LCE-NET: Contour Extraction for Large-Scale 3-D Point Clouds

Yu Zang, *Member, IEEE*, Binjie Chen, Yunzhou Xia, Hanyun Guo, Yunuo Yang, Weiquan Liu, *Member, IEEE*, Cheng Wang, *Senior Member, IEEE*, and Jonathan Li, *Fellow, IEEE*

***Abstract*— The contours, one of the most significant human perceptual features, have a significant impact on point cloud processing. In urban scenes, contour extraction is quite challenging due to the enormous number of unstructured and irregular points (typically greater than 10^7 points). In this article, we propose a large-scale 3-D point cloud contour extraction network (LCE-NET) to generate contours consistent with human perception of outdoor scenes. To our knowledge, it is the first time that an end-to-end learning-based framework has been proposed for contour extraction on point cloud at this scale. The proposed LCE-Net is essentially a two-phase system. In the first phase, potential vertexes are detected from the input point cloud by the vertex detection module; then, in the second phase, a designed overcomplete line proposal set is generated, and invalid line segments are further suppressed by the line proposal discrimination module. The two phases are jointly trained by a uniform loss function to promote the information interchange, thus leading to extraction results with satisfied accurate and false alarm ratings. Since there is hardly any available dataset with labeled contours for the large-scale outdoor scene, we open-sourced SemanticLine, the first dataset for large-scale point clouds with labeled contour information, based on reannotation of previous mapping-level point cloud dataset semantic3D. The experimental results demonstrate that LCE-NET can effectively extract parametric contour lines from large-scale point clouds of urban scenes. In addition, it outperforms the state-of-the-art approaches. The code will be open-sourced on GitHub soon.**

***Index Terms*— Contour extraction, point cloud.**

I. Introduction

With the rapid development of light detection and ranging (LiDAR) [1], structure from motion (SfM) [2], and some other technologies, the acquisition of point clouds has become extremely easy. However, raw point clouds are inconvenient to use directly due to their irregular and unstructured format in most applications.

To this end, a number of previous works [3], [4] focus on point-based feature extraction for point clouds. After the great success of PointNet [5] as the first point-based deep learning point cloud processing method, the mainstream method of point-based point cloud feature extraction gradually shifted from traditional machine learning methods to deep learning methods. The point-based learning methods quickly swept other areas of point cloud processing and the network structure became more sophisticated and specialized, such as object completion [6] and place recognition and scene localization [7].

Contours are widely used in graphics and mapping applications since they are one of the most essential features of human and intermediate features of point clouds, but still challenging to extract, especially for large-scale outdoor point clouds. As pointed out by Hackel et al. [8], in real point clouds, contours are almost impossible to extract by low-level rules because of occlusions and incomplete data. The contours can briefly describe the main structure of an object, are one of the important features of an object, and are also the basic features of an object that humans can directly perceive. In the field of computer vision, contours are widely used in vehicle positioning [9], road network extraction [10], 3-D reconstruction [11], shape matching [12], object recognition [13], and other applications. Converting 3-D point clouds to contours can achieve denoising and compression of point cloud data while retaining the main features of the object. It is an ideal and widely used intermediate feature.

Although humans can easily "see" contours in point clouds, it is quite challenge to achieve contour extraction for large-scale 3-D point clouds. First, the definition of contours is ambiguous and difficult to formalize. The contours perceived by humans may appear where there is a sudden change in surface curvature or it may be related to factors such as the length of the line structure and local direction changes. Second, due to the acquisition method of the 3-D point cloud itself, the point cloud data have the characteristics of irregularity, incomplete objects, and uneven density. Handcrafted features [14], [15] were used most commonly in the early days of 3-D point cloud contour extraction, but these methods only extracted line-like features rather than actual contours. As a result of such predefined rules, they have a limited ability

Manuscript received 19 May 2023; revised 2 August 2023; accepted 11 September 2023. Date of publication 15 September 2023; date of current version 6 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61971363, in part by the China Postdoctoral Science Foundation under Grant 2021M690094, and in part by the FuXiaQuan National Independent Innovation Demonstration Zone Collaborative Innovation Platform under Grant 3502ZCQXT2021003. *(Corresponding author: Weiquan Liu.)*

Yu Zang, Binjie Chen, Yunzhou Xia, Hanyun Guo, Yunuo Yang, Weiquan Liu, and Cheng Wang are with the Fujian Key Laboratory of Sensing and Computing for Smart Cities and the Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: zangyu7@126.com; chenbinjie@stu.xmu.edu.cn; 23020221154127@stu.xmu.edu.cn; hyguo@stu.xmu.edu.cn; yangyunuo555@stu.xmu.edu.cn; wqliu@xmu.edu.cn; cwang@xmu.edu.cn).

Jonathan Li is with the GeoSTARS Laboratory, Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: junli@uwaterloo.ca).

Digital Object Identifier 10.1109/TGRS.2023.3315677

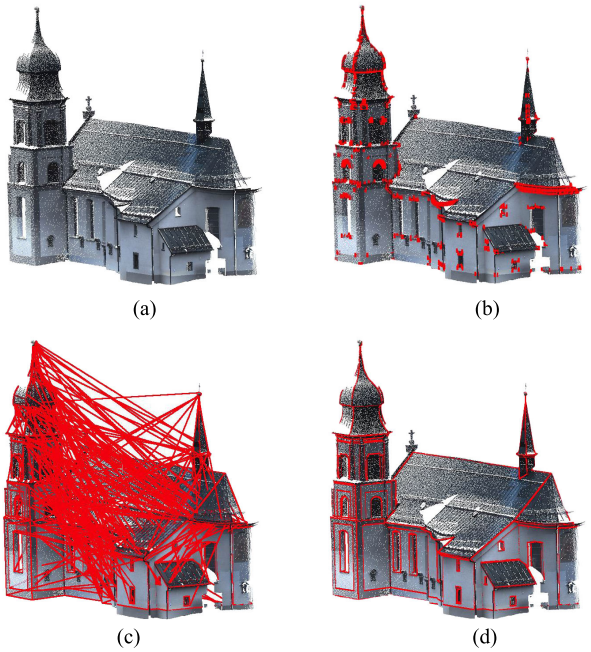


Fig. 1. Given a 3-D point cloud, our proposed LCE-NET detects vertices first. Then, line proposals are generated by connecting every two vertices. The final output contours are produced by suppressing invalid line proposals. (a) Input point cloud. (b) Vertices. (c) Line proposals. (d) Output contours.

to generalize. Recently, Hackel et al. [8] employed machine learning to extract real contours using a learning-based method for the first time. Inspired by the success of deep neural networks in point clouds feature extraction [5], many recent approaches [16], [17] applied deep learning to contour extraction. However, these deep learning-based approaches mainly focus on object-level point clouds, which feed the entire point clouds into the network directly, resulting in a significant amount of memory consumption. Since outdoor point clouds typically consist of greater than 10^7 points and the scene is more complex. It can hardly obtain a good detection accuracy by directly processing the complete point cloud, and it is easy to generate a large number of false positive contours. Hence, it is not feasible for previous deep learning-based approaches to effectively handle them.

In this article, we present an end-to-end deep learning-based large-scale 3-D point cloud contour extraction network (LCE-NET). Our framework avoids feeding the entire point cloud into the deep learning network for processing large-scale point clouds. In our LCE-NET, contour extraction is treated as a two-stage proposal and discrimination process. In the first stage, the input points are divided into small voxels. Then, the network predicts the potential vertex in each voxel. In the second stage, the network produces the final output by forming line segments from every two vertices and suppressing invalid line segments. The pipeline of our proposed LCE-NET is shown in Fig. 1.

The main contribution of this article can be described as follows.

1) We present a novel end-to-end contour extraction framework, which is the first deep learning-based method for large-scale point cloud contour extraction. Our framework avoids inputting the complete point cloud scene into the neural network directly and realizes the contour extraction for large-scale point clouds through a deep learning framework.
2) We propose a two-stage proposal and a discrimination scheme, to make our framework able to generate accurate parametric contour lines. The proposed method can extract concise parametric contours in line with human perception from point clouds of different scenes and different degrees of sparsity.
3) The experimental results on semantic3D.net show that our LCE-NET can extract contours from large-scale point clouds effectively and outperforms the state-of-the-art methods with higher generalization and better performance.

II. Related Works

Contour extraction is an important task that has been studied for a long time, especially for 2-D images. Traditional approaches mostly rely on priors for contour extraction of images [18], [19], [20], [21], [22]. Recently, some learning-based methods provided more precise results [23], [24], [25], [26], [27]. Compared with 2-D images, contour extraction on 3-D point clouds is more complicated, and most of these works can be divided into three categories: image-based methods, plane-based methods, and point-based methods.

