Neural Network-Based Urban Airborne LiDAR Point Cloud Classification Algorithm: Postprint

Authors: Shi Xiaosong, Cheng Yinglei, Zhao Zhongyang

Date: 2019-01-28T00:00:00+00:00

Abstract

To enable neural network application for urban LiDAR point cloud data classification and address the challenges of high computational cost and prolonged training time in large-scale point cloud data processing, we enhanced the original Point-Net architecture by incorporating extraction and analysis of point cloud neighborhood features, thereby proposing a novel point cloud classification algorithm. The method compresses the original point cloud data volume through gridded clustering and resampling, extracts multi-scale neighborhood point cloud information, and employs the improved Point-Net to perform urban point cloud classification. The proposed algorithm was validated using datasets from various regions. Experimental results demonstrate favorable classification performance with high accuracy and reduced computational load during training. Consequently, the algorithm enables effective classification of urban airborne LiDAR data.

Full Text

Preamble

Airborne LiDAR Point Cloud Classification Algorithm for Urban Areas Using Neural Networks

Shi Xiaosong, Cheng Yinglei, Zhao Zhongyang
(College of Information & Navigation, Air Force Engineering University, Xi' an 710077, China)

Abstract: To apply neural networks to urban LiDAR point cloud data classification and address the problems of large computational load and long training time in large-scale point cloud data processing, this paper improves the original Point-Net neural network by incorporating point cloud neighborhood feature extraction and analysis, and proposes a new point cloud classification algorithm based on this improved network. The algorithm compresses the original

point cloud data volume through grid-based clustering and resampling, extracts multi-scale neighborhood point cloud data, and completes urban point cloud classification using the improved Point-Net. The classification algorithm was validated using data from different regions. Results demonstrate that the algorithm achieves good classification performance with high accuracy while reducing computational load during training. Therefore, the algorithm can effectively classify airborne LiDAR data in urban areas.

Keywords: neural network; LiDAR; data compression; neighborhood feature extraction; point cloud classification

In recent years, with the rise of Light Detection and Ranging (LiDAR) technology, three-dimensional point cloud data has been widely applied in remote sensing, mapping, and other fields [1]. Point cloud data is not affected by factors such as illumination and shadows, and contains rich three-dimensional spatial information [2]. Currently, the most popular direction in point cloud processing is applying machine learning to point cloud data classification. The most commonly used machine learning methods include cluster analysis [3], Support Vector Machine (SVM) [4], Random Forest (RF) [5], and Bayesian networks [6].

Machine learning approaches extract multiple features from existing point cloud data for classification. Mallet et al. [7] extracted full-waveform features and used SVM for point cloud classification. Anguelov et al. [8] used a Markov Random Field model to classify point cloud data into three categories: buildings, vegetation, and shrubs. Vu et al. [9] proposed a K-means clustering method based on the elevation values of laser footprint data to segment data into three major categories: high-rise buildings, ground points, and other features. Feature extraction for point cloud data typically requires analysis together with neighboring points, and the definition of various features must consider the shape and size of the neighborhood, as these features characterize the properties of point cloud data within a small region [10]. LiDAR scanning typically produces extremely large point cloud datasets, and machine learning methods require manual feature extraction, which is complex, time-consuming, and suffers from limited expressive capability of the extracted features.

With the resurgence of deep learning, researchers have recently begun using deep neural networks for point cloud classification. As a branch of machine learning, deep learning obtains features through cascaded neural networks with layer-by-layer abstraction, eliminating the need for manual feature extraction, reducing researcher workload, and achieving better results than other machine learning methods [11]. Early deep learning classification methods for point clouds first voxelized or compressed the data into two dimensions for processing with binary neural networks. While this approach could order the input point clouds, it caused significant loss of point cloud information, and neural networks incurred enormous computational costs when processing large-scale point cloud data [12]. These issues have slowed research progress on deep learning classification of urban point cloud data. In 2017, Qi et al. [13] proposed the Point-Net neural network, which can directly input point cloud data for classification processing,

avoiding information loss caused by voxelization or 2D compression. However, Point-Net only considers single-point features and global features of point clouds, ignoring local neighborhood features, leading to classification errors in details. Additionally, Point-Net generates massive computational load when processing large-scale point cloud data.

