Multi-Scale Point-Wise Convolutional Neural Networks for 3D Object Segmentation From LiDAR Point Clouds in Large-Scale Environments

Lingfei Ma, *Student Member, IEEE*, Ying Li, Jonathan Li, *Senior Member, IEEE*, Weikai Tan, Yongtao Yu, *Member, IEEE*, and Michael A. Chapman

***Abstract*— Although significant improvement has been achieved in fully autonomous driving and semantic high-definition map (HD) domains, most of the existing 3D point cloud segmentation methods cannot provide high representativeness and remarkable robustness. The principally increasing challenges remain in completely and efficiently extracting high-level 3D point cloud features, specifically in large-scale road environments. This paper provides an end-to-end feature extraction framework for 3D point cloud segmentation by using dynamic point-wise convolutional operations in multiple scales. Compared to existing point cloud segmentation methods that are commonly based on traditional convolutional neural networks (CNNs), our proposed method is less sensitive to data distribution and computational powers. This framework mainly includes four modules. Module I is first designed to construct a revised 3D point-wise convolutional operation. Then, a U-shaped downsampling-upsampling architecture is proposed to leverage both global and local features in multiple scales in Module II. Next, in Module III, high-level local edge features in 3D point neighborhoods are further extracted by using an adaptive graph convolutional neural network based on the K-Nearest Neighbor (KNN) algorithm. Finally, in Module IV, a conditional random field (CRF) algorithm is developed for postprocessing and segmentation result refinement. The proposed method was evaluated on three large-scale LiDAR point cloud datasets in both urban and indoor environments. The experimental results acquired by using different point cloud scenarios indicate our method can achieve state-of-the-art semantic segmentation performance in feature representativeness, segmentation accuracy, and technical robustness.**

Manuscript received May 29, 2019; revised October 12, 2019; accepted December 16, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 41871380 and in part by the Natural Sciences and Engineering Research Council of Canada under Grant RGPIN-50503-10284. The Associate Editor for this article was L. M. Bergasa. *(Corresponding author: Jonathan Li.)*

L. Ma, Y. Li, and W. Tan are with the Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: l53ma@uwaterloo.ca; y2424li@uwaterloo.ca; weikai.tan@uwaterloo.ca).

J. Li is with the Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON N2L 3G1, Canada, and also with the Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: junli@xmu.edu.cn).

Y. Yu is with the Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huaian 223003, China (e-mail: allennessy.yu@gmail.com).

M. A. Chapman is with the Department of Civil Engineering, Ryerson University, Toronto, ON M5B 2K3, Canada (e-mail: mchapman@ryerson.ca).

Digital Object Identifier 10.1109/TITS.2019.2961060

***Index Terms*— Point clouds, mobile laser scanning, convolutional neural network, road object, segmentation, edge convolution, conditional random field, k-nearest neighbor.**

I. Introduction

WITH the increasing market demands of Advanced Driver-Assistance Systems (ADAS), Level-5 fully autonomous driving, autonomously operating robotics, smart cities, and semantic high-definition (HD) maps, mobile laser scanning (MLS) or mobile Light Detection and Ranging (LiDAR) systems have attracted extensive attention of many researchers over the past few years [1]. Such MLS systems could effectively collect high-density and precise 3D point clouds in large-scale road environments [2]. Accordingly, 3D point clouds have been commonly applied in many industrial applications, including 3D object extraction in urban road networks [3], [4], object registration, object tracking [5], object modeling and 3D reconstruction, object classification [6], and semantic segmentation [7]. As a significant requirement of 3D digital cities, 3D semantic segmentation aiming to assign the per point semantic label for all input point clouds is crucial in exploiting the informative values of point clouds for the aforementioned applications [8], [9]. Therefore, in this paper, we specifically concentrate on the foundational and theoretical problems of 3D semantic segmentation using MLS point clouds in large-scale urban environments.

3D point-wise segmentation is to classify each point in the entire point clouds into several homogeneous classes, and semantic labels will be assigned to the points belonging to the same objects or regions [10]. However, it is very challenging to achieve automated and effective point-wise segmentation regarding the high redundancy, uneven point density, and inexplicit structure of MLS point clouds [6]. Generally, 3D semantic segmentation is performed by creating hand-designed feature descriptors. The most representative feature descriptors are comprised of global feature descriptors and local feature descriptors. Such global feature descriptors, e.g. 3D statistical moment [11] and spherical harmonics descriptor [12], are commonly obtained based on the geometrical information of entire MLS point clouds. However, these feature descriptors are very sensitive to occlusions, distortions, and background interferences, resulting in segmentation ambiguities [13]. In addition, local feature descriptors including Spin Image [14], Signature of Histograms of OrienTations (SHOT) [15], Fast Point Feature Histograms descriptor (FPFH) [16], and Fourier power spectrum (FPS) [17], mainly concentrate on the descriptive information of point clouds in local regions. However, such methods could capture few geometrical information of 3D objects. Hence, the representativeness of developed feature descriptors is yet far from satisfaction.

To strengthen the descriptiveness and feature representation of these existing methods, it is effective to learn features at middle and high levels for inherent and additional information taking advantage of the increased performance of computational resources. One promising solution is to use deep learning (DL) models, e.g. deep convolutional neural networks (CNNs) and generative adversarial networks (GANs), to learn deeper and more distinctive feature representations [18]. Accordingly, various methods including voxel-based methods (e.g., VoxNet [19]), multiview-based methods (e.g., MV3D [20]), auto-encoder based methods (e.g., CAE-ELM [21]), graph cut based methods (e.g., ECCNet [22]), and symmetric function based methods (e.g., PointNet [23]), have been proposed to use MLPs for 3D data analysis, object recognition, and semantic segmentation. Although these existing CNN-based methods have achieved a significant enhancement in the representativeness and descriptiveness on several publicly available datasets (e.g., ShapeNet-Part and ModelNet40), it is still challenging to effectively and automatically manipulate MLS point clouds with unordered 3D points, various point densities, outliers, and occlusions, which are inevitable in complex urban environments.

To overcome these challenges, we investigate the feasibility of embedding point-wise CNNs with hierarchical feature representations of point clouds. Yet CNNs were initially developed to deal with 2D images with structured pixel arrays. Such images are organized with regular lattice grids in a specific order, which can be directly fed into CNN-based architectures. It is not feasible to directly perform CNNs on 3d MLS point clouds since they are not in a regular data format or an inherent order. Therefore, to solve this dilemma, we develop a 3D semantic segmentation model aiming to facilitate collaboration between point-wise CNNs and unordered 3D point clouds. The novel architecture of our proposed neural network is to directly consume unstructured 3D points and implement a point-wise semantic label assignment network to learn fine-grained layers of feature representations and reduce unnecessary convolutional computations. To this end, we propose an end-to-end DL framework comprised of the following four modules: (1) point-based 3D convolution, (2) U-shaped downsampling-upsampling framework, (3) dynamic graph edge convolution, and (4) conditional random field (CRF) based postprocessing. This proposed neural network is capable of robustly and efficiently extract global and local features of input point clouds in multiple scales. Furthermore, these proposed and revised models can directly consume 3D point clouds without data conversion and transformation.

Our proposed model has been evaluated on publicly accessible datasets including one large-scale MLS point cloud dataset and two indoor high-density LiDAR datasets. Experimental outputs conclusively demonstrate that the proposed method could achieve superior performance in feature representation, computational efficiency, and robustness. The significant contributions of this paper are described as follows: (1) we revised PointCONV, a multi-scale density-based reweight convolution, which can completely and efficiently approximate the 3D convolution on large-scale unordered point clouds with an efficient computation fashion; (2) we designed a hierarchical U-shaped downsampling-upsampling framework to implement both PointCONV and PointDeCONV for better point-wise segmentation outcomes; (3) we improved the EdgeConv descriptor by optimizing both symmetric aggregation function and edge function to achieve dynamically update the graph of edges and learn more representative features between adjacent points in local neighborhoods; and (4) we performed a CRF algorithm for the label assignment refinement generated by the proposed end-to-end model.

This paper is designed as follows: the related studies of 3D point cloud segmentation are presented in Section 2. Section 3 theoretically and mathematically details the proposed end-to-end DL framework. The test datasets used in this paper are presented in Section 4. Section 5 presents the experimental outcomes, comprehensive discussion, and comparative study, followed by the concluding remarks in Section 6.

II. Related Work

For the past several years, many methods have been developed for 3D object segmentation. This section provides an in-depth review and investigation from the perspectives of 3D point clouds. More detailed literature reviews are further addressed in recently published review articles [24], [25]. Generally, the commonly employed 3D point clouds segmentation methods are classified into two groups: hand-designed feature related algorithms and deep learning related algorithms.

A. Hand-Designed Feature Related Studies

Hand-designed feature descriptors, including both global feature descriptors and local feature descriptors, are created to derive inherent features from 3D point clouds, such features are afterward input into off-the-shelf classifiers (e.g., random forests) [32]. Global feature descriptors are commonly obtained from the geometrical information of entire 3D point clouds. A 3D statistical moment descriptor was developed for the coarse representation of shapes of 3D objects [11]. Furthermore, a shape distribution descriptor was proposed to measure geometrical information of 3D objects [26]. The essential idea is to convert arbitrary 3D object models into parameterized functions that can be directly compared with others. This shape distribution descriptor can effectively eliminate shape segmentation problems to the comparison of probability distributions, which is more robust and straightforward than other shape segmentation methods that need data registration, model fitting, and feature matching [24]. Yet the performance of these global feature descriptors is dramatically impacted by the selection of patch sizes and patch locations. These feature descriptors are highly vulnerable to occlusions, distortions, and background interferences [33]. Moreover, due to the complexity of 3D objects, especially for large-scale MLS point clouds, the computational cost will exponentially increase for the extraction of global feature descriptors.

Compared to global feature descriptors, local feature descriptors generally calculate geometrical information and statistical distributions of key points in the local neighborhoods to construct feature description vectors [34]. As one of the most representative local feature descriptors, Spin Image feature descriptor [14] has been regarded as the benchmark for the performance evaluation of the other local feature descriptors. However, the feature representation ability of Spin Image is relatively poor. To enhance the descriptiveness, a 3D Shape Context feature descriptor [35] was proposed through reconstructing 2D shape context methods on 3D point clouds. Besides, the spatial transformation based feature descriptors, including Heat Kernel Signature (HKS) [36] and 3D Speeded Up Robust Feature (SURF) [37], first transformed the spatial domain to other domains (e.g., spectrum domain), then used the specific information in the transformed domains to describe the key points within local neighborhoods. Accordingly, Rusu *et al.* [16] developed the Fast Point Feature Histograms descriptor (FPFH) by taking the angle differences from a key seed to its neighbors into consideration. Meanwhile, Salti *et al.* [38] proposed a histogram-based descriptor, called Signature of Histograms of OrienTations (SHOT), to extract local surface features. Guo *et al.* [39] presented a Rotational Projection Statistics (RoPS) method, which has been included in the open-access Point Cloud Library (PCL). Rather than learning global and local features or constructing grid-based data formats, Wang and Jia [40] introduced a Frustum ConvNet (F-ConvNet) for 3D object segmentation on outdoor KITTI datasets. Firstly, F-ConvNet generated a collection of frustums to assemble points in local regions. Then, point-wise features represented by frustum-level feature vectors were learned via a fully convolutional network within each frustum. Most significantly, F-ConvNet expects no prior knowledge of the data scenarios and is therefore dataset-agnostic.