A. Image-Based Methods

The image-based methods make full use of the relatively mature 2-D image edge extraction method. After the edge is extracted in two dimensions, it is backprojected back to 3-D to obtain the 3-D edge. Taylor and Kriegman [28] defined the objective function according to the total square distance of the projection of the edge line segment in the image plane to the reconstructed line segment, extracted the line structure on the 2-D image by optimizing the objective function, and reconstructed the 3-D wireframe. Jain et al. [29] obtained better wireframe detection results than local geometric constraints by constraining the global topological features of the wireframe. At the same time, by not requiring clear line segment correspondence in the 3-D reconstruction stage, it was avoided due to other views. Line segments are missing due to occlusion. Ceylan et al. [11] proposed a method for 3-D reconstruction of urban buildings based on symmetric priors, performing edge detection on 2-D images and using multiview stereo matching on 2-D line segments to obtain 3-D wireframes. Based on the 3-D point cloud data, Lin et al. [15] converted the point cloud into a set of 2-D grayscale images and then used the LSD algorithm [30] to extract the wireframe on the 2-D image and backproject the wireframe back to the point cloud and after a series of processing to generate the final 3-D wireframe. Lu et al. [31] segmented the 3-D point cloud into planes through region growing and region merging, then obtained the 2-D contour line segment through 2-D contour extraction and least squares fitting and backprojected to 3-D, and finally removed it through postprocessing redundant line segments and merge adjacent line segments to get the final 3-D

contour line segment. Meng et al. [32] used a deep learning network to extract 2-D line segments and planes from a series of RGBD images and then generated 3-D wireframes based on the consistency of 2-D and 3-D as well as line segments and planes.

B. Plane-Based Methods

The plane-based methods regard the contour segment as the intersection of two 3-D planes and perform 3-D edge extraction by dividing the 3-D model into a set of facets and finding the intersection. Ohtake et al. [33] extracted ridge-valley lines defined by the first- and second-order curvatures on dense triangular mesh surfaces and realized high-order surface derivatives by combining finite differences and multilevel implicit surface fitting methods approximate calculation. Sampath and Shan [34] divided the point cloud into plane points and nonplane points through feature analysis, calculated the surface normal, then clustered them into planes according to the surface normal of the plane points, and finally obtained the contour line segment according to the plane intersection line. Borges et al. [9] calculated the eigenvalues of the covariance matrix in the local point cloud to determine the local shape to extract the facets, merged the planes according to the surface normal of the facets, and finally obtained the 3-D by detecting the intersection of the planes edge segment. In addition, the modified method also extracts the edges that are not on the plane intersection through the discontinuity of the point cloud depth. Moghadam et al. [35] also adopted a similar method, calculating the local surface normal through the characteristic analysis of the second-order moment matrix, and merging the planes through the region growing algorithm, and finally calculating the intersection line of the plane containing a sufficient number of points to get 3-D edges. Lin et al. [36] first roughly divided the point cloud into facets through region growth, then further determined the boundaries of the facets through local k-means clustering, and finally detected the boundary points on the facets and extracted line segments. Xu and Stilla [37] used supervoxel structure and global graph-based optimization to segment the point cloud into facets and then extracted the boundary points of the facets through the α-shape algorithm and clustered them into line segments.

C. Point-Based Methods

Point-based methods detect boundary points in point clouds directly and then fit for the 3-D line segment. Weber et al. [14] first computed a Gauss map clustering on local neighborhoods and then applied a more precise iterative selection to the remaining feature candidates. Ioannou et al. [38] proposed a multiscale operator called difference of normals (DoN). Bazazian et al. [39] detected edge features by analyzing the eigenvalues of the covariance matrix that each point's k-nearest neighbors defined. Himeur et al. [40] used a scale space matrix to encode local geometric features around each point and detect edge points through a lightweight neural network.

The aforementioned 3-D contour extraction methods are mostly feature-based, but as mentioned by Hackel et al. [8], the higher complexity of point clouds makes it infeasible to design and tune such methods. To this end, Hackel et al. [8] proposed a learning-based 3-D contour extraction approach, which first computed pointwise contour scores with a binary classifier, then linked points with high contour score into a graph of candidate contours, and finally selected an optimal subset of those candidates via Markov random field. Yu et al. [16] presented the first deep learning-based edge-aware technique. Zhang et al. [41] refined the extracted contours via a generative adversarial network. Meng et al. [32] presented a multimodal line segment classification technique for extracting 3-D wireframes. Wang et al. [17] applied a deep neural network to extract parameter inference of feature edges over 3-D point clouds.

III. Method

In this article, we propose a deep learning-based 3-D point cloud contour extraction framework, LCE-NET, which can directly extract contour from raw large-scale outdoor point clouds. As shown in Fig. 2, our network mainly consists of two modules, vertex detection module (see Section III-A1) and line proposal discrimination module (see Section III-A2).

A. Network Architecture

Since all point clouds of outdoor scenes are usually greater than 10 million points, most networks are not able to process these points on such a significant scale. Therefore, the LCE-NET is able to extract contours from large point clouds by avoiding directly feeding them into the network. More specifically, our LCE-NET divides the points in a point cloud into small voxels. Next, the vertex detection module detects the vertices in each voxel. Finally, line proposals formed by every two vertices are filtered using the line proposal discrimination module to generate the final output.

1) Vertex Detection Module: Our vertex detection module is designed to accurately detect potential vertex in each voxel. Each voxel of the input point clouds is fed separately into the vertex detection module after voxelization. As shown in Fig. 2(b), the vertex detection module has two branches that have a similar architecture, and the upper branch predicts the probability that the voxel contains a vertex, while the lower branch predicts the position of the vertex in the voxel.

A relatively complex network architecture can be used to improve prediction accuracy without taking too long as the large-scale point clouds are already divided into small voxels. Vertex detection is accomplished using feature extraction encoder and decoder. The feature extraction network structure includes a four-layer EdgeConv [42] module for extracting local structural features of voxels, and the output feature vectors of the four-layer EdgeConv [42] module are aggregated to generate feature vectors of voxels after multilayer perceptron and pooling. The voxel feature vectors predicted by the two branches output voxel classification predictions and vertex position predictions through different multilayer perceptrons. Among them, the voxel classification prediction outputs whether the voxel contains vertices in a binary classification manner, and the vertex position prediction outputs the coordinates of the vertices in the voxel.

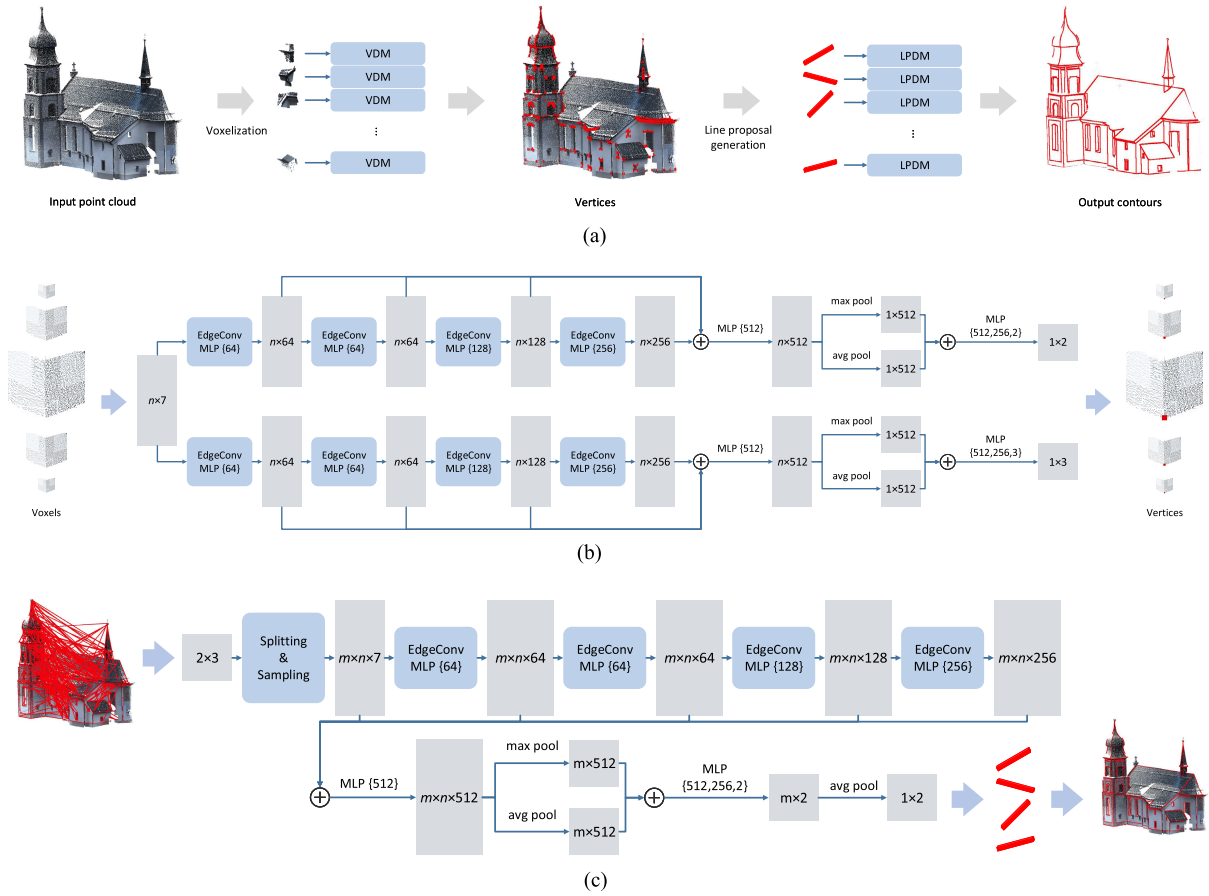


Fig. 2. Network architecture. In the first stage, the input points are divided into small voxels. Then, the network predicts the potential vertex in each voxel using the vertex detection module. In the second stage, the network produces the final output by forming line segments from every two vertices and suppressing invalid line segments using the line proposal discrimination module. (a) Framework. (b) Vertex detection module. (c) Line proposal discrimination module.