This paper improves upon the original Point-Net and proposes a new point cloud feature classification algorithm based on the improved network. The algorithm removes isolated points and redundant points to compress point cloud data volume while retaining necessary information for training. The improved Point-Net network incorporates local neighborhood information into the original Point-Net to train classification models, enhancing the model' s ability to classify point cloud data details. This paper will introduce the classification algorithm in detail from three aspects: algorithm principles and workflow, simulation experiments, and experimental results analysis, summarize the algorithm' s advantages and disadvantages, and propose future research directions.

1 Algorithm Description and Contributions

This section describes the specific workflow of the proposed point cloud classification algorithm and highlights the contributions of this work. The main idea of the algorithm is to compress point cloud data volume while preserving necessary information and train using the improved Point-Net network to obtain a point cloud classification model. [Figure 1: see original paper] shows the detailed algorithm flowchart.

1.1 Removing Isolated and Redundant Points

Point cloud data contains some points that are far from other points, commonly called isolated points (or noise points). Isolated points cause ambiguity in point cloud classification, reduce classification accuracy, increase unnecessary computational load, and consume storage resources. In urban point cloud data, some regions have excessively dense points, such as building roofs and dense tree clusters. These points are called redundant points. Removing redundant points has little impact on point cloud data, and Point-Net networks exhibit good robustness to local subtle changes in input point clouds. Therefore, redundant points can be appropriately removed to compress point cloud data while ensuring local features are preserved. [Figure 2: see original paper] shows isolated points and redundant points in point cloud data.

The proposed algorithm uses grid-based clustering analysis to screen and remove isolated points, and grid-based resampling to screen and remove redundant points. The specific flowchart is shown in [Figure 3: see original paper].

First, three-dimensional grids are used to segment point cloud data [14]. The grids are uniformly divided according to equation (1):

$$\begin{cases} Grid_x = \lfloor \frac{x - x_{\min}}{d_x} \rfloor + 1 \\ Grid_y = \lfloor \frac{y - y_{\min}}{d_y} \rfloor + 1 \\ Grid_z = \lfloor \frac{z - z_{\min}}{d_z} \rfloor + 1 \\ d_x = \frac{x_{\max} - x_{\min}}{n_x} \\ d_y = \frac{y_{\max} - y_{\min}}{n_y} \\ d_z = \frac{z_{\max} - z_{\min}}{n_z} \end{cases}$$

where (x, y, z) are point cloud coordinates; n_x, n_y, n_z are the numbers of grids in each coordinate direction; $Grid_x, Grid_y, Grid_z$ are grid positions; d_x, d_y, d_z are point spacing distances, with the average point spacing calculated using equation (2):

$$d = \sqrt[3]{\frac{V}{A}}$$

where V is the volume of the minimum bounding polyhedron of the point cloud; A represents the number of points in the point cloud dataset. The gridding of point cloud data is illustrated in [Figure 4: see original paper].

1.1.1 Removing Isolated Points When removing isolated points, a threshold T_1 is first selected. The algorithm sets T_1 as the weighted average grid density, i.e., $T_1 = a \cdot \bar{R}_n$, where $\bar{R}_n$ is the average grid density and a is a weight that can be adjusted according to actual point cloud data. When the density R_n of a grid cell is greater than T_1, that grid is selected as a seed grid, and points falling within seed grids are seed points. All seed grids in the point cloud data are identified, and all seed points in the densest grid are classified as one category, serving as the initial clustering center V_0.

Next, the distances from seed points in remaining grids V_n to seed points in the densest grid are calculated. The seed point distance is defined as Euclidean distance:

$$D(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) are the 3D coordinates of points i and j. Since the input point cloud data contains each point' s 3D coordinate values, distances between points can be conveniently calculated. When the distance D is less than threshold T_2, points are classified into the same category. All seed grids are traversed until no similar points can be clustered or the number of points in a class reaches a set threshold. Seed points not assigned to any class continue the process by selecting the densest grid as the initial clustering center V_0, repeating the above operations.

For points in remaining non-seed grids, the nearest seed point is calculated along with its distance D. If distance D is less than threshold T_2, the non-seed point is identified as the same class as the seed point; otherwise, it is judged as an isolated point. The point cloud data is traversed to identify and remove all isolated points. The results of isolated point removal are shown in [Figure 5: see original paper].

