sensors

MDPI

Article

Clustering-Based Plane Segmentation Neural Network for Urban Scene Modeling

Hongjae Lee [1] and Jiyoung Jung [2,*]

[1] Department of Electronic Engineering, Kyung Hee University, Yongin-si 17104, Korea; jimmy9704@khu.ac.kr
[2] Department of Artificial Intelligence, University of Seoul, Seoul 02504, Korea
* Correspondence: jyjung@uos.ac.kr

Abstract: Urban scene modeling is a challenging but essential task for various applications, such as 3D map generation, city digitization, and AR/VR/metaverse applications. To model man-made structures, such as roads and buildings, which are the major components in general urban scenes, we present a clustering-based plane segmentation neural network using 3D point clouds, called hybrid K-means plane segmentation (HKPS). The proposed method segments unorganized 3D point clouds into planes by training the neural network to estimate the appropriate number of planes in the point cloud based on hybrid K-means clustering. We consider both the Euclidean distance and cosine distance to cluster nearby points in the same direction for better plane segmentation results. Our network does not require any labeled information for training. We evaluated the proposed method using the Virtual KITTI dataset and showed that our method outperforms conventional methods in plane segmentation. Our code is publicly available.

Keywords: point cloud plane extraction; 3D point clustering; 3D segmentation; urban mapping

Citation: Lee, H.; Jung, J. Clustering-Based Plane Segmentation Neural Network for Urban Scene Modeling. *Sensors* **2021**, *21*, 8382. https://doi.org/10.3390/s21248382

Academic Editors: Javier Alonso Ruiz, Angel Llamazares and Martin Lauer

Received: 25 October 2021
Accepted: 13 December 2021
Published: 15 December 2021

1. Introduction

Large-scale 3D reconstruction has been one of the most popular research topics in the field of robotics and computer vision for decades. In recent years, researchers and companies have been actively utilizing various sensors to obtain large-scale 3D reconstruction results over a certain level of accuracy with less time and cost. In particular, range sensors, such as LiDARs, have become very popular in outdoor scene modeling, and many researchers and engineers in related fields are now interested in utilizing the obtained large point clouds.

In this study, we focus on urban scene modeling, which is a challenging but essential task for 3D map generation for autonomous cars, city digitization, and AR/VR/metaverse applications. To model the ground and buildings, which are the major components in general urban scenes, we present a clustering-based plane segmentation neural network using 3D point clouds.

Plane segmentation from a point cloud is considered a foundation system in various computer vision and robotics fields, including object detection, model reconstruction, and map compression. Although many effective methods [1–3] have been proposed to segment a point cloud into planes, these methods focus on organized point clouds, which are depth values in grid arrays. However, point clouds obtained from LiDARs are usually unorganized, and organized point clouds obtained from depth cameras can easily be expressed in an unorganized form as well. Random sample consensus (RANSAC) and region-growing-based algorithms have been widely used to extract planes from unorganized 3D point clouds.

We propose a novel approach to segment unorganized 3D point clouds into planes, called hybrid K-means plane segmentation (HKPS). The whole plane segmentation procedure using the proposed system is shown in Figure 1, and the detailed plane segmentation process of HKPS is shown in Figure 2. We extracted man-made structures from the

unorganized input point cloud and voxelized the point using voxel downsampling as pre-processing. The proposed HKPS involves two steps: hybrid K-means clustering and plane merging. Hybrid K-means clustering performs clustering based on K-means++ [4], which is a K-means [5]-based clustering algorithm that has the advantage of careful seeding. K-means++ requires a parameter K, which is the number of clusters. We used PointNet [6] to estimate parameter K from the point cloud. PointNet is an unsupervised learning model that uses cosine distance for training. The final plane merging step combines the small planes that need to be merged. Our code is publicly available at [7].