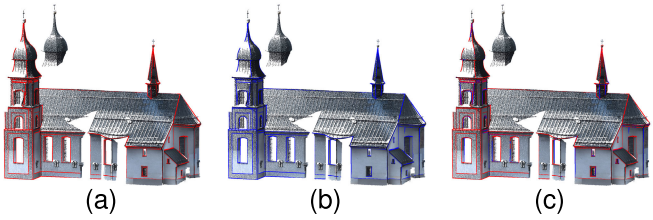


Fig. 3. Ground-truth contour after voxelization. (a)–(c) Ground-truth contour, ground-truth contour after voxelization, and the comparison between them, respectively. The red lines indicate the ground-truth contour and the blue lines indicate the ground-truth contour after voxelization.

Here, the vertex detection in the first stage contains an assumption that each voxel contains only one vertex. Since the vertex prediction is used to generate subsequent candidate contour line segments, if the size of the voxel is small enough, the distance between multiple vertices in the voxel is also small enough, which is regarded as the accuracy of a vertex to the subsequently generated contour line segment. The impact is negligible, as shown in Fig. 3. Also, if the voxel size is too small, it will also lead to too few point clouds in the voxel, and the amount of information contained is not enough for the network to extract features and accurately detect vertices. Through extensive experiments, the final voxel side length is determined to be 0.5 m.

2) Line Proposal Discrimination Module: After detecting the vertices of the point clouds, we connect every two vertices to form line proposals (see Section III-A4) and suppress invalid line proposals through the line proposal discrimination module. The line proposal contains two vertices and all these neighbors. The line proposal discrimination module predicts the likelihood that a line proposal is valid or invalid by splitting each proposal into segments and feeding them into it separately.

As shown in Fig. 2(c), we treat line proposal discrimination as a binary classification task. The input of the module is sampled neighbors of each segment of each line proposal. Similar to the vertex detection module, the feature extraction structure of the network includes a four-layer EdgeConv [42] module for extracting features of different scales, where m represents the number of unit line segments split by the line proposal and n represents the size of the neighborhood of the unit line segments. After aggregation of the four sets of feature vectors, the feature vector of the line proposal segment is generated through the multilayer perceptron and the pooling layer. The feature vector of the line proposal segment is generated by the multilayer perceptron to predict the contour line segment of the binary classification, and the final module uses the average pooling layer to generate the final line proposal segment discriminant prediction. The set

of all line proposals judged to be true is the point cloud parameterized contour prediction output by our method.

3) Voxelization: Voxelization is performed before the input point clouds are fed into the vertex detection module, allowing us to employ a more complex network architecture to obtain predicted vertices accurately. Using too small voxels will result in the network being unable to predict vertices correctly, while using too large voxels will result in multiple potential vertices within one voxel, affecting contour extraction accuracy. After extensive experiments, we divide point clouds into voxels with a side length of 0.5 m.

4) Line Proposal Generation: As mentioned before, every two vertices generate a line proposal. In this context, a line proposal refers to a set of points that includes the two vertices of the line and the points nearby. More formally, the definition of line proposal is shown as follows:

$$\mathrm{LP}_{i,j} = \left\{ p \,\middle|\, \begin{array}{l} \|p - p^*\|^2 \le r \\ p \in P \\ p^* = kp_i + (1-k)p_j \\ k \in [0,1] \end{array} \right\} \tag{1}$$

where P is the input point cloud and p_i and p_j are the two vertices that form the line proposal.

B. Loss Functions

Our LCE-NET is trained by two different loss functions.

1) Vertex Detection Module: There are two branches in the vertex detection module. The first branch predicts the probability that a voxel contains a vertex. We employ a cross-entropy loss in this branch since it is a binary classification task. Our second branch minimizes the distance between the predicted vertex and the actual vertex by using the mean squared error loss. The loss function of the vertex detection module is defined as follows:

$$L_{\text{vertex}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{CE}\left(V_i, \hat{V}_i\right) + \hat{V}_i \,\mathrm{mse}\left(P_i, \hat{P}_i\right)\right) \tag{2}$$

where N represents the number of voxels, V_i and $\hat{V}_i$ are the predicted probability and ground truth of the ith voxel contains a vertex, P_i and $\hat{P}_i$ are the predicted and ground truth of the coordinate of the vertex in the ith voxel, respectively, and $\mathrm{CE}(\cdot)$ and $\mathrm{mse}(\cdot)$ are the cross-entropy loss and mean squared error loss, respectively.

2) Line Proposal Discrimination Module: In contrast to the previous module, line proposal discrimination can also be regarded as a binary classification task. However, as all line proposals are generated by connecting every two vertices directly, the number of negative samples is much larger than the number of positive samples, so it cannot be used as a loss function based on cross entropy. Here, we use a focal loss [43], defined as follows:

$$L_{\text{line}} = \frac{1}{M}\sum_{i=1}^{M}\mathrm{FL}\left(D_i, \hat{D}_i\right) \tag{3}$$

where M represents the number of line proposals, D_i and $\hat{D}_i$ are the predicted probability and ground truth of the ith line proposal, respectively, and $\mathrm{FL}(\cdot)$ is focal loss.

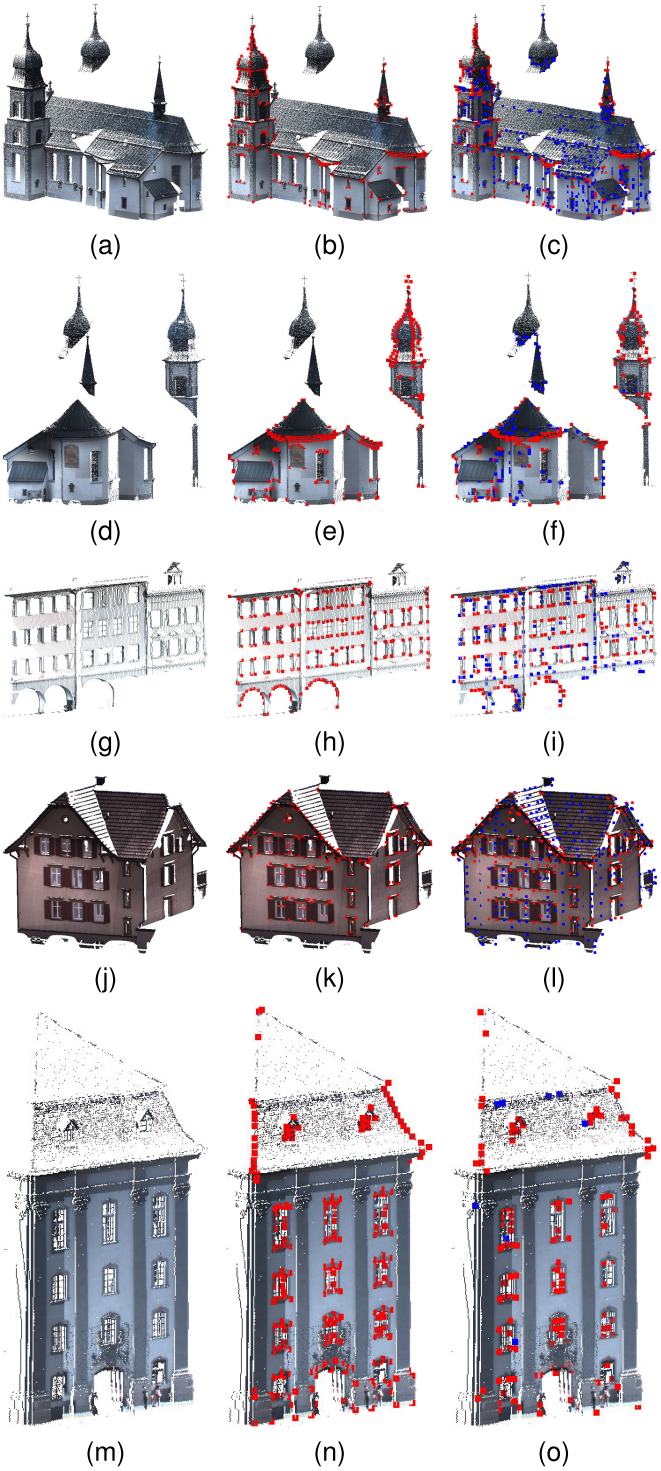


Fig. 4. Evaluation of the vertex detection module. From left to right are the input point clouds, ground truth, and our results for each set. (a), (d), (g), (j), and (m) Input point clouds. (b), (e), (h), (k), and (n) Ground truth. (c), (f), (i), (l), and (o) Our results. The red points indicate the correct vertex and the blue points indicate the wrong ones.