1.1.2 Removing Redundant Points When removing redundant points, a threshold T_3 is first set for each point cloud grid V_n based on the average grid density $\bar{R}_n$. The solution method for T_3 is the same as for T_1, with weights adjusted according to actual point cloud data. Grids with density greater than T_3 are selected as seed grids, while others are not processed. The point cloud data is traversed to screen and remove dense surface grids and sparse internal grids.

Afterward, resampling is performed on point cloud data in each seed grid V_n. The sampling rate η is selected based on the actual density of the point cloud data. After each sampling, the density of each point cloud grid R_n is recalculated, and the entire sampling process is repeated. After m iterations, the process completes when no seed grid density value exceeds threshold T_3. After redundant point removal, seed points in dense regions are appropriately reduced, but local information such as point cloud shape, elevation, and texture remains essentially unchanged. The results of redundant point removal are shown in [Figure 6: see original paper].

1.2 Improved Point-Net Neural Network

The Point-Net neural network incorporates a T-Net input transformation matrix and a max pooling layer. T-Net can learn a rotation matrix from point cloud position information and adjust the rotation matrix through a loss function to rotate input point cloud data to an angle more conducive to classification or segmentation. The max pooling layer is a downsampling structure after convolutional layers that reduces output dimensions while preserving salient features [15]. In Point-Net, it is used to extract global features of point clouds, with the max pooling layer formula shown in equation (5):

$$f(\{x_1, x_2, \dots, x_n\}) = \max_{i=1,\dots,n} h(x_i)$$

where $\{x_1, x_2, \dots, x_n\}$ represents input point cloud data for part of the network, f represents the max pooling layer, and h is a convolution function. The f function is a symmetric function that makes the neural network insensitive to changes in input point cloud ordering, solving the unordered nature of point clouds [16].

The introduction of T-Net and max pooling layers overcomes the unordered and rotational issues in neural network training of point cloud data, allowing Point-

Net to directly process point cloud data as input without requiring voxelization or 2D compression.

As mentioned in the introduction, Point-Net neural networks ignore local neighborhood information and features of points, leading to classification errors in certain details. This paper extracts local point cloud data at different scales and uses a small Point-Net network to extract local features for classification model training. The overall idea of local feature extraction is: first select some important points as center points for each local region, then select k nearest neighbor points around these center points based on Euclidean distance; these k nearest neighbor points form a local point cloud, and a small network is used to extract features.

1.2.1 Neighborhood Selection Typically, the more distinct the point cloud features of different object categories at multiple scales, the stronger the separability of the objects. Due to uneven point cloud data density, selecting a fixed number of nearest neighbor points within a fixed range is inappropriate, necessitating the use of different scales for local region selection.

Neighborhood selection first extracts several center points from sample point cloud data after redundant point removal, specifically using the farthest point sampling method [18]. Initially, 8 center points are manually selected from 8 different object categories to form a center point set. The point cloud data is then traversed to find the point farthest from each center point in Euclidean distance D as a new center point, which is added to the center point set. This process iterates until the farthest distance from all center points is less than threshold l, at which point iteration stops and the final center points form the complete center point set. Here, threshold l is set as the minimum distance between centers of different objects in the sample point cloud. The farthest point sampling method continuously finds qualified sampling points across the entire region, with sampling points distributed from sparse to dense, preventing oversampling-induced data redundancy while ensuring that sampled center points are distributed across different object categories. Compared to uniform distribution sampling, farthest point sampling better completes sampling of point clouds with uneven density distribution.

Afterward, neighborhood radius selection is performed. The average point spacing d determined earlier is used to determine scale s, with the scale calculation formula:

$$s = \alpha \cdot d$$

where α is an amplification factor that needs to be set according to specific point cloud data; d is the average point spacing. The minimum scale should be greater than the average point spacing d, and the maximum scale should be set according to actual object dimensions. Each center point determines neighborhoods at multiple scales, obtaining multiple Euclidean distance neighborhood

point clouds. This paper selects point cloud combinations from two neighborhood scales to extract neighborhood features. By testing classification accuracy of models trained with different scale combinations, the optimal combination is found. For the dataset, the average point spacing d is 0.3m, and the selected scale range is 0.6~6.9m, uniformly divided into 8 scales with an average scale interval of 0.9m. Experiments determined the optimal scale combination to be 1.5m and 6.9m. Using center point O_n as the center and scale s as the radius to select neighborhoods, the results of different-scale neighborhood selection are shown in [Figure 7: see original paper].