Nevertheless, for the above methods, the unavoidable task of unstructured point clouds triangulation could lead to considerable computational complexity and original information loss. Other local feature descriptor generation methods based on the geometric information histograms and orientation gradient histograms depend on the first and second derivatives of the point cloud mesh surfaces, which are prone to noise interferences. Moreover, the majority of local feature descriptors require to first detect and extract key points and then construct the local coordinate references (LRFs). Therefore, the robustness of an LRF has a great impact on the performance of the generated local feature descriptors. In addition, the larger size of local neighborhoods, the more information the local feature descriptors describe, and the more sensitive to occlusions and background interferences.

B. Deep Learning Related Studies

Deep learning is taking off in the communities of artificial intelligence and remote sensing [18], [40]. Compared to hand-designed feature descriptors, deep learning related algorithms follow end-to-end pipelines, where the multi-layer architectures can learn inherent feature representations of high-dimensional data with multiple levels of abstraction [41], [42]. Various DL-based methods have remarkably improved the state-of-the-art in many domains including image recognition, machine translation, and environmental perception [43]. However, the irregular format and unstructured distribution of 3D point clouds make direct application of traditional CNNs challenging. Thus, the fundamental problem of DL-based algorithms is to address feature representations of 3D point clouds. Several end-to-end DL-based methods have been investigated to deal with this dilemma. They are usually categorized into three groups based on the following data processing methods: voxelization-based methods, multiview-based methods, and 3D point-based methods [24].

1) Voxelization-Based Methods: Volumetric methods can transfer 3D point clouds with irregular format into structured voxel data, on which CNN-related neural networks are thus commonly performed. To overcome the over-segmentation and under-segmentation issues normally occurred in complex urban road environments, Luo *et al.* [44] introduced a probability occupancy grid-based method for real-time ground segmentation tasks by employing a single laser scanner. Maturana and Scherer [19] proposed a VoxNet architecture by integrating volumetric occupancy grid representation with a supervised CNN framework for 3D object recognition and autonomous robot operation. Meanwhile, Wu *et al.* [27] developed 3D ShapeNets to describe 3D geometric shapes as probability distributions of binary variables on 3D volumetric grids, then applied a convolutional deep belief network (DBN). But such methods lead to sparse volumes and need lots of memory space and computational powers with an increasing voxel size. Accordingly, space partition methods including Octree-based methods [45], [46] and KD-tree based methods [47]–[49] were created to tackle voxel size and memory explosion problems. However, the above methods solely depend on the partition of a bounding voxel rather than the locally geometrical structures. That is, if the point density is relatively low, there will be not enough points located in the sparsely sampling neighborhoods for volumetric convolutional operation. It normally leads to an excessive requirement of memory footprints and high computation cost.

2) Multiview-Based Methods: To fully take advantages of well-developed DL-based models in image processing and computer vision fields (e.g., AlexNet [50] and Mask R-CNN [51]), many studies converted 3D point clouds into 2D images. The multiview CNN methods were proposed by projecting 3D point clouds into a set of 2D images derived from multiple views. A basic CNN architecture was then employed to train these rendered images and learn representative features [52]. In order to support autonomous driving, Chen *et al.* [20] developed Multi-View 3D networks (MV3D) for onboard sensor fusion and 3D object detection based on the mechanism of multiple views. Bai *et al.* [53] proposed a 3D shape matching and retrieval framework by using projective images of 3D objects. Wen *et al.* [6] first transformed mobile LiDAR point clouds into 2D georeferenced intensity images with 4 cm^2 resolution, an autoencoder-based U-net was afterward proposed for road marking segmentation. Additionally, Qi *et al.* [28] designed an automated pipeline by combining both 3D voxelization and multiview CNNs for 3D object classification and segmentation. Instead of constructing proposals from RGB-D images or converting point clouds into multiple views or volumetric data blocks, Shi *et al.* [54] proposed the PointRCNN model that directly generates high-quality 3D object proposals from raw point clouds using a bottom-up strategy. Then, the resampled points in each proposal were transformed into canonical coordinates to capture more local spatial features. Although these 3D-2D dimensional transformation methods can achieve dominating performances, they introduce the resulting data with redundant volumes and ignore the rich 3D geometric information and spatial correlation of points. Besides, it is challenging to ascertain both the number and direction of views in order that they can cover the whole 3D scenes while preventing self-occlusions.

3) 3D Point-Based Methods: Compared to volumetric methods and multiview-based methods, 3D point-based methods could directly consume 3D points without data format transformation. Considering the permutation invariance and transformation invariance of point clouds, a CNN-based model, called PointNet [23], was proposed to learn inherent features for classification and segmentation tasks. However, PointNet cannot capture local features of point clouds, which decreases its strength to identify fine-grained patterns and generalizability to large-scale point clouds. Subsequently, an improved version, called PointNet++ [55], was developed to learn more local features than the PointNet by calculating the metric space distances. PointNet++ used the farthest point sampling (FPS) and multi-scale grouping algorithms to leverage local features from coarse layers to fine layers at multiple scales for robustness improvement. In general, both PointNet and PointNet++ are pioneers in DL-based models that directly use 3D point clouds for classification and segmentation in complex scenes. The fundamental structure developed in both PointNet and PointNet++ for feature aggregation from various input points is max-pooling operation. Nevertheless, a max-pooling layer uniquely remains the largest activation on different features in local neighborhoods or global regions, which leads to inevitable information loss for segmentation tasks. Furthermore, the lack of deconvolution operation also limits their performances.

To solve these problems, many PointNet-derived deep learning models apply PointNet recursively and optimize their performances to deliver state-of-the-art. Li *et al.* [29] developed a PointCNN framework to use a hierarchical convolution structure and an X-Conv operator that aggregate input points into fewer points with richer features. However, PointCNN is not capable of achieving permutation invariance, which is significant for point cloud segmentation. Jiang *et al.* [30] proposed the PointSIFT model applying a scale-invariant feature transform (SIFT) descriptor to capture the shape representation of input points. Additionally, dynamic graph CNN (DGCNN) [31] implemented a framework that is able to dynamically update the graph of point clouds. Moreover, Yi *et al.* [56] introduced a Generative Shape Proposal Network (GSPN) for 3D object segmentation by employing an analysis-by-synthesis approach and reconstructing shapes as object proposals from noisy observation, which achieves state-of-the-art performance on KITTI LiDAR datasets. Other methods such as SpiderCNN [57] also have demonstrated their superior performance in point cloud object detection and semantic segmentation tasks. However, there are very few applications that apply CNN-based models for segmentation using MLS point clouds, especially in large-scale urban road environments due to high computational complexity and memory occupation. Besides, it is challenging to directly apply traditional CNNs on point clouds regarding to their irregular formats. Additionally, such CNN-based methods always utilize fix-sized filters (e.g., 1×1 and 5×5) to apply convolution on unordered and irregular point clouds, which could lead to remarkably redundant convolutional operations and extra memory overhead.

To summarize, most of the existing hand-designed feature descriptors or feature-related methods that only concentrate on either global or local statistical information, which results in a performance reduction in representativeness and descriptiveness. Also, traditional CNN convolution applied to 3D point clouds could lead to high computational consumption and edge information loss. Our method follows an idea same to the 3D point-based methods. However, different from the way of only focusing on the global features in PointNet [23], we revise and propose point-wise convolution and edge convolution algorithms to capture both global and local features of 3D point clouds in multiple scales, avoiding the information loss in local neighborhoods and useless convolutional computations. Moreover, we develop a hierarchical downsampling-upsampling framework to extract representative and high-level features for robustly and accurately 3D object segmentation in complex urban environments.

III. Methodology for Road Object Segmentation

In this section, we detail the theoretical and logical principles of the proposed model for 3D road object segmentation from MLS point clouds. This novel model, named MS-PCNN, mainly contains four modules: convolution on 3D points, multi-scale feature extraction, dynamic edge feature extraction, and conditional random field post-processing.

More specifically, Module I is designed to construct a revised convolutional kernel, particularly for 3D point clouds. A Monte Carlo approximation of the 3D continuous convolutional operators is first applied followed by dynamic density scales to re-calculate the optimized weight functions. In Module II, a U-shaped downsampling-upsampling architecture is proposed to leverage both global and local features in multiple scales. Next, in Module III, high-level local edge features in 3D point neighborhoods are further extracted by using an adaptive graph convolutional neural network based on the K-Nearest Neighbor (KNN) algorithm. Finally, in Module IV, a conditional random field algorithm is developed for postprocess and segmentation result refinement, and 3D road objects are therefore segmented. Fig. 1 presents the detailed workflow of the proposed MS-PCNN model.

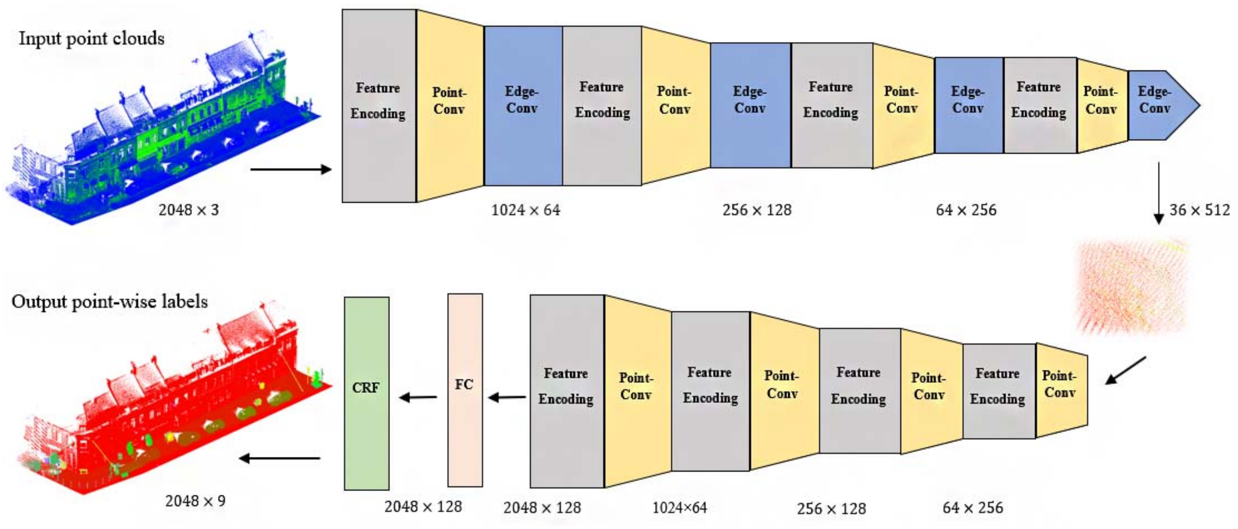


Fig. 1. Illustration of the proposed MS-PCNN model architecture in this paper.