Thus, the joint loss function of the proposed LCE-NET is defined as follows:

$$L = L_{\text{vertex}} + L_{\text{line}}. \tag{4}$$

IV. Experiments

To evaluate the proposed framework, we conduct experiments on semantic3D.net [47] with contour ground-truth

TABLE I

EVALUATION OF THE VERTEX DETECTION MODULE ON SEMANTIC3D WITH CONTOUR GROUND-TRUTH REANNOTATED

Point Cloud	Number of voxels	Precision	Recall	mDis
bildstein_station1_0	6310	0.9046	0.8529	0.1474
bildstein_station3_0	3039	0.9217	0.7215	0.1456
marketplacefeldkirch_station4_2	1620	0.8907	0.8326	0.1475
sg27_station9_10	1592	0.7764	0.9751	0.1560
stgallencathedral_station3_9	2106	0.9492	0.6187	0.1502
All	25988	0.8624	0.8754	0.1487

TABLE II

EVALUATION OF THE KEYPOINT DETECTION NETWORK USIP [44] ON SEMANTIC3D WITH CONTOUR GROUND-TRUTH REANNOTATED

Point Cloud	Number of voxels	Precision	Recall
bildstein_station1_0	6310	0.0845	0.0754
bildstein_station3_0	3039	0.1189	0.1456
marketplacefeldkirch_station4_2	1620	0.1768	0.3073
sg27_station9_10	1592	0.3378	0.3518
stgallencathedral_station3_9	2106	0.1467	0.2109
All	25988	0.1971	0.2612

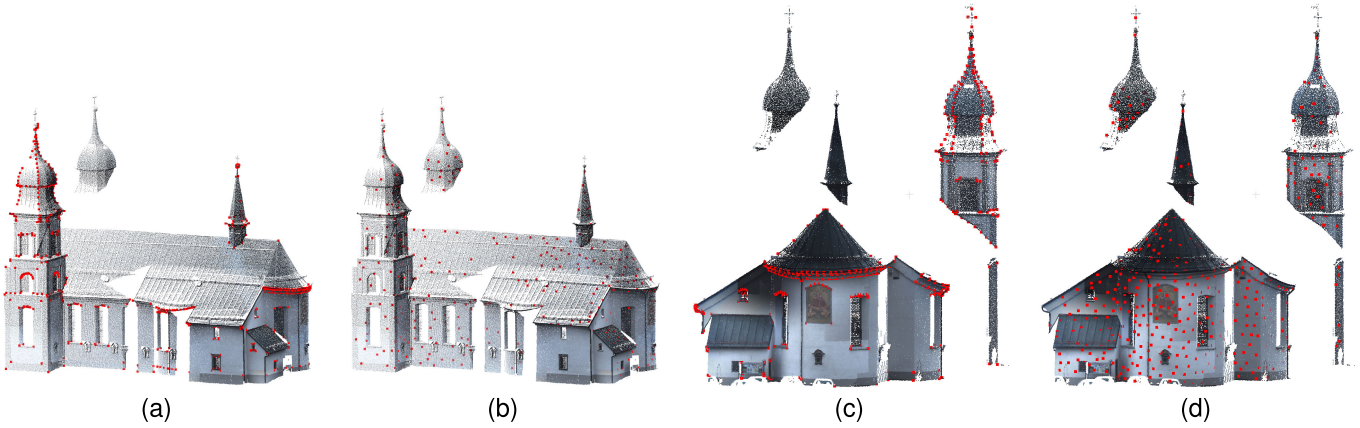


Fig. 5. Evaluation of the keypoint detection network USIP [44]. (a) Ground truth points of point cloud bildstein_station1_0, (b) keypoints detected by USIP [44] of point cloud bildstein_station1_0, (c) ground truth points of point cloud bildstein_station3_0, (d) keypoints detected by USIP [44] of point cloud bildstein_station3_0. The red points indicate the ground truth points and the keypoints detected by USIP [44].

reannotation. In the first stage, we conduct an independent evaluation of each module of the proposed LCE-NET. Furthermore, we compare our approach not only with traditional methods but also with learning-based methods in a separate set of experiments to comprehensively assess its performance.

A. Dataset and Implementation Details

1) Dataset: We evaluate our LCE-NET on semantic3D.net [47] with contour ground-truth reannotated, which contains 40 large-scale outdoor scene point clouds. The contour ground truth was labeled with 3-D lines represented as parameters, and we extract the endpoints of each 3-D line as the vertex ground truth for the training and evaluation of the vertex detection module. The training phase of the study was conducted using 28 out of the 40 available point clouds. The remaining 12 point clouds were reserved for the evaluation phase. It is important to note that this approach was used in all of our experiments conducted during this article.

2) Implementation Details: In this work, we use PyTorch to build our LCE-NET on a Ubuntu PC with two Intel Xeon Silver 4216 CPUs and one Nvidia GeForce RTX 3090 GPU. As for the network and training settings, the Adam optimizer with initial learning rate of 0.0001 is used for training, and the learning rate is reduced by half every 20 epochs. The total training epochs are set to 200.

B. Analyses of the Network Architecture

We treat contour extraction as a two-stage proposal and discrimination task. Our proposed model LCE-NET comprises two distinct modules that together accomplish the detection of vertices and the differentiation of line proposals. This section includes five sets of experiments that aim to fully evaluate the efficacy, necessity, intrinsic connectivity, and robustness of these two modules within the LCE-NET framework.

1) Vertex Detection Module: The results of the first group of experiments evaluating the vertex detection module are shown in Fig. 4. The first column of Fig. 4 is the input point cloud, and the second column is the ground truth. The vertex detection module produces the results that are presented in the final column. The correct vertex is denoted by red points, while the incorrect one is denoted by blue points. It can be seen from Fig. 4(c) and (i) that some false vertices appear in the detection results, but this does not affect the subsequent contour extraction. The module designed for

proposing lines can efficiently mitigate the presence of line proposals stemming from inaccurate vertices, thereby reducing discrimination against false vertices.

For the quantitative measurement of the vertex detection module, we employ Precision and Recall to evaluate the accuracy of voxel classification. Precision is a performance metric used in classification tasks, particularly in binary classification, to measure the accuracy of positive predictions made by a model. It quantifies the proportion of correctly predicted positive instances out of all instances that the model predicted as positive. Recall is a performance metric used in classification tasks to measure a model's ability to correctly identify all relevant instances of the positive class. In other words, it quantifies the proportion of true positive instances that the model correctly predicted out of all actual positive instances. The metrics are defined as

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}}. \end{aligned} \tag{5}$$

Also, for all voxels that contain a vertex, we also calculate the mean distance mDis between the ground true and predicted vertices, in meters. Table I shows the results of experiments on the vertex detection module. The first five lines in Table I correspond to the five point clouds shown in Fig. 4, and the last line is the evaluation of all 12 test point clouds. Based on the results of the experiments, it is evident that the 0.8754 recall metrics of the test point clouds exhibit satisfactory performance. The objective is to identify and detect as many vertices as possible, as this would facilitate the generation of line proposals. In turn, this would enable subsequent modules to receive inputs comprising nearly all the contour segments. In addition, a mean distance of less than 0.15 also ensures the accuracy of the position of line proposals.

We further use the deep learning-based keypoint detection network USIP [44] to replace the vertex detection module for contour point detection. Table II shows the quantitative results of experiments on USIP [44]. Compared to the vertex detection module, USIP [44] performs extremely poorly, mainly because the points detected by USIP [44] with well-defined positions that are highly repeatable on 3-D point clouds under arbitrary SE(3) transformations are actually not contour endpoints. This problem is magnified in large-scale 3-D point clouds, where the detected keypoints are more spread out on the surface as well as in the middle of the contour, rather than appearing exactly at the endpoint locations, as shown in Fig. 5.

2) Line Proposal Discrimination Module: To evaluate the performance of the line proposal discrimination module, we take the vertex ground truth as input to generate line proposals for training and evaluation. The results are shown in Fig. 6. The first column of Fig. 6 is the input point cloud, and the second column is the ground truth. The outputs of the line proposal discrimination module are shown in the last column in which the correct vertex is denoted by red lines, while the incorrect one is denoted by blue lines.