Different-scale neighborhoods selected samples of various objects including trees, buildings, and ground. Different samples present different details at different scales. Ground remains relatively flat across scales; entire trees exhibit a three-dimensional effect, while branches and shrubs within small ranges appear as lines or planes; buildings mostly have flat roofs at small scales but include edge corners and vertical wall details at large scales. Additionally, multi-scale extraction effectively addresses the problem of uneven point cloud data distribution. Small-scale neighborhoods struggle to effectively extract point cloud features in sparse regions, while large-scale neighborhoods can achieve effective feature extraction in corresponding regions.

1.2.2 Multi-scale Neighborhood Feature Extraction After multi-scale neighborhood point cloud extraction, large-scale and small-scale neighborhood point clouds are divided into two groups, with each group using a small neural network to extract features independently. The small neural network includes a T-Net input transformation matrix, a two-layer multilayer perceptron (MLP), and a max pooling layer.

The k points of each neighborhood point cloud block serve as a patch input to the small neural network. Data is first corrected through T-Net, then convolved and pooled to obtain a c-dimensional feature vector as the feature point for this center point. The output feature points from all neighborhood point cloud blocks can form a new neighborhood feature point block, which is again input into a Point-Net network with the same structure for feature extraction. As extraction depth increases, the number of feature points decreases, but each feature point contains increasingly more information.

In this algorithm, each scale group contains three small neural network layers. The third layer pools to obtain a 256-dimensional feature vector. Feature vectors from each group are connected through fully connected layers to obtain the final neighborhood feature vector. The neighborhood feature extraction and improved Point-Net network flowchart are shown in [Figure 8: see original paper].

The neighborhood feature extraction network operates in parallel with the original Point-Net neural network. Neighborhood features can be processed in parallel using other devices and then input to the original Point-Net network without

increasing the computational burden of the original network. The neighborhood feature vector is combined with the single-point features and global features obtained from the original Point-Net neural network through fully connected layers for point cloud object classification. The connection structure of single-point features, local features, and global features is shown in [Figure 9: see original paper].

The improved Point-Net neural network effectively overcomes classification errors in details that occurred in the original Point-Net neural network by extracting local features of point clouds, thereby improving overall classification accuracy.

2 Experimental Process

The LiDAR dataset used in this experiment is the Semantic3D outdoor dataset [19], which contains over 4 billion points total, including eight semantic categories such as terrain, vegetation, and buildings. It covers scenes including streets, squares, and soccer fields, with 15 training scenes and 4 test scenes. Each scene contains an average of 10-20 million data points, effectively simulating urban geographical environments. Some scenes are shown in [Figure 10: see original paper].

First, point cloud training data is processed to remove isolated and redundant points. Different point cloud scene data may have certain variations, and parameters can be appropriately adjusted during preprocessing based on actual conditions. After multiple experimental analyses, this algorithm sets the segmentation unit grid size to 20m×20m×10m, the grid point cloud density threshold T_1 to 60, and the distance threshold T_2 between points to 10m. When removing redundant points, the unit grid density threshold T_3 is set to 100, the sampling rate η is 66.66%, and the number of iterations m is 2. After processing, the average data volume of scene point clouds is reduced by 7-15 million points.

Next, neighborhood selection is performed on the data. As mentioned in Section 1.2.1, the average point spacing d is 0.3m. The small-scale amplification factor α_1 is set to 5 and the large-scale amplification factor α_2 to 23, with neighborhood radii s of 1.5m and 6.9m respectively. The farthest point sampling method selects center points O_n, with small-scale neighborhood point cloud data collected in dense regions and large-scale point cloud data collected in sparse regions.

The entire point cloud data is input into the original Point-Net network to obtain single-point features and global features. Neighborhood point cloud data is input into the improved neighborhood feature extraction network to obtain local neighborhood features. Local neighborhood features are added to the Point-Net network classifier through fully connected layers, and the network combines all features to classify point cloud data. During Point-Net network training, the maximum number of epochs is set to 200, the initial learning rate is 0.01, batch size is 32, and the Adam optimization algorithm [20] is used. After training on 15 training scenes, a trained classification model is obtained. The

trained model is then used to classify the test dataset provided by Semantic3D, with classification results shown in [Figure 11: see original paper].