A. Module I: Convolution on 3D Points

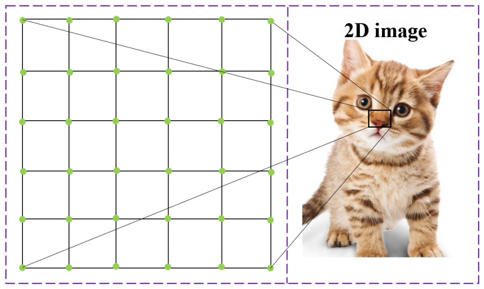
Although CNN-based methods have demonstrated the superior performance on recognition, classification and segmentation tasks using regular data formats, such as 2D images or 3D voxelized grids, it is very difficult to provide promising solutions that directly apply convolutions on 3D point clouds. Inspired by [58], as a revised convolutional operation, MS-PCNN extending conventional 2D image convolutions into 3D point clouds is accordingly proposed in this paper. In general, convolutional operations are determined by using:

$$(F * G)(x) = \iint_{\Delta x \in \mathbb{R}^d} F(\Delta x) G(x + \Delta x) d(\Delta x) \tag{1}$$

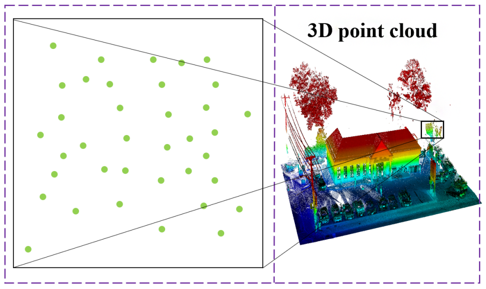
where $F(x)$ and $G(x)$ are two functions, x is a d-dimensional vector, and $\mathbb{R}^d$ denotes a d-dimensional Euclidean space. 2D Images represented by grid-structured matrices are normally regarded as discrete functions. In traditional CNNs, various kernel filters (e.g., 1×1, 5×5, and 7×7) are assigned to focus on small-sized local neighborhoods. Moreover, the relative positions among different pixels are always certain in each local region, as illustrated in Fig. 2(a). Diverse filters can be effectively employed to calculate the sum of real-valued weights for different locations in the given local neighborhood.

In contrast, point clouds are considered as a collection of discrete 3D points p_i $(i = 1, 2, \ldots, n)$ containing xyz coordinate information and related characteristics including color, intensity, and normal. Compared to grid-structured images, point clouds have an irregular format with the unfixed arrangement. Hence, as shown in Fig.2(b), the relative positions of point clouds are different within different local regions, resulting in traditional convolutional filters used on regular data formats (e.g., images) cannot be directly utilized on point clouds.

In order to make full use of convolutional operations on 3D point clouds, Wu *et al.* [58] proposed a permutation-invariant convolutional filter, called PointCONV. The main idea of PointCONV is to define the 3D convolutions for continuous functions by the following equation:



(a)



(b)

Fig. 2. Data format comparison between 2D images and 3D point clouds: (a) 2D images. (b) 3D point clouds.

$$3D\,\text{Conv}(H, J)_{xyz} = \iiint_{(\varphi_x,\varphi_y,\varphi_z)\in E} H(\varphi_x, \varphi_y, \varphi_z) \cdot J(x+\varphi_x, y+\varphi_y, z+\varphi_z)\, d\varphi_x\varphi_y\varphi_z \tag{2}$$

where $H(x)$ and $J(x)$ are two functions, $J(x + \varphi_x,\ y + \varphi_y,\ z+\varphi_z)$ represents the feature of a point p_i $(i = 1, 2, \ldots, n)$ in the local neighborhood E, where (x,y,z) is the center position of this local region. Specifically, point clouds are interpreted as non-uniform samples in the continuous 3D space. Therefore,

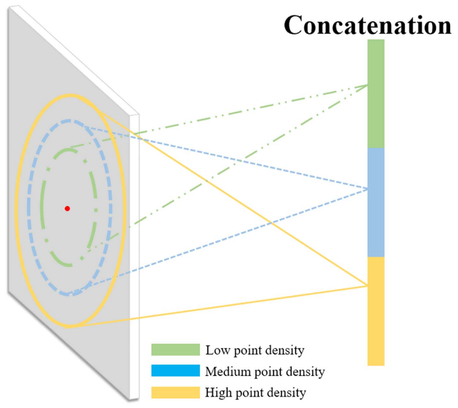


Fig. 3. Illustration of the multi-scale kernelized point density estimation.

PointCONV is defined as follows:

$$(F, H, J)_{xyz} = \sum_{(\varphi_x,\varphi_y,\varphi_z)\in E} F\left(\varphi_x, \varphi_y, \varphi_z\right) H\left(\varphi_x, \varphi_y, \varphi_z\right) \times J\left(x+\varphi_x, y+\varphi_y, z+\varphi_z\right) \quad (3)$$

where $F(\varphi_x, \varphi_y, \varphi_z)$ indicates the inverse density given the point $(\varphi_x, \varphi_y, \varphi_z)$. $F(\varphi_x, \varphi_y, \varphi_z)$ is significant since the downsampled point clouds are non-uniformly distributed. However, the point densities in different local neighborhoods are various across the entire point clouds. The key idea is to employ multi-layer perceptrons (MLPs) for the weight function approximation based on the 3D positions $(\varphi_x, \varphi_y, \varphi_z)$ and the inverse density values $F(\varphi_x, \varphi_y, \varphi_z)$ using a density estimation algorithm. However, Wu *et al.* [58] considered the approximation of the density scale in a fixed threshold rather than multi-scale or dynamic scales, which leads to approximations of the 3D convolutional operator far from satisfactory.

Different from [58], a multi-scale kernelized point density calculation algorithm is proposed in this paper followed by a non-linear transformation algorithm, which is implemented during feature extraction stages (see blue bars in Fig. 4). Different colors in Fig. 3 represent different point densities. The MS-PCNN network is designed to capture multi-scale patterns by grouping 3D points in multiple scales followed by according MLPs to extract inherent features within each scale. Then, features learned from various scales are concatenated together for the multi-scale feature encoding purpose. The raw points are randomly dropped out with a randomized probability for each point. According to prior knowledge, we choose the randomized dropout rate θ uniformly sampled in the range of $[0, \rho]$, where $\rho = 0.9$ to avoid the sampling deficiency. To achieve invariant permutation of 3D points, the weights learned from different MLPs in the revised PointCONV are shared in the whole point clouds. According to the proposed multi-scale kernel density estimation (MKDE) and non-linear transformation algorithms, the inverse density scales $F(\varphi_x, \varphi_y, \varphi_z)$ can be adaptively calculated with multi-scale point density estimation in local regions.

Fig. 4 indicates the revised PointCONV framework within a local neighborhood. As can be seen, the white rectangles represent the features by concatenating interpolated features with features learned from MLPs with the same resolution using across-level skip connections. Blue color bars indicate feature extraction results by using MLPs, while light green bars denote the downsampling and grouping modules that are similar to the ones employed in PointNet++. More specifically, we conducted the iterative farthest point sampling (FPS) to subsample the raw point clouds by calculating the Euclidean distances from 3D points to the given centroids, which can generate receptive fields in a data-dependent fashion. Assuming that C_i and C_o be the number of channels about the input features and output features, respectively. (k, C_i, C_o) is regarded as the index of K-th neighbor, C_i-th channel of input features and C_o-th channel of output features. The input p_i $(i = 1, 2, \ldots, n)$ provide 3D coordinates $p_i = (x_i, y_i, z_i)$ where $p_i \in \mathbb{R}^{3\times K}$, which is calculated by subtracting the centroid coordinate and the input feature $c \in \mathbb{R}^{C_i\times K}$ of the local neighborhood. 1×1 convolutional kernel size is ascertained to perform MLPs. The outputs of the weight functions are $W \in \mathbb{R}^{(C_i\times C_o)\times K}$. Accordingly, $W(k, C_i) \in \mathbb{R}^{C_o}$ denotes a weight vector, and the density scale is $F \in \mathbb{R}^K$. In order to capture more high-level local features at multiple scales in each local region, the multiple thresholds of K are selected based on the diverse distributions of point clouds, and the average value of MKDE is then estimated. According to prior knowledge and the accessible computation capability, K is predefined as 128, 64, 32, and 16 respectively in this paper. After convolutional operations, the input features F_i captured in each local region with multi-scale K points are fed into the following equation to obtain the output features $F_o \in \mathbb{R}^K$:

$$F_o = \sum_{k=1}^{K}\sum_{c_i}^{c_i} F(k) W(k, c_i) F_i(k, c_i) \quad (4)$$

However, such revised PointCONV operations are time-consuming and huge memory-overhead, especially for the weight approximation. For a certain point cloud, each local neighborhood is assigned to the equivalent weight functions that are encoded from MLPs. Nevertheless, the weights calculated by different weight functions from various point clouds are different. Accordingly, the sizes of the weight filters can be determined as follows:

$$S_w = B \times N \times K \times C_{in} \times C_{out} \quad (5)$$

where S_w is the size of weight filters computed by MLPs. B is the mini-batch size, N represents the number of points within each point cloud, K indicates the number of points within each local neighborhood, C_{in} is the number of input channels, and C_{out} denotes the number of output channels. For instance, if $B = 64$, $N = 1024$, $K = 64$, $C_{in} = C_{out} = 128$, respectively, the memory size for the generated weight filters are over 16 GB for each layer, which results in huge memory consumption in the training phase. Therefore, to tackle this problem, we further refine PointCONV implementation by optimizing matrix multiplication and 2D convolution operations. The revised PointCONV is equivalent to the following

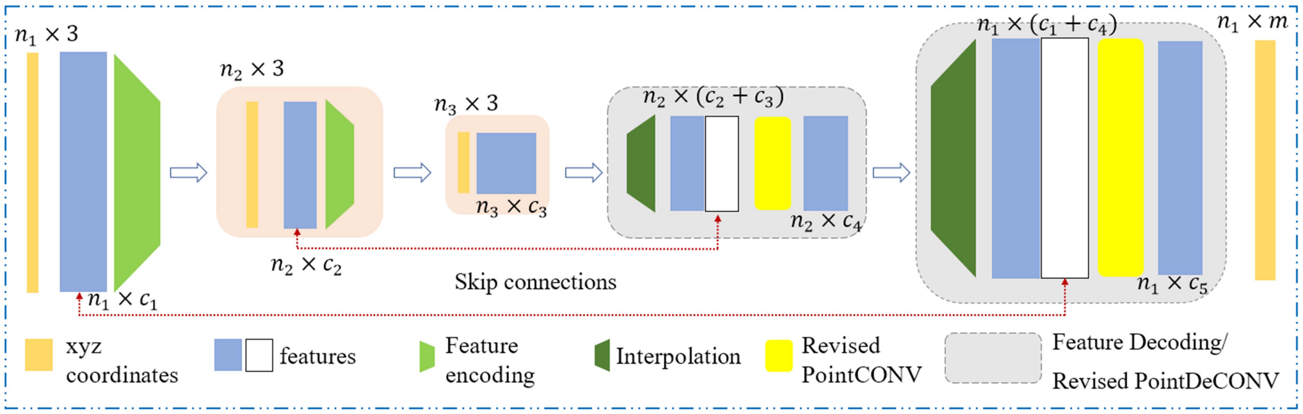


Fig. 4. The revised PointCONV framework with feature encoding and propagation.

