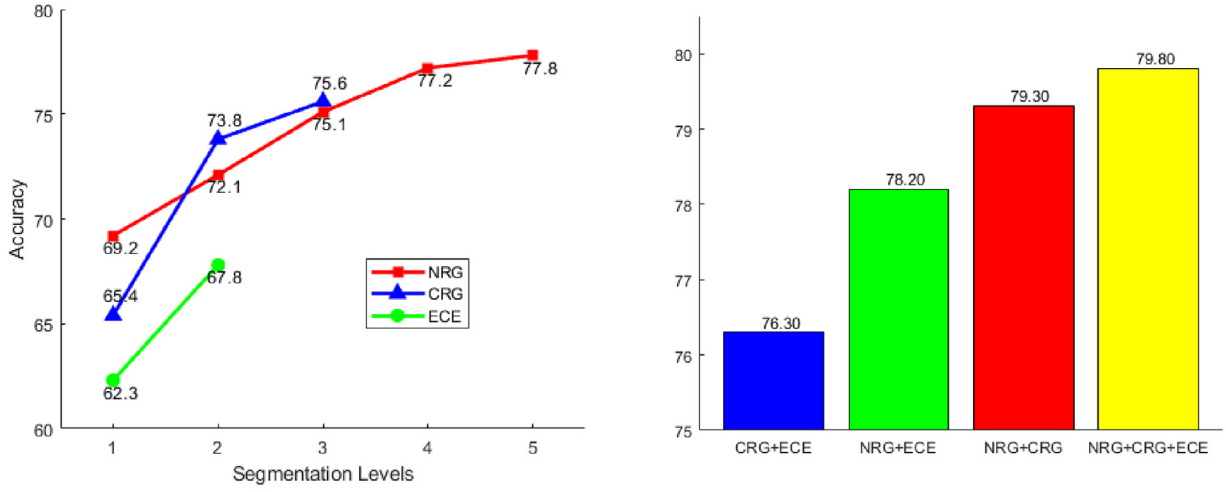


Fig. 6. Comparison of results of PointNet and our method on the S3DIS dataset. For most case PointNet failed to classify the board, window and column due to its limited geometry shape feature. And our method produce less label noise as the consistent points of a single object usually share some segmentation patches in suitable segmentation level.



(a) Multi-level segmentation results (b) Multi-method segmentation results

Fig. 7. The overall accuracy results of our PPN with different segmentation levels (a) and combinations of different methods (b). In (b) for each method, all the levels are used as this gives the best performance, as shown in (a). It is clear from (a) and (b) that the more segmentation levels and methods are used, the more reliable geometric features are extracted, leading to better classification accuracy.

tation results directly affect the pooling network PPN, we evaluate the PPN performance of each patch segmentation method with different levels of segmentation, and it clearly shows that more segmentation levels lead to better accuracy (Fig. 7(a)). We evaluate the effectiveness of the trained features by the overall accuracy (computed based on the class scores of Fig. 3). We further evaluate a combination of different segmentation methods (with all the segmentation levels of each method used, as this gives the best performance). The accuracy is improved with an increasing number of segmentation levels and a combination of more segmentation methods. This demonstrates the effectiveness of PPN, which effectively selects the most suitable patch segmentation and therefore benefits from more possible patch segmentation results to choose from.

4.4. Running times

The average time needed for semantic labeling of a scene in the S3DID dataset with about 0.8 million points is 2 min. This is largely due to the different patch segmentation methods used. Specifically, the

multi-method segmentation takes 33 s, the feature descriptor computation takes 83 s and the deep neural network inferencing takes 6 s.

5. Conclusion and future work

In this paper, we propose a novel approach to semantic labeling of point clouds at the scene level. For each point, we extract patch features with different segmentation methods at different scales by changing the segmentation parameters and design a patch feature pooling network (PPN) for patch feature learning. We further propose a novel classification network that combines these features with two key operators, namely *DM* and *RDM* which produce depth images around sampled points for 2D CNN to be applied to. Our network can directly work on point cloud data without sampling into certain size as most other methods [16–18,32] do. However, the input channels of PPN are hand-crafted feature descriptors. We would like to develop fully automatic learning method to extract the most suitable patch feature. Moreover, our depth mapping CNN architecture is general and can be applied to other point cloud processing tasks using deep learning.

Acknowledgments

This work was supported by the National Key Technology R&D Program (Project Number 2017YFB1002604), the Natural Science Foundation of China (Project Number 61521002, 61761136018) and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. We would like to thank the authors and contributors of the ScanNet dataset [22] and the S3DIS dataset [3] for making the datasets available.