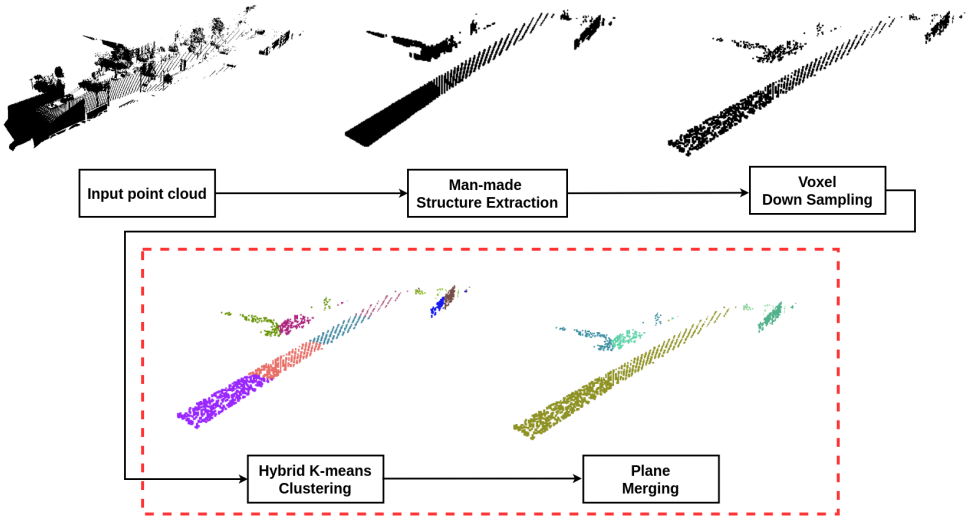


Figure 1. The plane segmentation procedure using the proposed system. We extract man-made structures from the unorganized input point cloud and voxelize the points using voxel down sampling as pre-processing. Then, the proposed hybrid K-means clustering method segments the point cloud into planes. Finally, the merging process combines the small planes that need to be merged.

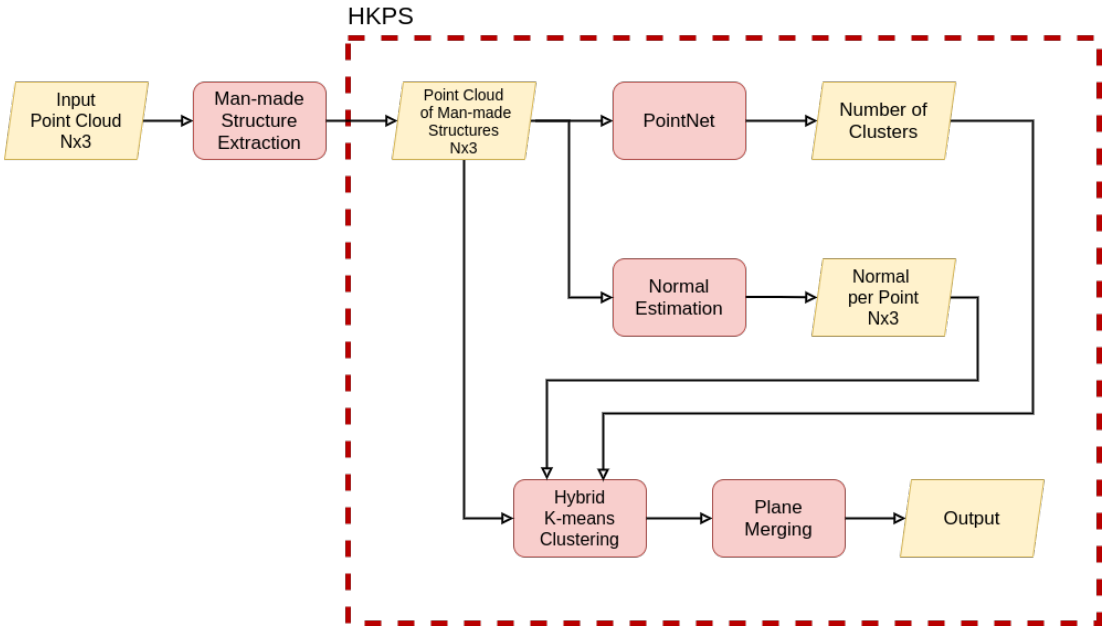


Figure 2. An overview of the proposed framework. PointNet [6] takes the input point cloud of man-made structure and outputs the number of clusters. We estimate the normal of each point, and then perform hybrid K-means clustering for plane segmentation (Sections 3.1 and 3.2). At the end of the network architecture, the plane merging process optimizes the number of planes (Section 3.3).