Section 1.2.1 extracted neighborhood point cloud data at 8 scales ranging from 0.6~6.9m. The minimum scale is twice the average point spacing d, ensuring that features can be extracted in dense regions at this scale. The maximum scale is set to 6.9m based on object dimensions in the point cloud scenes, ensuring that neighborhoods in sparse regions contain sufficient point cloud details. The four smaller scales within the range are classified as the small-scale set, and the four larger scales as the large-scale set. One scale is selected from each set each time to extract point cloud data from sparse and dense regions respectively for training classification models, and classification performance is analyzed. Classification performance is measured by overall classification accuracy, which is the ratio of correctly predicted samples to total samples across all test sets. Some scale combinations with higher classification accuracy and comparisons with single-scale classification accuracy are shown in .

Table 1 Comparison of overall classification accuracy across different scales

Small Scale (m)	Large Scale (m)	Overall Accuracy (%)

Analysis of Table 1 shows that single-scale overall classification accuracy is far lower than multi-scale combination accuracy. Comparing different scale combinations, larger large-scale radii increase neighborhood point cloud details and improve overall classification accuracy, so the large-scale radius is selected as 6.9m. When the small-scale radius is 1.5m, classification accuracy is not significantly different from radii of 2.4m and 3.3m, and is slightly higher than the 2.4m radius, indicating that the 1.5m scale neighborhood can sufficiently extract neighborhood features in dense regions. Considering that larger neighborhood radii increase training data volume, the small-scale radius is selected as 1.5m.

The classification performance of this algorithm is evaluated and compared with other algorithms using three specific metrics: mean Intersection over Union (IoU), IoU per object class, and overall accuracy, to test classification accuracy for both overall and individual objects. IoU is expressed as a percentage and calculated using equation (7):

$$IoU = \frac{T}{T + F_1 + F_2}$$

where IoU is the intersection over union; T is the number of points correctly classified as the class in a sample; F_1 is the number of points from the class incorrectly classified as other classes; and F_2 is the number of points from other classes incorrectly classified as the class. The classification accuracy evaluation results are shown in and .

Table 2 Overall IoU and per-category IoU on Semantic3D dataset

Dataset	3D-FCNN-TI	OctreeNet_CRF	SEGCloud	Point-Net	Our Algorithm
Mean IoU (%)					
Road IoU (%)					
Grass IoU (%)					
Tree IoU (%)					
Bush IoU (%)					
Building IoU (%)					
Man-made Object IoU (%)					
Scanning Artifact IoU (%)					
Vehicle IoU (%)					

Table 2 shows the IoU values for each category and mean IoU values of our algorithm compared with other neural networks on the Semantic3D dataset. compares the overall classification accuracy of our algorithm with several other algorithms on the Semantic3D dataset.

Table 3 Overall classification accuracy on Semantic3D dataset

Dataset	3D-FCNN-TI	OctreeNet_CRF	SEGCloud	Point-Net	Our Algorithm
Overall Accuracy (%)					

Our algorithm outperforms other neural networks in mean IoU and IoU for most categories, achieving higher overall classification accuracy than other neural networks.

This paper measures Point-Net neural network computational cost through training time. By statistically analyzing training times from multiple experiments, the average training time of the original Point-Net neural network and our algorithm is obtained, as shown in . The comparison shows that network training time in our algorithm is significantly reduced compared to the original network, proving that our algorithm effectively reduces computational load.

Table 4 Training time of algorithms

Algorithm	Training Time (h)
Point-Net	
Our Algorithm	

3 Experimental Results Analysis and Evaluation

First, the extraction effectiveness of different-scale neighborhood features in this paper is evaluated. Our algorithm achieves better classification results on the Semantic3D dataset, with improved classification accuracy for both overall and individual objects.

4 Conclusion

This paper proposes an object classification method for airborne LiDAR data based on an improved Point-Net network for airborne LiDAR point cloud data classification. By removing isolated and redundant points from point cloud data, the original point cloud data is compressed, reducing computational load during training. Based on the original Point-Net network, local features of point clouds are incorporated to compensate for deficiencies in detailed classification of the Point-Net network, improving classification accuracy for both overall and individual objects. Testing on the Semantic3D dataset achieved good classification results. Meanwhile, experiments show that the classification performance of Point-Net neural networks still needs improvement and computational costs remain high. Building more optimized neural network structures to further improve computational accuracy and efficiency is the direction for future work.