References

[1] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from RGBD images, in: European Conference on Computer Vision, 2012, pp. 746–760.
[2] I. Armeni, A. Sax, A.R. Zamir, S. Savarese, Joint 2D-3D-Semantic Data for Indoor Scene Understanding, 2017 arXiv:1702.01105.
[3] I. Armeni, O. Sener, A.R. Zamir, H. Jiang, I. Brilakis, M. Fischer, S. Savarese, 3D semantic parsing of large-scale indoor spaces, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016.
[4] O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornung, R. Pajarola, Object detection and classification from large-scale cluttered indoor scans, Comput. Graph. Forum 33 (2) (2014) 11–21.
[5] K. Chen, Y.-K. Lai, Y.-X. Wu, R. Martin, S.-M. Hu, Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information, ACM Trans. Graph. 33 (6) (2014) 208:1–12.
[6] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 3354–3361.
[7] R. Gomez-Ojeda, J. Briales, J. Gonzalez-Jimenez, PL-SVO: semi-direct monocular visual odometry by combining points and line segments, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4211–4216.
[8] C. Chen, A. Seff, A. Kornhauser, J. Xiao, DeepDriving: learning affordance for direct perception in autonomous driving, in: IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2722–2730.
[9] S. Wang, H. Zhao, X. Hao, Design of an intelligent housekeeping robot based on IOT, in: International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), IEEE, 2015, pp. 197–200.
[10] M. Serino, K. Cordrey, L. Mclaughlin, R.L. Milanaik, Pokémon go and augmented virtual reality games: a cautionary commentary for parents and pediatricians, Curr. Opin. Pediatr. 28 (5) (2016) 673 .
[11] T. Piumsomboon, A. Clark, M. Billinghurst, A. Cockburn, User-defined gestures for augmented reality, Lect. Notes Comput. Sci. 8118 (2017) 282–299.
[12] D. Maturana, S. Scherer, VoxNet: a 3D convolutional neural network for real-time object recognition, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2015, pp. 922–928.
[13] C.R. Qi, H. Su, M. Niebner, A. Dai, M. Yan, L.J. Guibas, Volumetric and multi-view CNNs for object classification on 3D data, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5648–5656.
[14] Y. Zhou, O. Tuzel, VoxelNet: end-to-end learning for point cloud based 3D object detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[15] P.S. Wang, Y. Liu, Y.X. Guo, C.Y. Sun, X. Tong, O-CNN: octree-based convolutional neural networks for 3D shape analysis, ACM Trans. Graph. 36 (4) (2017) 72.
[16] C.R. Qi, H. Su, K. Mo, L.J. Guibas, PointNet: deep learning on point sets for 3D classification and segmentation, IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) 1 (2) (2017) 4.
[17] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: deep hierarchical feature learning on point sets in a metric space, in: Advances in Neural Information Processing Systems, 2017, pp. 5105–5114.
[18] F. Engelmann, T. Kontogianni, A. Hermans, B. Leibe, Exploring spatial context for 3D semantic segmentation of point clouds, in: IEEE International Conference on Computer Vision Workshop, 2017, pp. 716–724.
[19] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R.A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A.J. Davison, A.W. Fitzgibbon, KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera, in: ACM Symposium on User Interface Software and Technology, 2011, pp. 559–568.
[20] T. Whelan, R.F. Salas-Moreno, B. Glocker, A.J. Davison, S. Leutenegger, Elasticfusion: real-time dense SLAM and light source estimation, Int. J. Rob. Res. 35 (14) (2016).
[21] A. Dai, S. Izadi, C. Theobalt, BundleFusion: real-time globally consistent 3D reconstruction using on-the-fly surface re-integration, ACM Trans. Graph. 36 (4) (2017) 76a.
[22] A. Dai, A.X. Chang, M. Savva, M. Halber, T. Funkhouser, M. Nießner, ScanNet: richly-annotated 3D reconstructions of indoor scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[23] B.-S. Hua, Q.-H. Pham, D.T. Nguyen, M.-K. Tran, L.-F. Yu, S.-K. Yeung, SceneNN: a scene meshes dataset with annotations, in: International Conference on 3D Vision (3DV), 2016.
[24] K. Lai, L. Bo, X. Ren, D. Fox, A large-scale hierarchical multi-view RGB-D object dataset, in: IEEE International Conference on Robotics and Automation, 2011, pp. 1817–1824.
[25] H.S. Koppula, A. Anand, T. Joachims, A. Saxena, Semantic labeling of 3D point clouds for indoor scenes, in: International Conference on Neural Information Processing Systems, 2011, pp. 244–252.
[26] K. Lai, L. Bo, D. Fox, Unsupervised feature learning for 3D scene labeling, in: IEEE International Conference on Robotics and Automation, 2014, pp. 3050–3057.
[27] G. Vosselman, Point cloud segmentation for urban scene classification, ISPRS - Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XL-7/W2 (7) (2013) 257–262.
[28] R. Klokov, V. Lempitsky, Escape from cells: deep kd-networks for the recognition of 3D point cloud models, in: IEEE International Conference on Computer Vision (ICCV), 2017, pp. 863–872.
[29] M. Atzmon, H. Maron, Y. Lipman, Point convolutional neural networks by extension operators, ACM Trans. Graph. 37 (4) (2018) 71.
[30] L.P. Tchapmi, C.B. Choy, I. Armeni, J. Gwak, S. Savarese, SEGCloud: semantic segmentation of 3D point clouds, in: International Conference on 3D Vision (3DV), 2017.
[31] M. Tatarchenko, J. Park, V. Koltun, Q.-Y. Zhou, Tangent convolutions for dense prediction in 3D, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3887–3896.
[32] Y. Li, R. Bu, M. Sun, B. Chen, PointCNN, arXiv:1801.07791 (2018).
[33] L. Landrieu, M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs, 2017 arXiv:1711.09869.
[34] H. Pan, S. Liu, Y. Liu, X. Tong, Convolutional neural networks on 3d surfaces using parallel frames, 2018 arXiv:1808.04952.
[35] R.B. Rusu, S. Cousins, 3D is here: point cloud library (PCL), in: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 2011.
[36] W.U. Yan-Xiong, L.I. Feng, F. Liu, L.N. Cheng, L.L. Guo, Point cloud segmentation using Euclidean cluster extraction algorithm with the smoothness, Meas. Control Technol. (2016).