2. Related Work

Semantic segmentation in 2D images using convolutional neural network (CNN) has been extensively studied in several decades [8–10]. Semantic segmentation in 3D point clouds using CNN is a more challenging task in many aspects, and researchers have focused on the related problems [6,11,12], including the recent studies on fast and efficient 3D segmentation for urban scene analysis [13–15]. For the task of segmentation in point clouds, many previous works [16,17] used PointNet [6]. These previous approaches require the ground truth for training the neural network, which is usually difficult to obtain. Labeling 3D points for plane segmentation is a laborious task owing to the noise and ambiguity in point clouds. More recent methods that do not require ground truth are usually based on the Hough transform, RANSAC, and region growing.

2.1. Hough Transform

Hough transform [18] is a feature extraction technique used to find objects with a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima. The classical Hough transform is used to detect lines in binary images. The key idea behind the Hough transform is to change the coordinate system from image to Hough space. If the coordinate system is moved to the Hough space, lines passing through the points can be displayed as intersection points. Vosselman et al. [19] presented a plane extraction method from 3D point clouds based on the Hough transform and proposed a 3D Hough transform using normal vectors to enhance the speed and reliability of the traditional Hough transform. Limberger and Oliveira [20] proposed a plane extraction algorithm for point clouds by extending the kernel-based Hough transform [21]. The method creates a cluster using an octree and principal component analysis (PCA) and then votes for each cluster using a Gaussian kernel to extract planes. These methods are computationally fast and effective but tend to be sensitive to outliers.

2.2. RANSAC

Random sample consensus (RANSAC) was first presented by Fischler and Bolles [22]. RANSAC randomly selects some sample data and finds the model parameters to represent the sample data. In the case of finding planes to fit the sample points, the model parameters are the parameters in the plane equation. To find the best parameters, the method counts the number of inliers for the selected model, repeats the process several times, and finally selects the model with the largest number of inliers as the final result. Typically, to extract planes from 3D point clouds, we run RANSAC sequentially until no more planes can be found. RANSAC-based plane extraction algorithms are computationally more expensive but usually more robust to noise than Hough transform-based algorithms. However, these methods tend to combine adjacent planes into the same plane, specifically when the size of the adjacent plane is small.

Gotardo et al. [23] used a robust estimator to avoid premature convergence, which resulted in the preservation of small regions and edge locations, and accelerated the optimization process by a genetic algorithm. Schnabel et al. [24] predicted a model using only nearby peripheral points with an octree. The method detects planes, spheres, cylinders, cones, and tori and shows improved speed and scalability. Point sets with several million samples are robustly decomposed within less than a minute. Gallo et al. [25] proposed CC-RANSAC, which only considers the largest connected components of inliers to evaluate the fitness of a candidate plane. CC-RANSAC recovers the planar patches composing a typical step or ramp with higher accuracy than the traditional RANSAC algorithm. Despite recent advances, RANSAC based algorithms are still weak in distinguishing small clusters of points. They usually fail to produce reliable results in situations with two nearby patches of limited extent, where a single plane crossing through the two patches may contain more inliers than the “correct” models.

2.3. Region Growing

The region growing algorithm works by randomly selecting a starting point called a seed from the point cloud and then growing the region by adding points that satisfy a set of criteria. PCA is generally used to calculate the normal and curvature required for regional growth. Nurunnabi et al. [26] proposed a minimum covariance determinant (MCD)-based robust PCA to solve the outlier sensitivity problem of PCA. Vo et al. [27] created an octree and generated planar patches using PCA at each node of the octree. Their method measures the planarity of planar patches using the mean square distance, and subdivides the patches recursively with a threshold. After extracting planar patches, planes and points are merged using the difference between the normal angle of each plane and its distance to the point. Araújo and Oliveira [28] presented a plane extraction algorithm that does not require parameter tuning. Their method is similar to that of Vo et al., but it creates a planar patch in an octree by a bottom-up approach, whereby it keeps subdividing until planes can be detected. Planarity tests are performed using angular differences at each step to avoid parameter tuning. Region growing methods tend to fail in extracting a plane composed of few points similar to RANSAC-based methods. They also show difficulties in dividing two adjacent planes with different directions at the edges.