equation:

$$F_{out} = CONV_{3\times3}\left(H, (F \cdot F_{in})^T \otimes M\right) \tag{6}$$

where $M \in \mathbb{R}^{K\times C_{mid}}$ denotes the inputs fed into the last layer of MLP to calculate the weight function, $H \in \mathbb{R}^{(C_{in}\times C_{out})\times C_{hid}}$ is the weights in the last layer of MLP, F indicates the density scale, and $CONV_{3\times3}$ is 3×3 convolutional operation. Since the last layers of MLPs are generally linear layers, $\widetilde{F} = F \cdot F_{in}$ is therefore rewritten within each local neighborhood. Accordingly, let the weight function $W = CONV_{3\times3}(H, M) \in \mathbb{R}^{(C_{in}\times C_{out})\times K}$, k is the index of points in local regions, and c_{in}, c_{hid}, c_{out} are the indices of the input, hidden and output layer, respectively. Therefore, the revised PointCONV can be expressed as follows:

$$F_{out} = \sum_{k=0}^{K-1}\sum_{c_{in}=0}^{C_{in}-1}\left(W(k, c_{in})\,\widetilde{F_{in}}(k, c_{in})\right) \tag{7}$$

$$W(k, c_{in}) = \sum_{c_{hid}=0}^{C_{hid}-1}\left(M(k, c_{hid})\,H(c_{hid}, c_{in})\right) \tag{8}$$

According to both Eqs. (7) and (8), the revised PointCONV is thus determined by:

$$\begin{aligned} F_{out} &= \sum_{k=0}^{K-1}\sum_{c_{in}=0}^{C_{in}-1}\left(\widetilde{F_{In}}(k, c_{in})\right)\sum_{c_{hid}=0}^{C_{hid}-1}\left(M(k, c_{hid})\,H(c_{hid}, c_{in})\right)\\ &= CONV_{3\times3}\left(H, \widetilde{F_{In}^{T}}M\right) \end{aligned} \tag{9}$$

Consequently, the previous PointCONV is equivalently converted into a 2D 3×3 convolution and a matrix multiplication. In this revised model, we refine the matrix multiplication by dividing the weight filters into two parts: the convolutional kernel H and the intermediate output M. In addition, instead of using 1×1 convolution, we employ 3×3 convolution to not only deliver promising outputs but also effectively reduce computational costs. Assuming that $C_{hid} = 64$, the memory usage is about 0.251 GB for each layer, which is only 1/64 of the original PointCONV with the same parameters as shown in Fig.4.

Therefore, this revised PointCONV operation can effectively construct a network and approximate the continuous weights for convolutions on point clouds. Compared to the traditional convolutions, the revised PointCONV-based convolution that only considers the relative coordinates as inputs could output multi-scale densities and weights across the whole point clouds, which considerably decreases the computational cost caused by traditional discretized and fix-sized convolutions.

B. Module II: U-Shaped Downsampling-Upsampling Architecture

After implementing PointCONV operations, the original input point clouds have been subsampled into various resolutions. However, for object segmentation task especially as semantic labeling, the point-wise segmentation for the entire point clouds is needed. To acquire high-level features for the whole point clouds in both global and local scales, a hierarchical framework that could propagate features from subsampled point clouds to relatively dense ones is required. Therefore, a U-shaped downsampling-upsampling architecture is proposed by taking PointCONV operations into consideration. According to the revised PointCONV mentioned in Section 3.1, we capture more high-level features by regarding a revised PointDeCONV layer as deconvolutional operations.

As shown in Fig. 4, PointDeCONV implementation mainly contains two processes: interpolation and revised PointCONV. First, we perform interpolation to assemble different level features from previous layers. According to the three nearest points, the interpolation is carried out to linearly interpolate features. Subsequently, such interpolated features are concatenated with features learned from MLPs with the same resolution using across-level skip links. Finally, the revised PointCONV is thus employed on the concatenated features to catch the final deconvolution outputs. Accordingly, this recursive process will not terminate until the features learned from all point clouds have been propagated back to the initial resolution.

C. Module III: Dynamic Graph Edge Convolution

Although the proposed MS-PCNN hierarchical framework embedded with revised PointCONV and PointDeCONV operations could obtain features for all input point clouds in multiple scales, edge features between a point and its adjacent neighbors have not been taken into consideration. To address

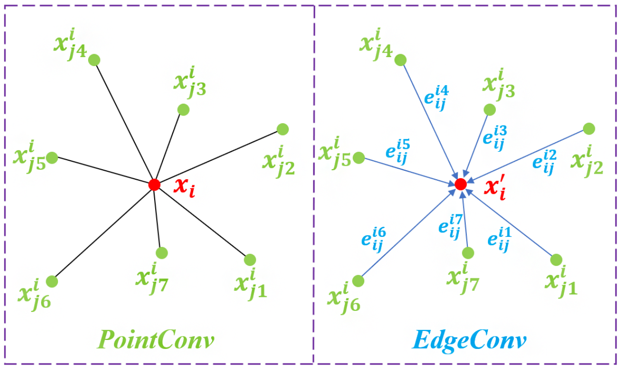


Fig. 5. Illustration the principles of the revised PointCONV and EdgeConv operations.

this drawback, a PointCONV-based dynamic graph edge convolution operator is proposed to capture local geometrical edge structures based on the EdgeConv descriptor [31]. As shown in Fig. 1, a local neighborhood graph is constructed followed by PointCONV-based convolutional operations on the connected edges. Compared to conventional graph CNNs, the graphs introduced in this paper are dynamically updated rather than fixed after feature extraction layer. Specifically, the K-nearest neighbors of a point dynamically change between two adjacent layers of the model and are accordingly calculated the sequence of embeddings. Fig. 5 illustrates the principle of revised EdgeConv operation compared with revised PointConv operation. As can be perceived, by adding dynamic graph edge convolutions into the proposed model, not only point-wise geometric information but also edge informative features between a certain point and its neighbors are considered to capture more descriptive features in a local region.

Assuming that a directed graph $G = (V, E)$ denoting the local structure of each point cloud, $V = (1, 2, \ldots, n)$ and $E \subseteq V \times V$ are the vertices and edges, respectively. To simplify this problem, we build a graph G as the KNN-graph in D-dimensional space (generally $D = 3$ representing xyz coordinates of point clouds) and define edge features as $e_{ij} = g_\phi(x_i, x_j)$, where $g_\phi \in \mathbb{R}^D \times \mathbb{R}^D$ denote parameterized nonlinear functions with a collection of parameters ϕ. After that, the EdgeConv operation is defined by conducting a channel-wise symmetric aggregation implementation (e.g., sum) on the edge features from each point. Therefore, the output of EdgeConv operation at the i-th vertex is performed as follows:

$$x'_i = \square_{j:(i,j)\in E}\, g_\phi\left(x_i, x_j\right) \tag{10}$$

where $\square$ represents a symmetric aggregation function. In this paper, differently from [31], we apply max as the aggregation function rather than sum to reduce the computational consumption. Instead of $g_\phi(x_i, x_j) = g_\phi(x_i)$, $g_\phi(x_i, x_j) = g_\phi(x_j - x_i)$ or $g_\phi(x_i, x_j) = g_\phi(x_i, x_j - x_i)$ tried in [31], $g_\phi(x_i, x_j) = g_\phi(x_i, x_j + x_i)/2$ is adopt in this paper as an symmetric edge function. By combining both the global structures and local neighborhood characteristics, such a function is capable of acquiring more inherent and high-level features in an effective manner. Moreover, due to the variations of the number of points in each local neighborhood, the average-based asymmetric edge function $g_\phi(x_i, x_j) = g_\phi(x_i, x_j + x_i)/2$ is prone to error reduction and keep much information for further feature encodings.

Furthermore, it is remarkably significant to recalculate a new graph using the KNN algorithm within the D-dimensional feature space generated by previous layers. Inspired by [31], the key idea of dynamic graph construction is employed in this paper. Thus, a new graph $\mathrm{G}^l = (V^l, \mathrm{E}^l)$ is constructed at each layer. Consequently, the D^{l+1}-dimensional outputs are calculated by using the revised EdgeConv to the D^l-dimensional outputs of the l-th layer from the following equation:

$$x_i^{(l+1)} = \square_{j:(i,j)\in E^{(l)}}\, g_\phi^{(l)}\left(x_i^{(l)}, x_j^{(l)}\right) \tag{11}$$

where $\mathrm{g}_\phi{}^{(l)} \in \mathbb{R}^{D^{(l)}} \times \mathbb{R}^{D^{(l)}}$. Such revised EdgeConv can be easily fed into existing architectures to boost the segmentation performance. In this paper, we combine the revised EdgeConv with the basic version of hierarchical PointCONV and PointDeCONV framework. As depicted in Fig. 1, an EdgeConv layer is employed after each revised PointCONV layer, followed by a fully connected layer then fed back to PointDeCONV layers. Within each EdgeConv module, $\mathrm{g}_\phi{}^{(l)}(\mathrm{x}_\phi{}^{(l)}, \mathrm{y}_\phi{}^{(l)}) = g_\phi(\mathrm{x}_\phi{}^{(l)}, \mathrm{y}_\phi{}^{(l)} + x_\phi{}^{(l)})/2$ is applied as a shared edge function, and we perform the max operation as the aggregation function. Moreover, the number of nearest neighbors k is predefined to be 32 in this paper for the effective segmentation process.