Table III shows the quantitative results of experiments on the line proposal discrimination module. We measure the classification accuracy of line proposals using mean accuracy (mAcc) and mean IoU (intersection over union) (mIoU). mAcc is a common metric used to evaluate the performance of classification models, it measures the proportion of correctly classified instances or data points in a classification problem. mIoU is a metric commonly used to evaluate the performance

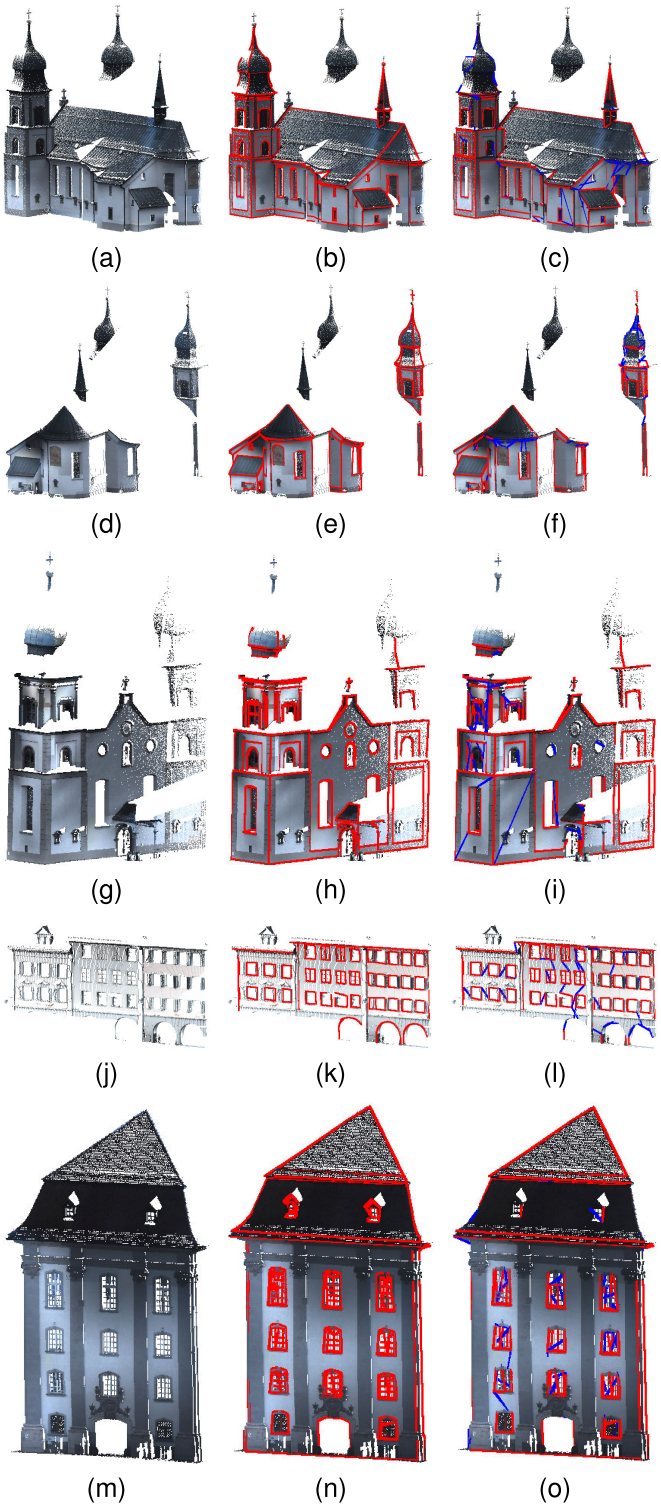


Fig. 6. Evaluation of the line proposal discrimination module. From left to right are the input point clouds, ground truth, and our results for each set. (a), (d), (g), (j), and (m) Input point clouds. (b), (e), (h), (k), and (n) Ground truth. (c), (f), (i), (l), and (o) Our results. The red lines indicate the correct ones and the blue lines indicate the wrong ones.

TABLE III
EVALUATION OF THE LINE PROPOSAL DISCRIMINATION MODULE ON SEMANTIC3D WITH CONTOUR GROUND-TRUTH REANNOTATED

Point Cloud	Number of line proposals	mAcc	mIoU
bildstein_station1_0	77915	0.9988	0.6389
bildstein_station3_0	49076	0.9982	0.5578
bildstein_station5_1	66814	0.9996	0.8246
marketplacefeldkirch_station4_2	23481	0.9985	0.8023
stgallencathedral_station1_6	59864	0.9993	0.7425
All	387821	0.9987	0.6963

TABLE IV
INTRINSIC CONNECTION BETWEEN TWO MODULES ON SEMANTIC3D WITH CONTOUR GROUND-TRUTH REANNOTATED

Point Cloud	Number of voxels	Precision	Recall	mDis	Number of line proposals	mAcc	mIoU
bildstein_station1_0	6310	0.9046	0.8529	0.1474	77915	0.9988	0.6389
bildstein_station1_0	6310	0.7887	0.8442	0.1491	90525	0.9983	0.6254
bildstein_station1_0	6310	0.8984	0.7111	0.1465	49455	0.9966	0.5502
bildstein_station1_0	6310	0.6679	0.9196	0.1480	149878	0.9948	0.4254

TABLE V
ROBUSTNESS ANALYSIS OF LCE-NET ON SEMANTIC3D WITH CONTOUR GROUND-TRUTH REANNOTATED

Point Cloud	Number of input points	Number of voxels	Precision	Recall	mDis	Number of line proposals	mAcc	mIoU
bildstein_station1_0	100%	6310	0.9046	0.8529	0.1474	77915	0.9988	0.6389
marketplacefeldkirch_station4_2	100%	1620	0.8907	0.8326	0.1475	23481	0.9985	0.8023
All	100%	25988	0.8624	0.8754	0.1487	387821	0.9987	0.6963
bildstein_station1_0	70%	6222	0.8026	0.8265	0.1472	84255	0.9961	0.6047
marketplacefeldkirch_station4_2	70%	1598	0.8390	0.7996	0.1475	21945	0.9956	0.7541
All	70%	25638	0.8291	0.8069	0.1486	430126	0.9955	0.6482
bildstein_station1_0	40%	6000	0.7098	0.6405	0.1468	63190	0.9941	0.5154
marketplacefeldkirch_station4_2	40%	1560	0.6858	0.7013	0.1472	24753	0.9946	0.5502
All	40%	24950	0.7392	0.6706	0.1483	353826	0.9943	0.5254
bildstein_station1_0	10%	5282	0.6930	0.5015	0.1461	36856	0.9925	0.3556
marketplacefeldkirch_station4_2	10%	1317	0.6132	0.5384	0.1457	15931	0.9946	0.4355
All	10%	22524	0.6639	0.5516	0.1472	282302	0.9939	0.4034

of computer vision models, particularly in tasks like image segmentation and object detection. It measures the overlap between the predicted and ground truth regions in an image. mIoU is a value between 0 and 1, with higher values indicating better model performance. The metrics are defined as

$$\text{mAcc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$
$$\text{mIoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}. \quad (6)$$

The first five lines in Table III correspond to the five point clouds shown in Fig. 6, and the last line is the evaluation of all 12 test point clouds. Based on the accuracy metric, our line proposal discrimination module achieves 0.9987 in mAcc and 0.6963 in mIoU, which can efficiently eliminate a significant number of noncontour line segments. This assertion is further supported by the visualization results, which demonstrates that only a small number of false contours are retained.

3) Intrinsic Connection Between Two Modules: LCE-NET is a framework consisting of two modules. Theoretically, if all voxels are considered as contour vertices in the first stage, LCE-NET only needs the line proposal discrimination module. To validate the necessity of the vertex detection module and explore how its performance affects the final results of LCE-NET, additional experiments were done and the quantitative results are shown in Table IV.

We use different checkpoints to make the vertex detection module differ in metrics and choose one point cloud from the test set for the experiment to better demonstrate the impact of the vertex detection module metrics on the line proposal discrimination module. The first three lines in Table IV illustrate that Recall is the main factor affecting the performance of LCE-NET after Precision has reached a certain level, due to the fact that the line proposals generated by FP point are suppressed by the line proposal discrimination module. The second line and the fourth line in Table IV illustrate that even though Recall is high, the performance of LCE-NET still deteriorates as Precision decreases, which suggests that the vertex detection module actually shares a large portion of the discriminative pressure of the line proposal discrimination module and is necessary for LCE-NET. We recommend that the Precision should be at least around 0.8.

TABLE VI
COMPARISONS OF LCE-NET WITH PREVIOUS MODELS ON SEMANTIC3D WITH CONTOUR GROUND-TRUTH REANNOTATED

Method	Precision	Recall	IoU
LCE-NET	0.8996	0.5355	0.4559
Lin et al. [36]	0.7723	0.4033	0.1984
Lu et al. [31]	0.8103	0.5242	0.2809
VCM [45]	0.7647	0.5154	0.2257
EC-Net [16]	0.8220	0.4627	0.2847
PIE-NET [17]	0.8302	0.5132	0.3216
RandLA-Net [46]	0.7080	0.2250	0.1787

4) Robustness Analysis of LCE-NET: To analyze the robustness of LCE-NET, we use the random sampling method to retain 100%, 70%, 40%, and 10% points of the test set point cloud and then input them into LCE-NET for the experiment. Table V shows the performance of LCE-NET under different