2.4. Clustering

Clustering algorithms can be classified as hierarchical or non-hierarchical. Most previous clustering-based plane segmentation algorithms are based on hierarchical clustering. Feng et al. [1] presented a plane extraction algorithm using hierarchical clustering. After extracting planar patches from the point cloud using PCA, the plane extraction algorithm constructs a graph to represent the patches and their relationship with each other. Then, agglomerative hierarchical clustering is performed using the mean-squared orthogonal point-to-plane fitting error. Finally, the roughly extracted planes are fine-tuned using the region growth in pixels. Schaefer et al. [3] utilized the maximum likelihood estimation (MLE) for agglomerative hierarchical clustering. When clustering, a plane is always extended in the direction that decreases the measurement likelihood of the scan. However, these methods can only be applied to organized point clouds.

Non-hierarchical clustering can be divided into density-based and center-based methods. Ester et al. [29] proposed a density-based clustering algorithm, named DBSCAN, based on the assumption that similar data are closely distributed to each other. Continuously dense regions are defined as clusters. DBSCAN performs well when dense and non-dense areas are easily distinguished. K-means clustering [5], a center-based method, performs clustering with K clusters while moving the centroids in a direction that minimizes the distance between the centroid and points in each cluster. However, it has a disadvantage in that it requires an appropriate parameter K to cluster the point cloud. To solve the problem of a random selection of initial centroids, Arthur and Vassilvitskii [4] proposed an improved version of the initial centroid selection and named the method K-means++. The improved method uses distance proportional probabilities to select a new centroid as far as possible from other centroids.

In this study, we propose a plane segmentation algorithm for unorganized point clouds based on the K-means++ non-hierarchical clustering algorithm. While the conventional K-means++ clustering algorithm only measures the Euclidean distance for clustering, the proposed hybrid K-means clustering method considers both the Euclidean distance and cosine distance for plane segmentation. Therefore, our method better distinguishes small plane segments located far from the sensor or partly occluded by obstacles, which RANSAC and region growing-based methods tend to miss. The requirement of setting K is also resolved by training a neural network without any labeled data to estimate an appropriate number of clusters for the point cloud.

3. Hybrid K-Means Plane Segmentation Neural Network

In this section, we first describe how the proposed hybrid K-means clustering segments a point cloud into planes. Next, we explain how PointNet estimates the number of planes. Finally, we demonstrate how to merge planes extracted from hybrid K-means clustering that need to be combined into the same plane.

3.1. Hybrid K-Means Clustering

K-means++ clustering is one of the most popular algorithms for segmenting data into k clusters. Nevertheless, K-means++ clustering is not suitable for segmenting a point cloud into planes because K-means++ clustering only considers the Euclidean distance for clustering. Spherical K-means clustering [30] is suitable for clustering high-dimensional data using the cosine distance between vectors. Inspired by K-means++ and spherical K-means clustering methods, we propose a hybrid K-means clustering method that clusters data into planes using both Euclidean distance and cosine distance. When hybrid K-means clustering measures the correlation between points, positional difference uses Euclidean distance, and normal difference uses cosine distance. Therefore, distant planes were clustered using Euclidean distances, and the directions of the planes were clustered using cosine distances. Therefore, distant planes are clustered using Euclidean distances, and the direction of the planes are clustered using cosine distances.

To estimate the normals of each point, we used K-nearest neighbors with the K-d tree provided by Open3D [31] for the entire point cloud. Open3D is an open-source Python library that supports rapid development of software that deals with 3D data. The K-d tree is a space-partitioning data structure that structures points in a K-dimensional space. The K-d tree is a useful data structure for finding the nearest neighbors. We set the maximum neighbors as 30, and the search radius as 3. We define the centroid of the i-th cluster as c_i and each point in the i-th cluster as $x_j \in S_i$. The goal of the hybrid K-means clustering is to find S_i that minimizes the overall variance V:

$$V = \sum_{i=1}^{K} \sum_{x_j \in S_i} d(x_j, c_i), \tag{1}$$

The function $d(x_j, c_i)$ computes the difference between x_j and c_i.