References

[1] Wu Jun, Liu Rong, Guo Ning, et al. Aerial LiDAR data classification using weighted support vector machines [J]. Geomatics and Information Science of Wuhan University, 2013, 38(1): 1-5.

[2] Hao Wen, Wang Yinghui, Ning Xiaojuan, et al. Survey of 3D object recognition for point clouds [J]. Computer Science, 2017, 44(9): 11-16.

[3] Zhu Lei, Wang Jian, Xu Kaihui, et al. Classification method of vehicle-borne LiDAR point cloud based on clustering [J]. Science of Surveying and Mapping, 2016, 41(4): 77-82.

[4] Wang Xudong, Duan Fuzhou, Qu Xinyuan, et al. Building extraction based on UAV imagery data with the synergistic use of objected-based method and SVM classifier [J]. Remote Sensing for Land & Resources, 2017, 29(1): 97-103.

[5] Sun Jie, Lai Zulong. Airborne LiDAR feature selection for urban classification using random forests [J]. Geomatics and Information Science of Wuhan University, 2014, 39(11): 1310-1313.

[6] Stassopoulou A, Caelli T. Building detection using bayesian networks [J]. International Journal of Pattern Recognition & Artificial Intelligence, 2000, 14(6): 715-733.

[7] Mallet C, Soergel U, Bretar F. Analysis of full-waveform LiDAR data for classification of urban areas [J]. Photogrammetrie Fernerkundung Geoinformation, 2008, 5(5): 337-349.

[8] Anguelov D, Taskar B, Chatalbashev V, et al. Discriminative learning of markov random fields for segmentation of 3D scan data [C]// Proc of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Washington DC: IEEE Computer Society, 2005: 169-176.

[9] Vu T T, Tokunaga M. Wavelet and scale-space theory in segmentation of airborne laser scanner [C]// Proc of International Conference on Computer Analysis of Images and Patterns. Cham: Springer, 2017: 95-107.

[10] Guo Bo, Huang Xianfeng, Zhang Fan, et al. Point cloud classification using JointBoost contextual information for feature reduction [J]. Acta Geodaetica et Cartographica Sinica, 2013, 42(5): 715-721.

[11] Wei Shufa. Scene Understanding based on Point Clouds and 2D Images [D]. Beijing: University of Chinese Academy of Sciences, 2016.

[12] Zhou Y, Tuzel O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. Washington DC: IEEE Computer Society, 2017: 4490-4499.

[13] Charles R Q, Su H, Mo K, et al. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. Washington DC: IEEE Computer Society, 2017: 652-660.

[14] Hovad J, Komarkova J, Sedlak P. Slope based grid creation using interpolation of LIDAR data sets [C]// Proc of International Conference on Software Engineering and Applications. Piscataway NJ: IEEE Press, 2013: 227-232.

[15] Hackel T, Savinov N, Ladicky L, et al. Semantic3D.net: a new large-scale point cloud classification benchmark [C]// ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2017: 91-98.

[16] Schmidhuber J. Deep learning in neural networks: an overview[J]. Neural Networks, 2015, 61(1): 85-117.

[17] Yue Chong, Liu Changjun, Wang Xiaofang. Research on point cloud data classification algorithm for high and steep slope based on multi-scale dimension features and SVM [J]. Geomatics and Information Science of Wuhan University, 2016, 41(7): 882-888.

[18] Moenning C, Dodgson N A. Fast Marching farthest point sampling (poster presentation) [C]// Proc of Eurographics. [S.l.]: Eurographics Association, 2003.

[19] Kingma D P, Ba J. Adam: a method for stochastic optimization [EB/OL]. (2017-01-30). https://arxiv.org/pdf/1412.6980.pdf.

[20] Tchapmi L, Choy C, Armeni I, et al. SEGCloud: semantic segmentation of 3D point clouds [C]// Proc of IEEE International Conference on 3D Vision. Picataway, NJ: IEEE Press, 2017: 537-547.

[21] Lawin F J, Danelljan M, Tosteberg P, et al. Deep projective 3D semantic segmentation [C]// Proc of International Conference on Computer Analysis of Images and Patterns. Cham: Springer, 2017: 95-107.

Note: Figure translations are in progress. See original paper for figures.