D. Module IV: Post-Processing

Both CNNs and CRFs have demonstrated dominating performance in semantic segmentation tasks for 3D point clouds [60], [61]. Precise point-wise semantic segmentation requires completely understanding not only high-level features of road objects but also mid- or low-level details. Such details are essential to ensure the consistency of point-wise label prediction. For instance, if two points are close to each other and have similar reflectance values, it is reasonable that these two points pertain to the same road object and therefore have the same semantic label. Thus, we perform a CRF algorithm for the label map refinement produced by the proposed PointCONV. Typically, an energy function is applied to CRF models using the following equation:

$$E(l) = \sum_{i=1}^{n} u_i\left(l_i\right) + \sum_{i,j}^{n} v_{i,j}\left(l_i, l_j\right) \tag{12}$$

where l_i denotes the i-th predicted label, $i = 1, 2, , n$, and n is the total number of point clouds. We use $u_i(l_i) = -log P(l_i)$ as the predicted probability $P(l_i)$ from the revised PointCONV. The second term in Eq. (12) indicates the penalty to assign labels to a couple of points and is therefore determined by $v_{i,j}(l_i, l_j) = \mu(l_i, l_j)\sum_{p=1}^{P} w_p k^p(f_i, f_j)$, where $\mu(l_i, l_j) = 1$ if $l_i \neq l_j$ or 0 otherwise, k^p represents the p-th Gaussian kernel depending on extracted features f from points i and j, and w_p denotes constant coefficients. In this paper, two

TABLE I
DETAILED DESCRIPTIONS OF DIFFERENT TEST DATASETS USED IN THIS PAPER

Datasets	Types	No. of classes	No. of objects	No. of points	Scale	Sensor
Paris-Lille-3D [60]	Urban	9	2,479	143.8 M	1,940 m	Velodyne HDL-32E LiDAR
ScanNet [61]	Indoor	20	>100,000	2.5 M (views)	-	RGB-D sensor
S3DIS [62]	Indoor	13	6,005	215 M	6,020 m^2	RGB-D sensor

Gaussian kernels are chosen as follows:

$$k_1 \exp\left(-\frac{\|p_i-p_j\|^2}{2\sigma_\alpha^2}-\frac{\|x_i-x_j\|^2}{2\sigma_\beta^2}\right)+k_2 \exp\left(-\frac{\|p_i-p_j\|^2}{2\sigma_Y^2}\right) \tag{13}$$

where $k_1\exp(\cdot)$ is determined by the 3D coordinates (x,y,z) and angular positions p of two adjacent points, and $k_2\exp(\cdot)$ is calculated only relying on angular positions. σ_α, σ_β and σ_γ are three predefined hyperparameters. Accordingly, the fine-grained point-wise label prediction is achieved by minimizing the CRF energy function defined in Eq. (12). Although accurate minimization of Eq. (12) is intractable, Chen *et al.* [59] developed and revised a mean-field iteration method to handle this problem effectively and appropriately. More detailed exact minimization process can be found in [59]. The CRF used in this paper can effectively leverage the prediction and confidence produced by the PointCONV-based classifier, as well as semantic label assignment between two similar points in each local region.

To compute and minimize the loss generated by MS-PCNN model, the off-the-shelf softmax cross entropy loss function is utilized after implementing CRF-based post-processing. More specifically, the softmax cross entropy is defined as follows:

$$LOSS = L(g, h(y)) = -\sum_{i=1}^{N} g_i \log S_i \tag{14}$$

where g_i represents the one-hot label of i-th training sample, N denotes the batch size, and $S_i = \mathrm{e}^{V_i}/\sum_j \mathrm{e}^{V_j}$ is the softmax prediction score vector. The main objective is to minimize the loss function expressed in Eq. (14). Finally, the point-wise semantic label is determined based on the prediction score vector S_i.

E. Implementation Details

In all designed experiments, we test the proposed neural network using Tensorflow on Nvidia GTX 1080 Ti GPU and 32 GB RAM. Moreover, we optimized the network using adaptive moment estimation (Adam) optimizer which is built-in in Tensorflow. Batch normalization (BN) and rectified linear unit (ReLU) were employed after each MLP layer, except for fully connected (FC) layers. Several hyperparameters such as the batch size and initial learning rate were optimized during the training phase to determine the optimal combination by using the grid search approach. Specifically, the batch size, initial learning rate, the momentum of Adam, and dropout rate were predefined in the range of [8, 16, 32], [0.01, 0.001, 0.001], [0.80, 0.85, 0.90], and [0.5, 0.6, 0.7], respectively. The Overall Accuracy (OA) and Intersection over Union (IoU) were used as the performance evaluation matrices. By implementing many experiments with all possible hyperparameter combinations, an optimal combination was determined as (8, 0.001, 0.9, 0.5). Namely, MS-PCNN model was trained using Adam with a momentum of 0.9, a dropout rate of 0.5, and a batch size of 8. The initial learning rate was 0.001 with a decrease rate of 50% in every 25 iterations. Each test dataset was divided into 70%, 20%, and 10% subsets for training, testing, and validating, respectively. Finally, a total of 200 epochs was applied for training purpose.

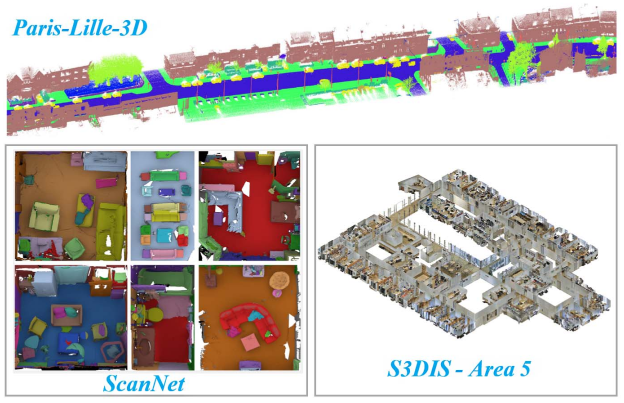


Fig. 6. Samples of test datasets used in this paper.

IV. DATASETS

We employed a large-scale outdoor MLS dataset, called Paris-Lille-3D [60], collected in complex urban environments. Moreover, to test the robustness and scalability of our developed model, two highly dense indoor LiDAR datasets ScanNet [61] and S3DIS [62], were further used. Fig. 6 indicates several sample data, and Table I details the descriptions of different test datasets.

A. Paris-Lille-3D

Paris-Lille-3D point cloud dataset was collected in the two metropolitan areas, namely Paris and Lille, France, using an MLS system equipped with a Velodyne HDL-32E LiDAR. The Velodyne HDL-32E LiDAR sensor can obtain a maximal measurement rate of 700,000 points per second in an effective scanning range of 80 m to 120 m. Such a sensor is installed at the rear roof of the vehicle with an angle of 30° between the horizontal axis and rotation axis, which can achieve a 2 cm measurement accuracy at the speed of up to 60 km/hr, resulting in MLS point densities ranging from 1500-2000 points/m^2.

In Paris-Lill3-3D, there are three data acquisition trajectories: Lille1_1 is a length of 620 m urban road segment with 30.2 million points, Lille1_2 is a length of 530 m urban road corridor with 30.1 million points, Lille 2 is a length of 340 m urban road with 26.8 million points, and Paris provides a 450 m length of urban road with 45.7 million points, respectively. There are 9 object classes were manually labeled as Ground, Buildings, Poles, Bollards, Trash Cans, Barriers, Pedestrians, Cars, and Natural with a total number of 2,479 object instances. Moreover, a total of 30 million points without labels are released as official test datasets. This dataset is obtained from complex urban road environments, it shows surveying conditions with occlusions and varying point densities in the real-world scenarios, thus resulting in considerable difficulties for road object segmentation using this dataset.

B. ScanNet and S3DIS

The ScanNet dataset was generated from over 1,500 scans by using RGB-D video streaming in indoor environments, such as offices, apartments, conference rooms, etc. There is a total of over 100,000 CAD instances, which are retrieved and placed on the surface reconstructions for semantic voxel labeling. This dataset was manually interpreted and labeled into 20 classes, such as Floor, Desk, Curtains, and Bathtubs.

The S3DIS dataset contains 13 object categories and 6,005 object instances with structural elements including Ceiling, Floor, Wall, Beam, Column, Window, Door, and moveable elements including Table, Chair, Sofa, Bookcase, Board, and others, which were collected in 11 scene categories (e.g., hallways and lobbies) and accordingly labeled. This dataset was generated from 6 different building areas with a total area of 6,020 m^2, 1.2 million of mesh faces, and a total number of 695 million 3D points, respectively. Both ScanNet and S3DIS datasets are commonly applied for object semantic segmentation tasks, which conduces to implement a comparative study between MS-PCNN model proposed in this paper and other existing methods.

V. Results and Discussion

A series of experiments were carried out to evaluate the performance of MS-PCNN network. This section introduces the optimized hyperparameters and experimental results, followed by efficiency evaluation and comparative study in terms of accuracy and efficiency.

A. Hyperparameter Optimization

The proposed MS-PCNN framework has two essential hyperparameters: σ, the bandwidth in multi-scale kernel density estimation; and k, the number of points in each local neighborhood. To achieve the optimal hyperparameter settings, MS-PCNN model performance was evaluated through multiple experiments based on two evaluation matrices OA and IoU, which can be calculated as follows:

$$OA = \frac{\sum_{i=1}^{N} c_{ii}}{\sum_{j=1}^{N}\sum_{k=1}^{N} c_{jk}} \tag{15}$$

$$IoU = \frac{c_{ii}}{c_{ii} + \sum_{j\neq i} c_{ij} + \sum_{k\neq i} c_{ki}} \tag{16}$$

where OA metric represents the overall accuracy of segmentation results, and IoU metric measures the percent overlap between the target mask and the segmentation output. N is the number of classes, $c \in \mathbb{R}^{N\times N}$ is a confusion matrix of the segmentation method, where c_{ij} is the number of points from ground-truth class i predicted as class j.

Before conducting various experiments, the Paris-Lille-3D dataset was preprocessed by first downsampling input point clouds and then rotating and jittering them to enhance the robustness and applicability of the MS-PCNN network. Additionally, to ensure geospatial correlation among point clouds in the local neighborhoods, the coordinates of all point clouds were normalized to $[-1, 1]$ in a trajectory interval of 5 m. According to prior knowledge, we tested the performance of MS-PCNN model using 5 (options of σ) 4 (options of k) $=$ 20 combinations. Since the number of combinations is relatively large, we evaluated the different performance of each hyperparameter setting through the variable-controlling approach. That is, we only changed the value of one hyperparameter each time, while remained the other hyperparameter values the same. To evaluate the influence of different hyperparameter combinations, the OA-IoU curve for all categories can be generated. Intuitively, the OA-IoU curve would fall in the top-right region of the plot, which indicates the MS-PCNN model can produce both high overall accuracy and IoU [33].

1) Size of σ: The size of bandwidth σ has a significant impact on the performance of MS-PCNN model. An appropriate value of σ enables the model to learn more local features from input point clouds. The value of σ varies in the range of [0, 1]. More specifically, the smaller σ is, more local information the model can capture, but more computational energy consumes. To achieve an optimal balance between model performance and computational costs, we tested the performance of MS-PCNN network by using different σ values, i.e., 0.05, 0.10, 0.15, 0.20, and 0.25 on three test datasets, while keeping $k = 32$ all the time by running 200 training epochs.

Fig. 7(a) presents the OA-IoU curve based on different σ values. Note that, the performance of MS-PCNN enhances with the decrease of σ, which achieves the best performance (i.e., $OA = 97.2\%$ and $IoU = 68.4\%$) while setting $\sigma = 0.10$. The reason is that in the process of multi-scale kernel density estimation, MS-PCNN could capture more details with the decreasing values of σ. However, point-wise semantic segmentation performance decreases by 1.1% when changing bandwidth values from $\sigma = 0.10$ ($IoU = 68.4\%$) to $\sigma = 0.05$ ($IoU = 67.3\%$), that is because the MS-PCNN model is overfitting due to much redundant information used in the training phase. Therefore, we determined $\sigma = 0.10$ as the optimal hyperparameter value in order that the MS-PCNN model can deliver high robustness and computational efficiency.