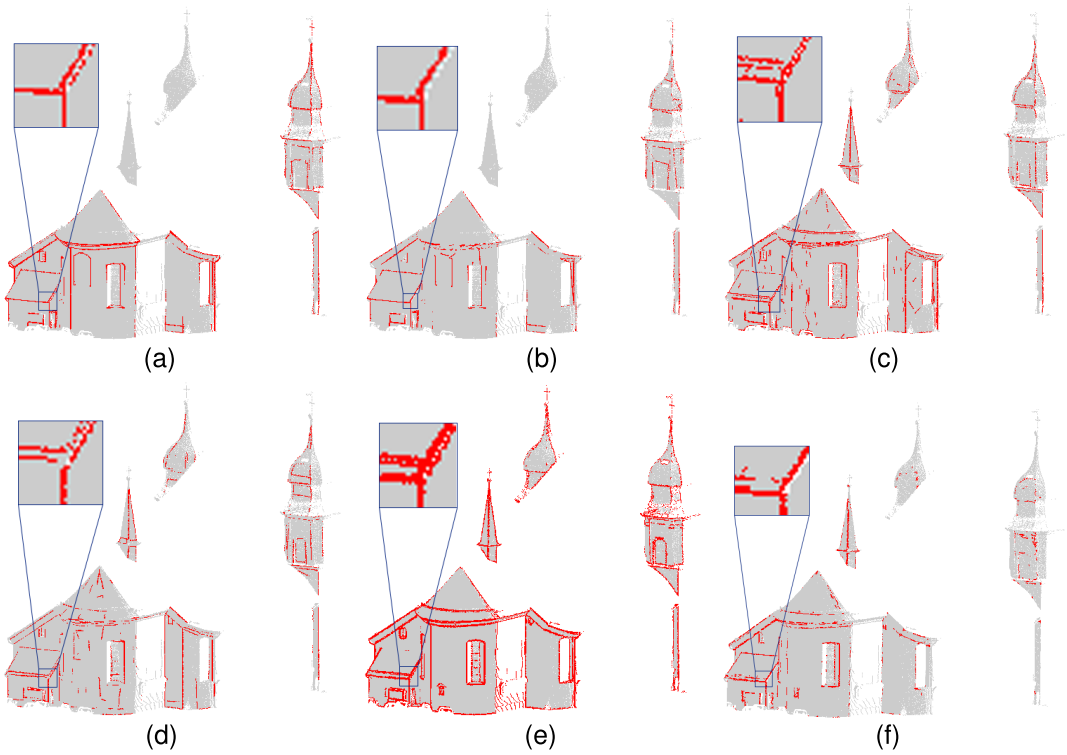


Fig. 7. Comparisons to previous works on point cloud bildstein_station3_0. (a) Ground truth. (b) LCE-NET. (c) Lin et al. [36]. (d) Lu et al. [31]. (e) VCM [45]. (f) EC-Net [16].

input point cloud densities. Basically, the performance of LCE-NET shows a linear decreasing trend as the density of the point cloud decreases, which is mainly due to the fact that the decrease in the number of points reduces the local neighborhood information of the point cloud, thus increasing the difficulty of the network's discrimination. We recommend that the density of the input point cloud should not be less than 70% of the original point cloud.

C. Comparing With Other Methods

Previous learning-based point cloud contour extraction methods, such as PIE-NET [17], are mainly focused on small-scale point clouds. Thus, it is nearly impossible for those methods to deal with large-scale point clouds with more than 10^7 points. In this section, we compare our LCE-NET with state-of-the-art feature-based methods, VCM [45], Lin et al. [36], and Lu et al. [31], and learning-based methods, EC-Net [16] and PIE-NET [17]. In addition, we also conduct experiments on RandLA-Net [46] to demonstrate the performance of semantic segmentation network on this task.

We train our LCE-NET and other learning-based methods on the same 28 point clouds and the other 12 point clouds are used for evaluation. Furthermore, we compute VCM [45] for each point with 0.2 offset radius and 0.1 convolution radius, which are the same parameters used by Wang et al. [17]. For the experiments on Lin et al.'s [36] and Lu et al.'s [31] methods, we use the default parameters provided by them. As for RandLA-Net [46], we treat contour extraction as a semantic segmentation task containing only two classes, contour and noncontour points, and then use the same dataset for training.

To enable quantitative comparison, we adopted a sampling strategy for determining the ground-truth contour points within a 5-cm radius of each contour line segment, as an integral part of our dataset. In addition, for evaluating the precision and recall of line segments, we also implemented a sampling method, which captures all points within a 5-cm proximity of each segment.

Table VI shows the quantitative results of the comparison. Here, we use the metrics commonly used in other contour extraction methods [17]. Specifically, the metrics are defined as

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP}+\text{FN}} \\ \text{mIoU} &= \frac{\text{TP}}{\text{TP}+\text{FP}+\text{FN}}. \end{aligned} \tag{7}$$

It can be seen that LCE-NET outperforms other methods in all evaluation metrics. Here, we list the results of RandLA-Net [46] when treating contour extraction as a point cloud semantic segmentation task. From the results, RandLA-Net [46] has an extremely low recall value, indicating that there are a large number of contours that cannot be detected. RandLA-Net [46] is a network designed for point cloud semantic segmentation tasks. The data features of the point cloud contour extraction dataset we use are quite different from the semantic segmentation tasks. Such a comparison is unfair to RandLA-Net [46]. To adapt to large-scale point clouds (greater than 10^7 points), semantic segmentation networks are commonly designed with

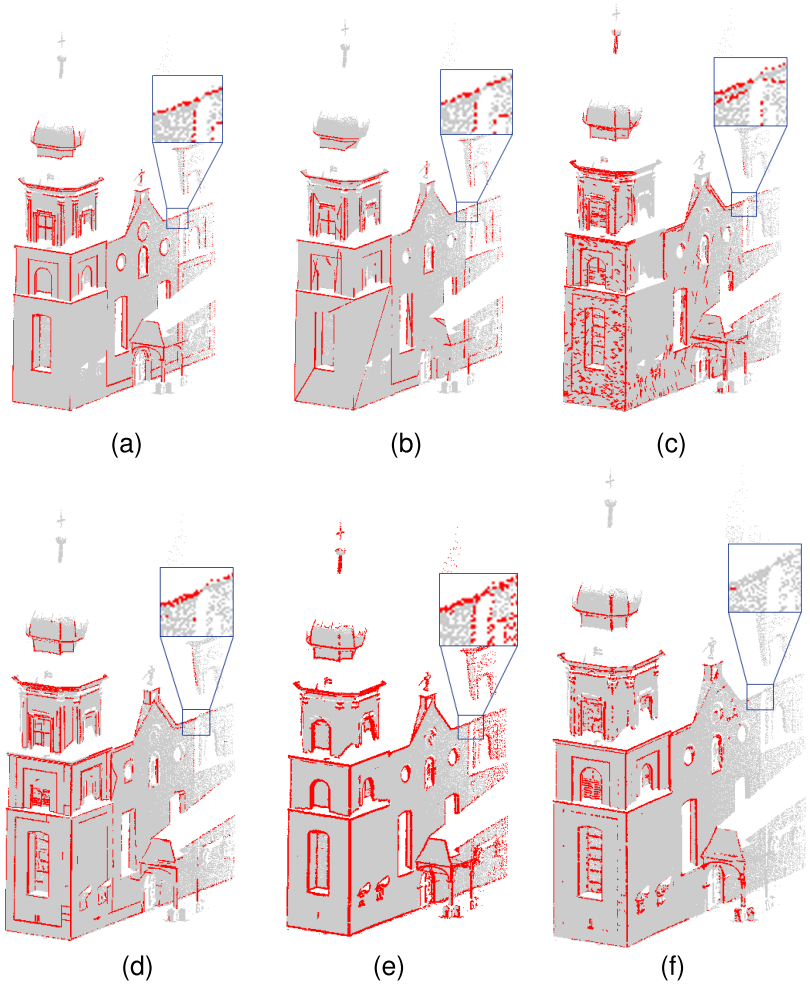


Fig. 8. Comparisons to previous works on point cloud bildstein_station5_1. (a) Point cloud. (b) LCE-NET. (c) Lin et al. [36]. (d) Lu et al. [31]. (e) VCM [45]. (f) EC-Net [16].

a special downsampling strategy, which is able to retain the effective semantic features in the point cloud neighborhood after multiple downsampling but inevitably destroys the original geometric structure of the point cloud. Unlike the semantic segmentation task, the class of points will change frequently over a very small region of the point cloud in the contour extraction task, which makes the features extracted by the semantic segmentation methods unable to be adapted so that contour extraction cannot simply be achieved using point cloud semantic segmentation networks.

The visualization results are shown in Figs. 7–10. The first column is the input point clouds. The remaining columns show the results of our approach, Lin et al. [36], Lu et al. [31], VCM [45], and EC-Net [16].

As we can see from Fig. 7(c), some false line segments appear on curved walls. Lin et al. [36] segmented point cloud into facets, resulting in their method not getting good results on the curved surface. Fig. 8(c) also shows similar results. When the structure of the wall is relatively complex, the plane-based method will have very obvious false contours. In the scene in Fig. 9(c), the density of the point cloud gradually decreases from right to left. Since Lin et al.'s [36] method is feature-based, it is difficult to adapt to different densities without tuning the parameters. Therefore, it can be seen from Fig. 9(c) that the contours of the point cloud on the left with low density cannot be detected. The top of Fig. 10(c) is also the case where the point cloud density is not uniform. Lin et al.'s [36] method also has the same problem that the contour is not detected.

As can be seen from Fig. 7(d), Lu et al.'s [31] method performs slightly better than Fig. 7(c) on curved walls, but still significantly worse than Fig. 7(b) and (e). Because Lu et al. [31] needed to project the point cloud to a 2-D plane for contour extraction, it still cannot achieve good performance on curved surfaces. It can be seen in Fig. 8(d) that Lu et al. [31] tended to generate too many contours, which leads to multiple line segments appearing in a single contour. On the left of Fig. 9(d), although it is located on a flat and regular surface, Lu et al.'s method [31] has not completely detected all the contours. This shows that the density of the point cloud has a great influence on Lu et al.'s method [31]. The results in Fig. 10(d) also well verify this conclusion.