The algorithm starts by setting the initial cluster centroids c_i. K-means-based clustering algorithms have a disadvantage in that their performance varies greatly depending on how the initial value is selected. To overcome this disadvantage, we set the initial cluster centroids c_i using K-means++, which has the advantage of careful seeding. After selecting the initial centroids, we allocate each point to the most similar cluster by using $d(x_j, c_i)$, which is the difference between x_j and c_i. $d(x_j, c_i)$ is the sum of the Euclidean distance and cosine distance, as shown in Equation (2).

$$d(x_j, c_i) = E(x_j, c_i) + \lambda C(x_j, c_i) \tag{2}$$

$$E(x_j, c_i) = \sqrt{\|x_j - c_j\|^2} \tag{3}$$

$$C(x_j, c_i) = 1 - \frac{x_j \cdot c_j}{|x_j||c_j|} \tag{4}$$

$E(x_j, c_i)$ computes the Euclidean distance between positional differences and $C(x_j, c_i)$ computes the cosine distance between normal differences, as shown in Equations (3) and (4). λ is a parameter that determines the ratio between the Euclidean distance and the cosine distance. As λ grows, parallel planes are classified as the same plane, owing to the increased influence of the cosine distance. In contrast, if λ is too small, the point cloud cannot cluster as planes. For the clustering points into planes, we chose λ = 60. After allocating

each point to the most similar cluster, we rearrange the centroid c_i of cluster S_i using the center of gravity of points in S_i. Equation (5) shows how to rearrange the centroid c_i.

$$c_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j \tag{5}$$

The proposed hybrid K-means clustering repeats the above two steps until the centroids converge.

3.2. Parameter Estimation

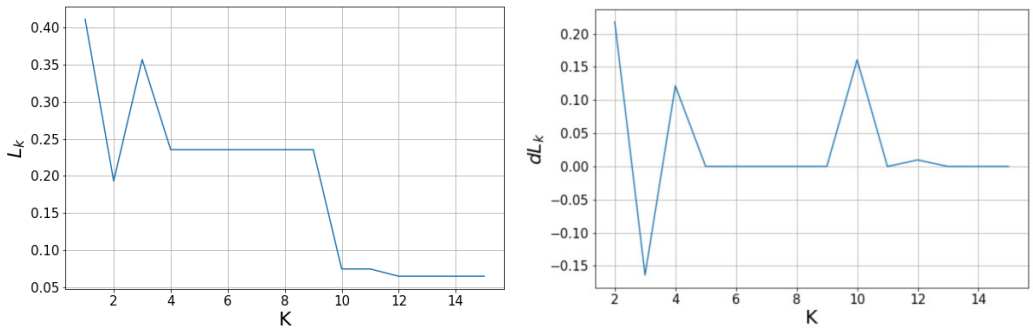
The proposed hybrid K-means clustering method has the disadvantage that K, the number of planes to be extracted, must be passed as an input parameter. We used PointNet to automatically determine parameter K from the point cloud. PointNet takes an unorganized point cloud represented in the format of $\mathbb{R}^{N\times 3}$ as the input, where N is the number of points. PointNet is a widely used library in the field of computer vision for classification and segmentation by analyzing the characteristics of input point clouds. Our purpose is to train PointNet to infer parameter K, which is the number of planes. In general, a pre-processing process for labeling a dataset is essential for obtaining labeled point clouds. However, in this study, we propose an unsupervised training method using hybrid K-means clustering.

After hybrid K-means clustering, the input points are segmented into K planes. The average cosine distance L_k between each plane's centroid c_i and their assigned M points $X = \{x_1, x_2, ..., x_M | x_j \in S_j\}$ can be defined by

$$L_k = \sum_{i=1}^{k} \frac{1}{M} \sum_{x_j \in S_j}^{M} C(x_j, c_i). \tag{6}$$

As K increases, L_k decreases when the point cloud is divided into planes, as shown in Figure 3a. We use dL_k, which is the difference of L_k, which can be obtained as the difference between L_{k-1} and L_k.

$$dL_k = L_{k-1} - L_k \tag{7}$$



(**a**) (**b**)

Figure 3. (**a**) Average cosine-distance value of a point cloud calculated by Equation (4) according to K, the number of planes. (**b**) Difference of average cosine-distance calculated by Equation (7). If K is sufficient to cluster point cloud into planes, we can see that the difference of average cosine-distance converges close to zero and no longer changes. By using difference of average cosine-distance, we can obtain the appropriate parameter K.

If K is sufficient to cluster the input point cloud into planes, we can see that the difference in the average cosine distance converges close to zero and no longer changes, as shown in Figure 3b. Therefore, the smallest parameter K, in which dL_k begins to converge close to zero is the appropriate number of planes for the hybrid K-means clustering method. We experimentally define that dL_k converges close to zero when $|dL_k| < 0.1$ is satisfied.