2) Size of k: The number of points in each local neighborhood, namely k, determines both the descriptiveness and robustness of MS-PCNN model in local feature extraction. It is normally predefined as $k = 8$, 16, 32,or 64 based on the different point densities for different test datasets. Accordingly,

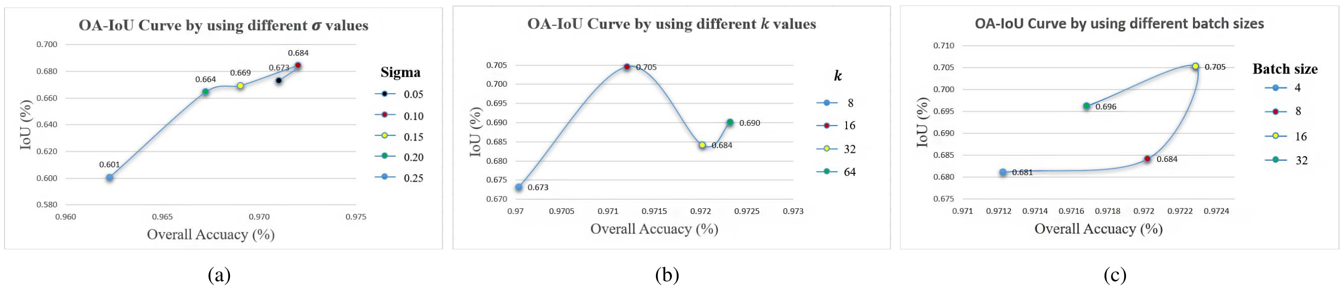


Fig. 7. Model performance evaluation through OA-IoU curves: (a) Using different values. (b) Using different k values. (c) Using different batch sizes.

we evaluated the performance of MS-PCNN model by using different predefined k values on three test datasets, while keeping $\sigma = 0.10$ all through 200 training epochs.

Fig. 7(b) shows the $OA\text{-}IoU$ curve by using different k values. Different point densities have a significant impact on the selection of k values. More specifically, to obtain representative features in local regions, a relatively large k value should be selected for point cloud scenes with high point densities. Note that, the MS-PCNN model can achieve the best performance (i.e., $OA = 97.1\%$ and $IoU = 70.5\%$) for point-wise segmentation while setting $k = 16$ on Paris-Lille-3D dataset. Obviously, for the per-point segmentation task, the $mIoU$ increases by 2.1% by changing $k = 32$ to $k = 16$. Moreover, compared to $k = 32$ or 64, the computational efficiency at the stages of k-nearest neighbor searching and edge convolutions is considerably improved by setting $k = 16$. Thus, in MS-PCNN, we defined $k = 16$ as the optimal hyperparameter value for high segmentation accuracy and relatively low computational costs.

Moreover, we also evaluated the influence of using different batch sizes during the training phase. Fig. 7(c) presents the $OA\text{-}IoU$ curve by varying batch sizes from 4 to 24 (i.e., 4, 8, 16 and 24) based on the computational power that we access, while keeping other hyperparameters the same (e.g., $\sigma = 0.10$ and $k = 16$). Generally, the larger the batch size, the more global feature the model captures, yet the more computational power the model requires. As can be seen, the MS-PCNN model can deliver the best segmentation accuracy by setting the batch size to be 16 ($mIoU = 70.5\%$). Accordingly, we ascertained an optimal hyperparameter combination as $\sigma = 0.10$, $k = 16$, and batch size to be 16, respectively.

B. Segmentation on Paris-Lille-3D

According to different experiments by using various combinations, we determined the optimal combination as $\sigma = 0.10$ and $k = 16$ on the Paris-Lille-3D test dataset. Moreover, the initial learning rate, batch size, momentum of Adam, dropout rate and epochs are 0.001, 16, 0.9, 0.5, and 200, respectively, which can deliver the best segmentation result. Since the design of the MS-PCNN architecture depends on experience, other parameters are thus ascertained through trial and error. For instance, when determining the dimension of the output channel, it is common to utilize an increasing size (e.g., from 64 to 512) in the encoding layers and a decreasing size (e.g., from 512 to 128) in the decoding layers [58].

Fig. 8 illustrates the experimental result by testing with Lille2 dataset, which demonstrates that MS-PCNN model is able to achieve promising solutions for point-wise segmentation tasks in large-scale urban environments. Although mobile LiDAR point clouds collected in urban road scenes are very different from small-scale CAD models, the segmentation results indicate a large number of road objects (e.g., buildings and poles) were effectively segmented and the road surfaces were completely extracted. However, some points failed to be segmented, which indicates that certain points were mis-assigned as other semantic labels, as illustrated in Fig.9. The complexity of road scenarios has a significant impact on the descriptiveness of the MS-PCNN network. Based on the zoom-in visual inspection, the decay, ground settlement, occlusion, and moving obstacles (e.g., cyclists) in the Paris-Lille-3D dataset could lead to the false point-wise label assignment. Fig.9 also shows some pedestrian points were misclassified as natural and some points belonging to cars were predicted as barriers. Such unavoidable errors evolving in the process of data preprocessing, such as batch normalization, also conduce to overall accuracy reduction of point cloud segmentation.

Accordingly, based on the same testing protocols, we compared the proposed MS-PCNN model with these existing networks. Table II presents the performance comparison results by calculating the mean IoU ($mIoU$) matrix, which is the mean of IoU across all the object categories. As can be perceived, our method dramatically outperforms both PointNet (38.6% $mIoU$) and PointNet++ (32.0% $mIoU$), which are pioneers that directly consume point clouds using deep learning. In addition, compared to the DGCNN, our method could dynamically update K-nearest neighbors between two adjacent layers of the model and accordingly calculate the sequence of embeddings, resulting in a 17.6% $mIoU$ improvement. Moreover, PointSIFT (62.7% $mIoU$) obtain lower segmentation accuracy than MS-PCNN. Most importantly, for certain types of road objects including signages, bollards, pedestrians and cars, our proposed model can deliver the dominating performance in semantic segmentation. In conclusion, the MS-PCNN model can achieve state-of-the-art point-wise segmentation performance in large-scale urban environments. Meanwhile, the comparative study inspires us to optimize the MS-PCNN model by using more low-level features of point clouds, e.g., RGB and normal vectors.

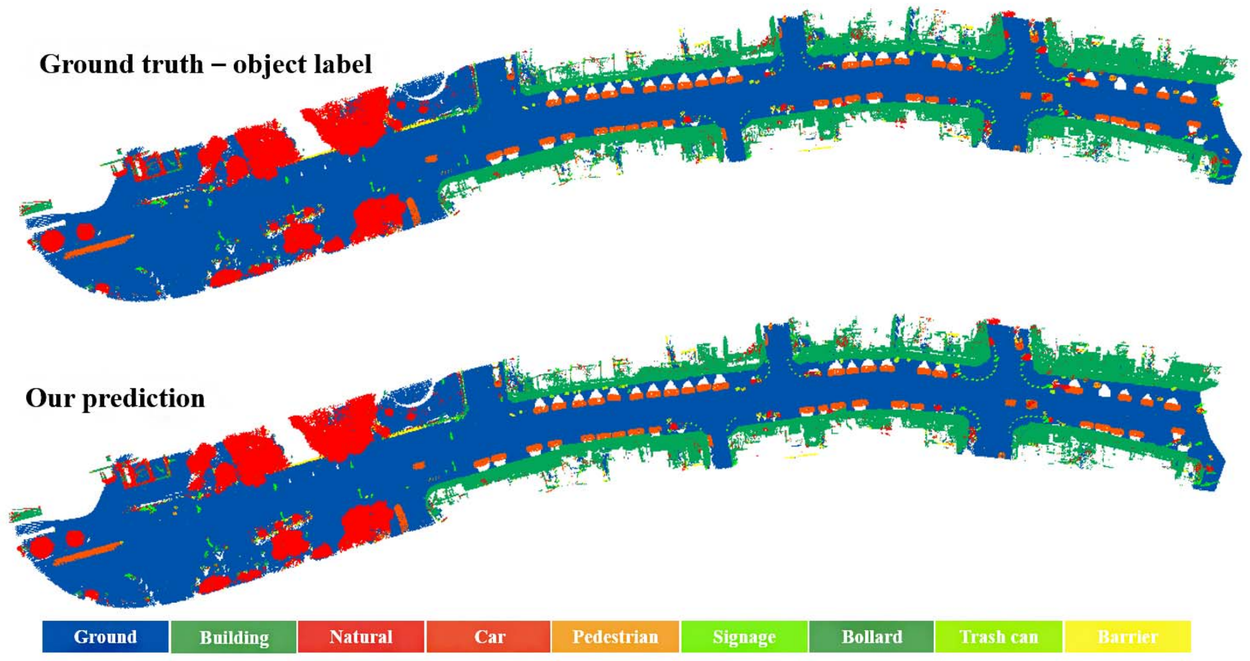


Fig. 8. Point-wise segmentation results by using MS-PCNN network on Paris-Lille-3D dataset.

TABLE II
SEMANTIC SEGMENTATION RESULTS ON LILLE2 BY USING DIFFERENT METHODS

Methods	**Ground**	**Building**	**Signage**	**Bollard**	**Trash Can**	**Barrier**	**Pedestrian**	**Car**	**Natural**	**mIoU(%)**
PointNet [23]	97.3	90.4	22.9	8.7	3.2	2.5	24.3	71.9	26.3	38.6
PointNet++ [55]	96.6	78.6	15.2	4.3	1.2	0	19.3	46.3	26.1	32.0
DGCNN [31]	98.3	93.1	52.9	36.1	19.5	15.0	16.6	88.6	56.5	52.9
PointSIFT [30]	98.4	95.6	51.2	44.8	53.9	31.4	31.3	87.4	70.7	62.7
Ours	98.1	95.4	57.6	64.6	63.0	34.1	57.7	95.2	68.3	70.5

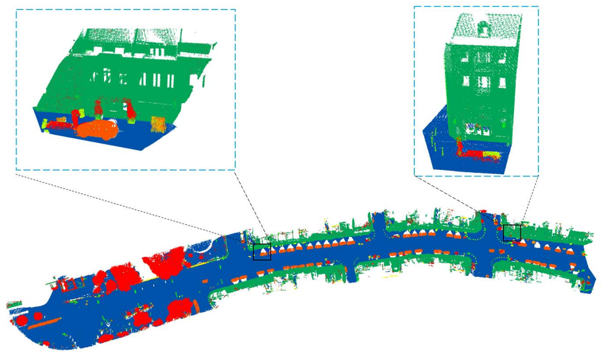


Fig. 9. Two zoom-in views of point-wise segmentation results from Paris-Lille-3D dataset.

C. Segmentation on ScanNet and S3DIS

To evaluate the robustness and performance of MS-PCNN model in indoor environments with high-density point clouds, two open-access point cloud datasets with labels, namely ScanNet and S3DIS datasets, were further utilized. According to the same hyperparameter settings and testing protocols as using Paris-Lille-3D dataset, we determined the optimal combination as $\sigma = 0.10$ and $k = 32$ on both ScanNet and S3DIS test datasets. Additionally, the initial learning rate, batch size, momentum of Adam, dropout rate and epochs were 0.001, 16, 0.9, 0.5, and 200 in ScanNet, and 0.001, 8, 0.9, 0.5, and 250 in S3DIS, respectively, which can achieve the best performance through multiple experiments. Other parameters such as the radius in point density estimation and dimensions of the output channels were experimentally determined through trial and error.