The performance of the VCM [45] in Fig. 7(e) is acceptable, with only a few false contours appearing in a few places. However, in Fig. 8(e), some less dense areas appear on the right side of the point cloud, and VCM [45] starts to have a

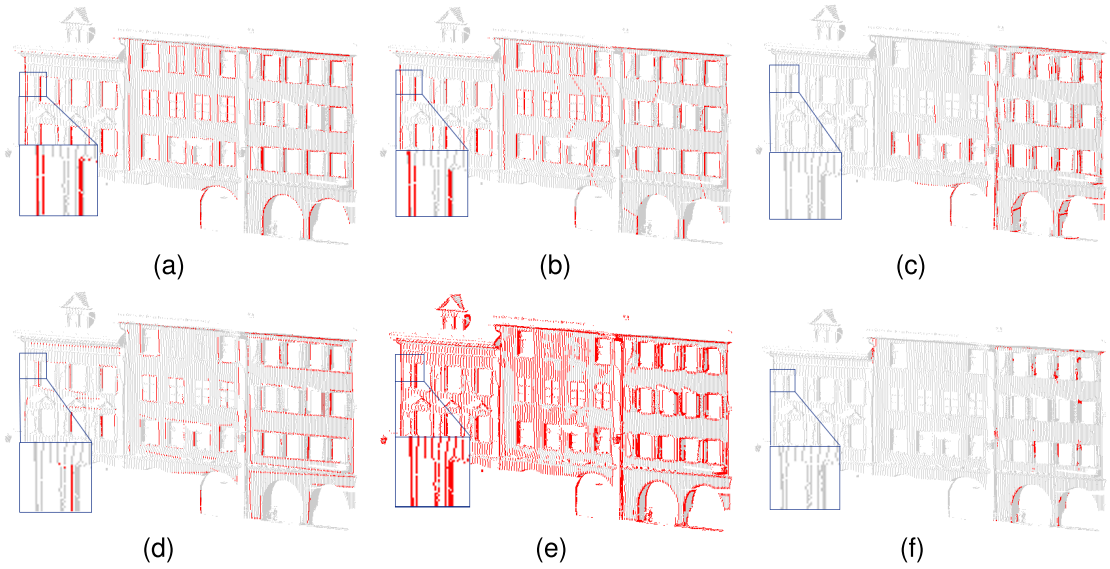


Fig. 9. Comparisons to previous works on point cloud marketplacefeldkirch_station4_2. (a) Point Cloud. (b) LCE-NET. (c) Lin et al. [36]. (d) Lu et al. [31]. (e) VCM [45]. (f) EC-Net [16].

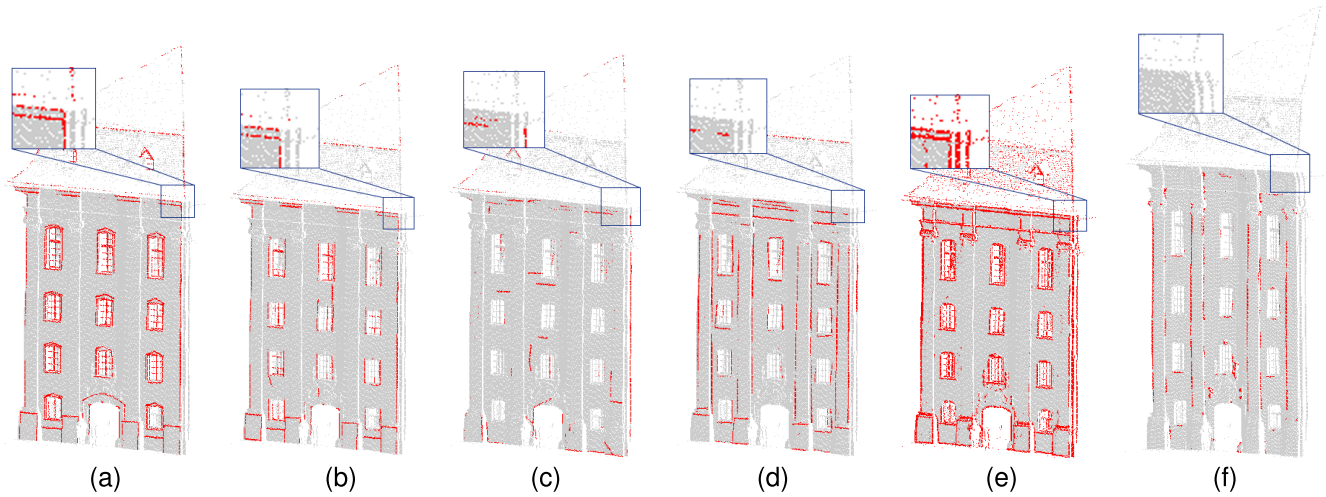


Fig. 10. Comparisons to previous works on point cloud stgallencathedral_station3_9. (a) Point Cloud. (b) LCE-NET. (c) Lin et al. [36]. (d) Lu et al. [31]. (e) VCM [45]. (f) EC-Net [16].

lot of false contours. The results in Fig. 9(e) further prove this situation, and VCM [45] has the most false contours of all methods.

EC-Net [16], as a learning-based method, significantly outperforms other feature-based methods in the results in Fig. 7(f). However, in Fig. 8(f), EC-Net [16] also has some false contour points in the top left of the result. In addition, on the right side of the result, where the point cloud density is low, a large number of edges have not been successfully extracted. Such performance is significantly worse than the proposed method. In the scene in Fig. 9(f), the density of the point cloud is low and not uniform. EC-Net [16] performed poorly and could barely extract contours. Fig. 10(f) also shows almost the same situation, which shows that the generalization performance of EC-Net [16] is poor when the point cloud density changes.

Overall, it is feasible for feature-based techniques to generate inaccurate outlines when the density of the point clouds they operate on is not uniform. Conversely, our approach based on machine learning can handle point clouds with varying densities effectively without requiring parameter adjustments. Lin et al. [36] faced challenges in obtaining satisfactory outcomes on intricate walls and curved surfaces with complex architectures. Lu et al.'s method [31] also performs poorly on curved surfaces and tends to generate multiple line segments on a single contour. VCM [45] can detect suitable contours when the parameter settings are appropriate and the density of the point cloud does not change much, but it performs the worst on the point cloud with uneven density. EC-Net [16], as a learning-based method, exhibits better performance than feature-based methods but falls short of our method in terms of generalization. Overall, the LCE-NET exhibits the most optimal results for contour detection, with only a limited number of occurrences of false contours. The incorporation of a learning-based method within LCE-NET enables the identification of contours that align with human perception and can effectively adjust to point clouds featuring varying densities.

V. Conclusion

This article proposes an LCE-NET, which is the first end-to-end deep learning-based method to extract contours from large-scale point clouds. Based on a two-stage network, our LCE-NET is capable of producing accurate parameterized contour predictions. Several experiments have shown that the proposed method outperforms state-of-the-art methods in terms of generalization and performance when the density of the point clouds is uneven. Besides, we open-sourced SemanticLine, the first dataset for large-scale point clouds with labeled contours, based on reannotation of semantic3D.

References

[1] R. Collis, "LiDAR," in *Advances in Geophysics*, vol. 13. Amsterdam, The Netherlands: Elsevier, 1969, pp. 113–139.

[2] S. Ullman, "The interpretation of structure from motion," *Proc. Roy. Soc. London B, Biol. Sci.*, vol. 203, no. 1153, pp. 405–426, Jan. 1979.

[3] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3D object recognition in cluttered scenes with local surface features: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2270–2287, Nov. 2014.

[4] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021.

[5] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[6] Y. Xia, Y. Xia, W. Li, R. Song, K. Cao, and U. Stilla, "ASFM-Net: Asymmetrical Siamese feature matching network for point completion," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 1938–1947.

[7] Y. Xia et al., "SOE-Net: A self-attention and orientation encoding network for point cloud based place recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11348–11357.

[8] T. Hackel, J. D. Wegner, and K. Schindler, "Contour detection in unstructured 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1610–1618.

[9] P. Borges, R. Zlot, M. Bosse, S. Nuske, and A. Tews, "Vision-based localization using an edge map extracted from 3D laser range data," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 4902–4909.

[10] F. Tupin, H. Maitre, J.-F. Mangin, J.-M. Nicolas, and E. Pechersky, "Detection of linear features in SAR images: Application to road network extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 36, no. 2, pp. 434–453, Mar. 1998.

[11] D. Ceylan, N. J. Mitra, H. Li, T. Weise, and M. Pauly, "Factored facade acquisition using symmetric line arrangements," *Comput. Graph. Forum*, vol. 31, nos. 2–3, pp. 671–680, May 2012.

[12] Y. Su, Y. Liu, B. Cuan, and N. Zheng, "Contour guided hierarchical model for shape matching," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1609–1617.

[13] J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: A learned mid-level representation for contour and object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3158–3165.

[14] C. Weber, S. Hahmann, and H. Hagen, "Sharp feature detection in point clouds," in *Proc. Shape Modeling Int. Conf.*, Jun. 2010, pp. 175–186.

[15] Y. Lin et al., "Line segment extraction for large scale unorganized point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 102, pp. 172–183, Apr. 2015.