Table III shows the segmentation results on ScanNet by using different point-based deep learning methods. Note that, ScanNet as the pioneer DL-based method proposed for ScanNet dataset achieves 30.6% $mIoU$, which is far from satisfactory in terms of robustness and segmentation accuracy. Although PointNet++ utilized farthest point sampling and multi-scale grouping algorithms to leverage local features from high-density point clouds, it only obtained 38.3% $mIoU$ and 71.4% OA due to the non-uniform distributions and varying point densities in different input scenarios. Our method is superior to both SPLATNet and PointSIFT even though they capture hierarchical and spatially-aware features of input point clouds. Additionally, MS-PCNN

TABLE III
SEMANTIC SEGMENTATION RESULTS ON SCANNET BY USING DIFFERENT METHODS

Methods	**mIoU(%)**	**OA(%)**
ScanNet [61]	30.6	-
PointNet++ [55]	38.3	71.4
SPLATNet [63]	39.3	-
PointSIFT [30]	41.5	86.2
PointCNN [29]	52.2	85.1
Ours	56.8	87.6

TABLE IV
SEMANTIC SEGMENTATION PERFORMANCE ON S3DIS BY USING DIFFERENT METHODS

Methods	**mIoU(%)**	**OA(%)**
PointNet [23]	47.7	78.6
PointNet++ [55]	54.5	81.0
MS+CU [64]	47.8	79.2
G+RCU [64]	49.7	81.1
SegCloud [65]	48.9	-
DGCNN [31]	56.1	84.1
SPGraph [57]	62.1	85.5
PointSIFT [30]	67.2	-
Ours	67.8	87.3

method outperforms PointCNN that ignores edge information among adjacent points in a local region. Compared to other methods, our proposed MS-PCNN model achieves the best performance in the sense of per-object segmentation accuracy.

Furthermore, the semantic segmentation performance on S3DIS by using different deep learning networks is presented in Table IV. As can be perceived, MS+CU, G+RCU, and SegCloud slightly outperform PointNet, while PointNet++ is superior to them. Compared to DGCNN and SPGraph methods, our proposed MS-PCNN method outperforms them by a significant margin. Additionally, our network achieves competitive performance compared with the PointSIFT (i.e., 67.2% for PointSIFT and 67.8% for MS-PCNN) on S3DIS dataset. The experimental results indicate the strengths and robustness of our MS-PCNN method in semantic segmentation particularly in large-scale indoor environments with highly dense point clouds.

D. Segmentation on ShapeNetPart

To evaluate the extensive applicability of the revised PointCONV convolutional operator in small-scale 3D point cloud scenes, ShapeNetPart test datasets were further employed. ShapeNetPart datasets consist of 16,881 3D CAD instances, which are classified into 16 categories and 50 part annotations. The majority of 3D objects are labeled with two to five parts. Moreover, ground truths are labeled on down-sampled points on all categories. Thus, we converted part segmentation task into point-wise segmentation problem.

According to the same hyperparameter settings and testing protocols as using Paris-Lille-3D dataset, we ascertained the optimal combination as $\sigma = 0.10$ and $k = 16$ on ShapeNetPart dataset. Additionally, the initial learning rate, batch size, momentum of Adam, dropout rate and epochs were 0.001, 16, 0.9, 0.5, and 200, respectively, which can achieve the best performance through experimental tests. Table VI shows the per-point segmentation results on ShapeNetPart by using various deep learning methods. It is notable that MS-PCNN obtains an object instance average $mIoU$ of 86.6%, which is on par with the state-of-the-art methods, e.g., PointNet, DGCNN, SpiderCNN, and PointCNN only considering xyz coordinate information of point clouds as inputs. Although 2D rendered images used in SPLATNet, MS-PCNN could provide more accurate and efficient segmentation results by directly consuming point clouds without data conversion. Furthermore, the processing time presented in Table VI indicates the time consuming for both forward and backward propagations through the entire testing dataset, which demonstrates the proposed MS-PCNN can achieve higher computational efficiency and lower time complexity compared with other DL-based methods.

E. Efficiency Evaluation

Although the CRF-based postprocessing could strengthen robustness, it would have a direct influence on the memory consumption and time complexity (i.e., forward and backward propagation) of the whole framework. Most notably, it may affect the segmentation results. To estimate these influences, we tested the MS-PCNN network using a desktop equipped with Intel$^{\circledR}$ i7 8700K CPU @ 4.7GHz and Nvidia GTX 1080 Ti GPU with and without the CRF module. We ran 200 epochs on Paris-Lille-3D dataset. Additionally, we recorded the average processing time and tracked the highest GPU memory size for two different models.

Table V presents the comparison results. It is notable that the model size and GPU-memory usage using the network architecture with CRF module is about 260 MB and 5,015 MB, respectively. The reason is that performing the CRF module could linearly increase the number of parameters of MS-PCNN network. Besides, the mean IoU increases by 2.7% by introducing the CRF module into MS-PCNN architecture, which demonstrates the CRF module is capable of further achieving the point-wise segmentation refinement. Additionally, the time consumption of each forward and backward propagation process in MS-PCNN is over two times than that in the network without the CRF operation. Obviously, the point-wise semantic segmentation performance is greatly improved by employing a CRF post-processing module. Since CRFs are able to directly model spatial structures and capture more inherent geometric characteristics (e.g., connectivity between two adjacent points), the MS-PCNN model can be fine-tuned by formulating the CRF module. Furthermore, the proposed MS-PCNN network can achieve state-of-the-art point-wise segmentation performance in both outdoor and indoor environments with different data distributions and requires less GPU memory usage. By predefining the batch size as 16, the proposed MS-PCNN baseline only consumes 4,824 MB GPU memory space compared to 11,450 MB used in PointSIFT network, which demonstrates the MS-PCNN

TABLE V
PERFORMANCE EVALUATION OF MS-PCNN NETWORK WITH AND WITHOUT THE CRF MODULE ON PARIS-LILLE-3D DATASET

Model	Size (MB)	GPU memory usage (MB)	Time spend of each forward propagation (ms)	Time spend of each backward propagation (ms)	mIoU (%)
MS-PCNN	224.3	4,824	108.06	307.25	67.8
MS-PCNN + CRF	259.9	5,015	235.27	631.61	70.5

TABLE VI
SHAPENET PART SEGMENTATION RESULTS BY USING DIFFERENT METHODS

Methods	mIoU(%)	Processing time(ms)
PointNet [23]	83.7	26.4
PointNet++ [55]	85.1	168.8
SPLATNet [63]	84.6	260.1
DGCNN [31]	85.1	95.6
SpiderCNN [56]	85.3	184.2
PointCNN [29]	86.1	77.3
Ours	86.6	69.0

model is less sensitive to data distributions and computational consumptions.

VI. CONCLUSION

This paper tackles the problems related to 3D point cloud segmentation tasks, particularly in large-scale scenes. Such problems result in computation complexity and robustness reduction when dealing with 3D highly dense point cloud, most notably due to its various point density and irregular data format, as well as occlusion and background interference in the real world. In this paper, we have proposed a novel end-to-end neural network, MS-PCNN, by combining point-wise CNNs with dynamic edge convolutions for 3D point cloud segmentation. The proposed network was evaluated by estimating efficiency and robustness on three publicly accessible LiDAR datasets, including one real urban-scene dataset (i.e., Pairs-Lille-3D), and two indoor-scene datasets (i.e., ScanNet and S3DIS).

In conclusion, our proposed neural network has four main strengths: First, the revised point-wise convolutional filters that can learn spatial relationships and extract geometric information of point clouds in local regions contributing to permutation invariance and translation invariance. Second, MS-PCNN applies a hierarchical PointCONV-based downsampling and DePointCONV-based upsampling architecture in order that more high-level features are extracted in multiple scales. Third, by improving the dynamic graph edge convolution, MS-PCNN can learn edge features between a point and its adjacent neighbors to improve the descriptiveness. Finally, we use a CRF post-processing algorithm to ensure the consistency of point-wise label prediction and refine segmentation results. MS-PCNN model is robust to occlusion and diverse point density for both urban-scene and indoor-scene point clouds. Therefore, this paper demonstrates that MS-PCNN model can provide promising solutions in industrial applications, such as fully autonomous driving. Compared to other point-based networks, e.g., PointSIFT and PointCNN, MS-PCNN is less memory-consuming and time-consuming in both forward and backward propagations, which can considerably save the training time. Additionally, the comparative study certainly indicates that MS-PCNN is superior to other DL-based methods in the testing scenarios in segmentation accuracy and computational complexity. Overall, it is concluded that our proposed neural network can achieve dominating performance in 3D point cloud segmentation under large-scale point cloud scenes more effectively and robustly.

For further research, we are dedicated to improving MS-PCNN performance in several perspectives: using more basic features of point clouds, including RGB, reflectance and normal vector, to enhance the descriptive ability; decreasing the number of model parameters with an appropriate fashion to extend the application of MS-PCNN; and exploiting more robust and effective loss functions to improve the segmentation performance.

REFERENCES

[1] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, “Simultaneous localization and mapping: A survey of current trends in autonomous driving,” *IEEE Trans. Intell. Veh.*, vol. 2, no. 3, pp. 194–220, Sep. 2017.

[2] Y. Yu, J. Li, H. Guan, C. Wang, and C. Wen, “Bag of contextual-visual words for road scene object detection from mobile laser scanning data,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3391–3406, Dec. 2016.

[3] C. Ye, J. Li, H. Jiang, H. Zhao, L. Ma, and M. Chapman, “Semi-automated generation of road transition lines using mobile laser scanning data,” *IEEE Trans. Intell. Transp. Syst.*, to be published.

[4] L. Ma, Y. Li, J. Li, Z. Zhong, and M. A. Chapman, “Generation of horizontally curved driving lines in HD maps using mobile laser scanning point clouds,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 5, pp. 1572–1586, May 2019.

[5] Z. Luo, M. Attari, S. Habibi, and M. Von Mohrenschildt, “Online multiple maneuvering vehicle tracking system based on multi-model smooth variable structure filter,” *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2019.2899051.

[6] C. Wen, X. Sun, J. Li, C. Wang, Y. Guo, and A. Habib, “A deep learning framework for road marking extraction, classification and completion from mobile laser scanning point clouds,” *ISPRS J. Photogram. Remote Sens.*, vol. 147, pp. 178–192, Jan. 2019.

[7] H. Luo *et al.*, “Patch-based semantic labeling of road scene using colorized mobile LiDAR point clouds,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1286–1297, May 2015.

[8] Y. Lin, W. Cheng, D. Zhai, L. Wei, and J. Li, “Toward better boundary preserved supervoxel segmentation for 3D point clouds,” *ISPRS J. Photogramm. Remote Sens.*, vol. 143, pp. 39–47, Sep. 2018.

[9] H. Luo *et al.*, “Semantic labeling of mobile LiDAR point clouds via active learning and higher order MRF,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 7, pp. 3631–3644, Jul. 2018.

[10] A. Nguyen and B. Le, “3D point cloud segmentation: A survey,” in *Proc. 6th IEEE Conf. Robot., Automat. Mechtron.*, Nov. 2013, pp. 225–230.