[16] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "EC-Net: An edge-aware point set consolidation network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 386–402.

[17] X. Wang et al., "PIE-NET: Parametric inference of point cloud edges," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 20167–20178.

[18] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[19] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognit.*, vol. 13, no. 2, pp. 111–122, Jan. 1981.

[20] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, Apr. 2010.

[21] N.-G. Cho, A. Yuille, and S.-W. Lee, "A novel linelet-based representation for line segment detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1195–1208, May 2018.

[22] W. Li, Q. Huang, and G. Srivastava, "Contour feature extraction of medical image based on multi-threshold optimization," *Mobile Netw. Appl.*, vol. 26, no. 1, pp. 381–389, Feb. 2021.

[23] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, "Learning to parse wireframes in images of man-made environments," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 626–635.

[24] N. Xue, S. Bai, F. Wang, G.-S. Xia, T. Wu, and L. Zhang, "Learning attraction field representation for robust line segment detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1595–1603.

[25] Z. Zhang et al., "PPGNet: Learning point-pair graph for line segment detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7105–7114.

[26] Y. Zhou, H. Qi, and Y. Ma, "End-to-end wireframe parsing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 962–971.

[27] R.-W. Bello, A. S. A. Mohamed, and A. Z. Talib, "Contour extraction of individual cattle from an image using enhanced mask R-CNN instance segmentation method," *IEEE Access*, vol. 9, pp. 56984–57000, 2021.

[28] C. J. Taylor and D. J. Kriegman, "Structure and motion from line segments in multiple images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 11, pp. 1021–1032, Nov. 1995.

[29] A. Jain, C. Kurz, T. Thormählen, and H.-P. Seidel, "Exploiting global connectivity constraints for reconstruction of 3D line segments from images," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1586–1593.

[30] R. Grompone Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A line segment detector," *Image Process. Line*, vol. 2, pp. 35–55, Mar. 2012.

[31] X. Lu, Y. Liu, and K. Li, "Fast 3D line segment detection from unorganized point cloud," 2019, *arXiv:1901.02532*.

[32] Q. Meng, J. Zhang, Q. Hu, X. He, and J. Yu, "LGNN: A context-aware line segment detector," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 4364–4372.

[33] Y. Ohtake, A. Belyaev, and H.-P. Seidel, "Ridge-valley lines on meshes via implicit surface fitting," in *Proc. ACM SIGGRAPH Papers*, Aug. 2004, pp. 609–612.

[34] A. Sampath and J. Shan, "Segmentation and reconstruction of polyhedral building roofs from aerial LiDAR point clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 3, pp. 1554–1567, Mar. 2010.

[35] P. Moghadam, M. Bosse, and R. Zlot, "Line-based extrinsic calibration of range and image sensors," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 3685–3691.

[36] Y. Lin, C. Wang, B. Chen, D. Zai, and J. Li, "Facet segmentation-based line segment extraction for large-scale point clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 9, pp. 4839–4854, Sep. 2017.

[37] Y. Xu and U. Stilla, "Contour extraction of planar elements of building facades from point clouds using global graph-based clustering," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 4, pp. 211–219, Sep. 2019.

[38] Y. Ioannou, B. Taati, R. Harrap, and M. Greenspan, "Difference of normals as a multi-scale operator in unorganized point clouds," in *Proc. 2nd Int. Conf. 3D Imag., Modeling, Process., Vis. Transmiss.*, Oct. 2012, pp. 501–508.

[39] D. Bazazian, J. R. Casas, and J. Ruiz-Hidalgo, "Fast and robust edge extraction in unorganized point clouds," in *Proc. Int. Conf. Digit. Image Comput., Techn. Appl. (DICTA)*, Nov. 2015, pp. 1–8.

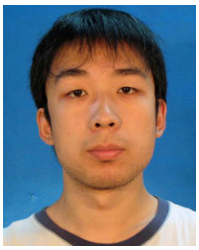
[40] C.-E. Himeur, T. Lejemble, T. Pellegrini, M. Paulin, L. Barthe, and N. Mellado, "PCEDNet: A lightweight neural network for fast and interactive edge detection in 3D point clouds," *ACM Trans. Graph.*, vol. 41, no. 1, pp. 1–21, Feb. 2022.

[41] W. Zhang, L. Chen, Z. Xiong, Y. Zang, J. Li, and L. Zhao, "Large-scale point cloud contour extraction via 3D guided multi-conditional generative adversarial network," *ISPRS J. Photogramm. Remote Sens.*, vol. 164, pp. 97–105, Jun. 2020.

[42] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Oct. 2019.

[43] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[44] J. Li and G. H. Lee, "USIP: Unsupervised stable interest point detection from 3D point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 361–370.
[45] Q. Mérigot, M. Ovsjanikov, and L. J. Guibas, "Voronoi-based curvature and feature estimation from point clouds," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 6, pp. 743–756, Jun. 2011.
[46] Q. Hu et al., "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11108–11117.
[47] T. Hackel, J. D. Wegner, N. Savinov, L. Ladicky, K. Schindler, and M. Pollefeys, "Large-scale supervised learning for 3D point cloud labeling: Semantic3d.Net," *Photogram. Eng. Remote Sens.*, vol. 84, no. 5, pp. 297–308, May 2018.

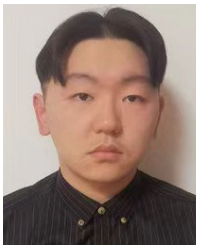


Yu Zang (Member, IEEE) received the B.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 2008 and 2014, respectively.

He is currently a Research Associate Professor with the School of Informatics, Xiamen University, Xiamen, China. His main research interests include remote sensing image processing, computer vision and graphics, and mobile light detection and ranging (LiDAR) data analysis.

Binjie Chen received the B.S. degree from the East China University of Political Science and Law, Shanghai, China, in 2020. He is currently pursuing the M.S. degree with the ASC Laboratory, Xiamen University, Xiamen, China, under the supervision of Prof. Yu Zang.

His research interests include point cloud representation learning and deep learning.



Yunzhou Xia received the B.S. degree in computer science from Suzhou University, Suzhou, China, in 2020. He is currently pursuing the M.S. degree with the ASC Laboratory, Xiamen University, Xiamen, China, under the supervision of Prof. Yu Zang.

His research interests include point cloud feature extraction and contour detection.

Hanyun Guo received the B.S. degree from the Nanjing University of Science and Technology, Nanjing, China, in 2020. He is currently pursuing the M.S. degree with the School of Informatics, Xiamen University, Xiamen, China.

His research interests include point cloud processing and deep learning.

Yunuo Yang received the B.S. degree in computer science from Northeast Normal University, Changchun, China, in 2019, and the M.S. degree in electronics and computer science from the University of Southampton, Southampton, U.K., in 2020. She is currently pursuing the Ph.D. degree with the ASC Laboratory, Xiamen University, Xiamen, China, under the supervision of Prof. Cheng Wang.

Her research interests include remote sensing images and object detection.

Weiquan Liu (Member, IEEE) received the B.S. and M.S. degrees in applied mathematics from the College of Science, Jimei University, Xiamen, China, in 2013 and 2016, respectively, and the Ph.D. degree in computer science and technology from the School of Informatics, Xiamen University, Xiamen, in 2020.

He currently holds a post-doctoral position at the Information and Communication Engineering Postdoctoral Research Station and the Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Informatics, Xiamen University. His research interests include remote sensing, computer vision, machine learning, mobile laser scanning point cloud data processing, and augmented reality.

Cheng Wang (Senior Member, IEEE) received the Ph.D. degree in signal and information processing from the National University of Defense Technology, Changsha, China, in 2002.

He is currently a Professor with the School of Informatics and the Executive Director of the Fujian Key Laboratory of Sensing and Computing for Smart Cities, Xiamen University, Xiamen, China. He has coauthored more than 150 papers in refereed journals and top conferences, including IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, PR, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE Conference on Computer Vision and Pattern Recognition, Association for the Advancement of Artificial Intelligence (AAAI), and *International Society for Photogrammetry and Remote Sensing (ISPRS) Journal of Photogrammetry and Remote Sensing*. His research interests include point cloud analysis, multisensor fusion, mobile mapping, and geospatial big data.

Dr. Wang is a fellow of the Institution of Engineering and Technology. He is also the Chair of the Working Group I/6 on Multi-Sensor Integration and Fusion of the International Society of Remote Sensing.

Jonathan Li (Fellow, IEEE) 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 of geomatics and a Systems Design Engineer at the University of Waterloo, Waterloo, ON, Canada. He is also a Founding Member of the Waterloo Artificial Intelligence Institute, University of Waterloo. His research interests include artificial intelligence (AI)-based information extraction from mobile light detection and ranging (LiDAR) point clouds and Earth observation images. He has coauthored more than 450 publications, more than 260 of which were published in refereed journals, including IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *International Society for Photogrammetry and Remote Sensing (ISPRS) Journal of Photogrammetry and Remote Sensing*, and *Remote Sensing of Environment*.

Dr. Li was a recipient of the ISPRS Samuel Gamble Award in 2020. He is also the Editor-in-Chief of the *International Journal of Applied Earth Observation and Geoinformation* and an Associate Editor of IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, and *Canadian Journal of Remote Sensing*.