[11] E. Paquet, M. Rioux, A. Murching, T. Naveen, and A. Tabatabai, “Description of shape information for 2-D and 3-D objects,” *Signal Process., Image Commun.*, vol. 16, nos. 1–2, pp. 103–122, 2000.

[12] T. Funkhouser *et al.*, "A search engine for 3D models," *ACM Trans. Graph.*, vol. 22, no. 1, pp. 83–105, 2003.

[13] W. Wang, R. Yu, Q. Huang, and U. Neumann, "SGPN: Similarity group proposal network for 3D point cloud instance segmentation," in *Proc. IEEE CVPR*, Jun. 2018, pp. 2569–2578.

[14] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 433–449, May 1999.

[15] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proc. ECCV*. Berlin, Germany: Springer, 2010, pp. 356–369.

[16] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. 2nd IEEE Conf. Robot. Autom. Mechatronics*, May 2009, pp. 3212–3217.

[17] T. Masuda, "Log-polar height maps for multiple range image registration," *Comput. Vis. Image Understand.*, vol. 113, no. 11, pp. 1158–1169, 2009.

[18] X. X. Zhu *et al.*, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017.

[19] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE IROS*, Sep./Oct. 2015, pp. 922–928.

[20] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE CVPR*, Jul. 2017, pp. 1907–1915.

[21] Y. Wang, Z. Xie, K. Xu, Y. Dou, and Y. Lei, "An efficient and effective convolutional auto-encoder extreme learning machine network for 3D feature learning," *Neurocomputing*, vol. 174, pp. 988–998, Jan. 2016.

[22] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE CVPR*, Jul. 2017, pp. 3693–3702.

[23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE CVPR*, Jul. 2017, pp. 652–660.

[24] L. Ma, Y. Li, J. Li, C. Wang, R. Wang, and M. A. Chapman, "Mobile laser scanned point-clouds for road object detection and extraction: A review," *Remote Sens.*, vol. 10, no. 10, p. 1531, 2018.

[25] E. Che, J. Jung, and M. J. Olsen, "Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review," *Sensers*, vol. 19, no. 4, pp. 810-1–810-42, Feb. 2019.

[26] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Trans. Graph.*, vol. 21, no. 4, pp. 807–832, 2002.

[27] Z. Wu *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE CVPR*, Jun. 2015, pp. 1912–1920.

[28] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE CVPR*, Jun. 2016, pp. 5648–5656.

[29] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 820–830.

[30] M. Jiang, Y. Wu, and C. Lu, "PointSIFT: A sift-like network module for 3D point cloud semantic segmentation," 2018, *arXiv:1807.00652*. [Online]. Available: https://arxiv.org/abs/1807.00652

[31] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," 2018, *arXiv:1801.07829*. [Online]. Available: https://arxiv.org/abs/1801.07829

[32] Z. Dong, B. Yang, F. Liang, R. Huang, and S. Scherer, "Hierarchical registration of unordered TLS point clouds based on binary shape context descriptor," *ISPRS J. Photogramm. Remote Sens.*, vol. 144, pp. 61–79, Oct. 2018.

[33] Z. Luo, J. Li, Z. Xiao, Z. G. Mou, X. Cai, and C. Wang, "Learning high-level features by fusing multi-view representation of MLS point clouds for 3D object recognition in road environments," *ISPRS J. Photogramm. Remote Sens.*, vol. 150, pp. 44–58, Apr. 2019.

[34] Z. Shen, X. Ma, and Y. Li, "A hybrid 3D descriptor with global structural frames and local signatures of histograms," *IEEE Access*, vol. 6, pp. 39261–39272, 2018.

[35] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *Proc. ECCV*. Berlin, Germany: Springer, 2004, pp. 224–237.

[36] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *Comput. Graph. Forum*, vol. 28, no. 5, pp. 1383–1392, 2009.

[37] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool, "Hough transform and 3D SURF for robust three dimensional classification," in *Proc. ECCV*. Berlin, Germany: Springer, 2010, pp. 589–602.

[38] S. Salti, F. Tombari, and L. Di Stefano, "SHOT: Unique signatures of histograms for surface and texture description," *Comput. Vis. Image Understand.*, vol. 125, pp. 251–264, Aug. 2014.

[39] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3D local surface description and object recognition," *Int. J. Comput. Vis.*, vol. 105, no. 1, pp. 63–86, 2013.

[40] Z. Wang and K. Jia, "Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection," 2019, *arXiv:1903.01864*. [Online]. Available: https://arxiv.org/abs/1903.01864

[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[42] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[43] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. IEEE CVPR*, Jun. 2015, pp. 3128–3137.

[44] Z. Luo, M. V. Mohrenschildt, and S. Habibi, "A probability occupancy grid based approach for real-time LiDAR ground segmentation," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2019.2900548.

[45] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proc. IEEE CVPR*, Jul. 2017, pp. 3577–3586.

[46] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs," in *Proc. IEEE ICCV*, Oct. 2017, pp. 2088–2096.

[47] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3D point cloud models," in *Proc. IEEE ICCV*, Oct. 2017, pp. 863–872.

[48] W. Zeng and T. Gevers, "3DContextNet: Kd tree guided hierarchical learning of point clouds using local and global contextual cues," in *Proc. ECCV*, 2018.

[49] Z. Wang and F. Lu, "VoxSegNet: Volumetric CNNs for semantic part segmentation of 3D shapes," *IEEE Trans. Vis. Comput. Graphics*, to be published.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[51] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE ICCV*, Oct. 2017, pp. 2961–2969.

[52] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE ICCV*, Dec. 2015, pp. 945–953.

[53] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. J. Latecki, "GIFT: A real-time and scalable 3D shape search engine," in *Proc. IEEE CVPR*, Jun. 2016, pp. 5023–5032.

[54] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE CVPR*, Jun. 2019, pp. 770–779.

[55] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5099–5108.

[56] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas, "GSPN: Generative shape proposal network for 3D instance segmentation in point cloud," in *Proc. IEEE CVPR*, Jun. 2019, pp. 3947–3956.

[57] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. ECCV*, 2018, pp. 87–102.

[58] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," 2018, *arXiv:1811.07246*. [Online]. Available: https://arxiv.org/abs/1811.07246

[59] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[60] X. Roynard, J.-E. Deschaud, and F. Goulette, "Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification," *Int. J. Robot Res.*, vol. 37, no. 6, pp. 545–557, 2018.

[61] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE CVPR*, Jul. 2017, pp. 5828–5839.

[62] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-semantic data for indoor scene understanding," 2017, *arXiv:1702.01105*. [Online]. Available: https://arxiv.org/abs/1702.01105

[63] H. Su *et al.*, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE CVPR*, Jun. 2018, pp. 2530–2539.

[64] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3D semantic segmentation of point clouds," in *Proc. IEEE ICCV*, Oct. 2017, pp. 716–724.

[65] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "SEGCloud: Semantic segmentation of 3D point clouds," in *Proc. Int. Conf. 3D Vis.*, 2017, pp. 537–547.

Lingfei Ma (S'18) received the B.Sc. degree in giscience from the China University of Geoscience, Beijing, China, in 2015, and the B.Sc. and M.Sc. degrees in geomatics specializing in remote sensing from the University of Waterloo, Canada, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree in photogrammetry and remote sensing with the Mobile Sensing and Geodata Science Laboratory.

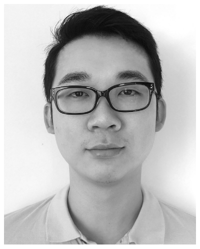
His research interests include autonomous driving, high-definition mapping, mobile laser scanning, intelligent processing of point clouds, 3D scene modeling, and deep learning.

Ying Li received the B.Eng. degree in geomatics engineering from the Hefei University of Technology, China, in 2014, and the M.Sc. degree in remote sensing from Wuhan University, China, in 2017. She is currently pursuing the Ph.D. degree with the Mobile Sensing and Geodata Science Laboratory, Department of Geography and Environmental Management, University of Waterloo, ON, Canada.

Her research interests include autonomous driving, mobile laser scanning, intelligent processing of point clouds, geometric and semantic modeling, and augmented reality.

Jonathan Li (M'00–SM'11) received the Ph.D. degree in geomatics engineering from the University of Cape Town, Cape Town, South Africa, in 2000. He is currently a Professor and the Head of the Mobile Sensing and Geodata Science Group, Department of Geography and Environmental Management, cross-appointed with the Department of Systems Design Engineering, University of Waterloo, Canada. He is also a Founding Member of the Waterloo Artificial Intelligence Institute. He is also with the Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Informatics, Xiamen University, China.

He has coauthored more than 400 publications, more than 200 of which were published in refereed journals, including IEEE-TGRS, IEEE-TITS, IEEE-JSTARS, IEEE-GRSL, ISPRS-JPRS, and RSE. His research interests include AI-based information extraction from mobile LiDAR point clouds and Earth observation images. He was a recipient of the 2019 Outstanding Achievement Award for recognizing his significant contributions to mobile mapping technology. He is a Chair of the ISPRS WG I/2 on LiDAR, Air- and Space-borne Optical Sensing from 2016 to 2020 and the ICA Commission on Sensor-driven Mapping from 2015 to 2019 followed by 2019 to 2023. He is an Associate Editor of the IEEE-TITS, IEEE-JSTARS, and *Canadian Journal of Remote Sensing*.

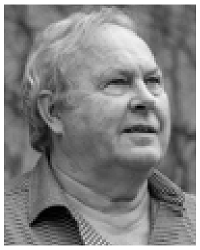


Weikai Tan received the B.Sc. degree from Nanjing University, Nanjing, China, in 2014, and the B.Sc. and M.Sc. degrees in geomatics from the University of Waterloo, Waterloo, ON, Canada, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in geography with the Mobile Sensing and Geodata Science Laboratory, Department of Geography and Environmental Management, University of Waterloo.

His research interests include SAR and LiDAR data analysis, machine learning, and environmental monitoring.

Yongtao Yu (M'16) received the Ph.D. degree in computer science and technology from Xiamen University, Xiamen, China, in 2015. He is currently an Assistant Professor with the Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huaian, China.

He has authored or coauthored more than 30 research articles published in refereed journals and proceedings, including the IEEE-TGRS, IEEE-TITS, JSTARS, IEEE-GRSL, and ISPRS-JPRS. His research interests include intelligent transportation systems, computer vision, deep learning, intelligent interpretation of 3-D point clouds, and remotely sensed imagery.



Michael A. Chapman received the Ph.D. degree in photogrammetry from Laval University, Quebec, QC, Canada. He was a Professor with the Department of Geomatics Engineering, University of Calgary, Canada, for 18 years prior joining Ryerson University in 1999. He is currently a Professor with the Department of Civil Engineering, Ryerson University, Toronto, Canada.

He has authored or coauthored more than 200 technical articles, including those in top remote sensing journals such as *ISPRS Journal of Photogrammetry and Remote Sensing* and the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING. His research interests include algorithms and processing methodologies for airborne sensors using GNSS/IMU, geometric processing of digital imagery in industrial environments, terrestrial imaging systems for transportation infrastructure mapping, mobile laser scanning, and algorithms and processing strategies for biometrology applications.