Because of L_k and dL_k, as shown in Figure 3, the value of K for the with-noise-00 point cloud was automatically selected as 11. PointNet can be trained in an unsupervised manner using the K obtained by the above method. PointNet requires a parameter that sets the maximum limit of K, which we set to 15.

The parameter estimation for hybrid K-means clustering using PointNet has an advantage in terms of computational time. It takes a lot of time to train PointNet, but after training it is more time efficient than without using PointNet, which involves performing hybrid K-means up to the maximum limit of K each time to obtain the appropriate parameter K.

3.3. Plane Merging

In the previous sections, we clustered a point cloud into planes with an appropriate number of planes. Figure 1 illustrates the process of the proposed approach. The proposed hybrid K-means clustering sometimes oversegments input points into two or more planes that eventually need to be grouped into the same plane, owing to the Euclidean-distance term in Equation (3). Figure 1 shows that hybrid K-means clustering segments the road into four planes, even though it can be merged into a single plane. Therefore, we present a plane-merging algorithm that combines planes that should be merged into the same plane. To merge different planes, the following two conditions must be satisfied: angular similarity and distance limit. To find a plane S_j that satisfies the condition of angular similarity with another plane S_i, we compute the cosine distance between centroid c_i of plane S_i and centroid c_j of plane S_j. If the cosine distance is smaller than threshold σ, we determine that the angular similarity condition is satisfied. We create a new set I that consists of planes satisfying the condition of angular similarity with plane S_i, as shown in Equation (8).

$$I = \{S_j \in S | C(c_i, c_j) < \sigma\} \tag{8}$$

If we only consider angular similarity, different planes facing in the same direction apart from each other can be merged into the same plane. To avoid such situations, we also consider the distance limit condition. If the closest points in planes S_j and S_i are closer than threshold ω, we consider that the planes satisfy the distance limit condition. Such planes are collected from I to form set H, as described in Equation (9).

$$H = \{S_j \in I | |SE(S_i, S_j)| > 1\} \tag{9}$$

$$SE(S_i, S_j) = \{x_i \in S_i | E(x_i, x_j) < \omega\} \tag{10}$$

$|\cdot|$ represents the cardinality of the set. $SE(S_i, S_j)$ is the set of points belonging to planes S_i, where the Euclidean distance between points is closer than threshold ω, as in Equation (10). The members of H merge with plane S_i to form a new merged plane S_m. The above process is repeated until there are no additional planes to merge. We set $\sigma = 0.1$ and $\omega = 4$.

4. Results

To evaluate the proposed hybrid K-means plane segmentation (HKPS) method, we used the photorealistic synthetic Virtual KITTI Dataset [32], which closely mimics the real-world KITTI dataset [33]. The Virtual KITTI dataset was labeled with 13 different labels. The dataset obtains a semantically annotated 3D point cloud by projecting a given 2D depth into a 3D space. We extracted man-made structures, such as roads and buildings, from the Virtual KITTI dataset to evaluate the plane segmentation performance. This is because these man-made structures can be expressed in planes, which are effective in composing an urban scene model. The Virtual KITTI dataset consisted of unorganized point clouds.

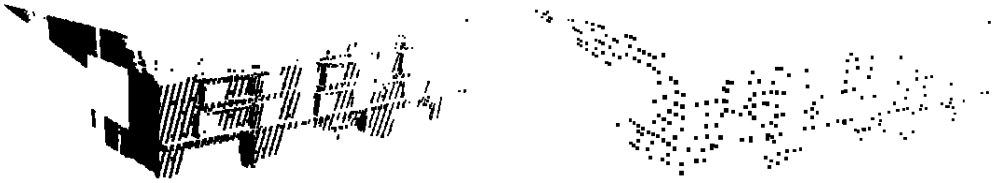
For comparison evaluation, we tested the following most widely used plane segmentation techniques: RANSAC, region growing, and K-means clustering-based techniques. For RANSAC, we used the pyRANSAC-3D library [34] to fit the planes in a point cloud.

For region growing, we used the latest technology, RSPD [28]. RSPD had the advantage of extracting planes robustly against noise, and it exhibited better performance in various indoor environments than the existing plane segmentation techniques. For K-means clustering, we used K-means++, which is most similar to our technique among the previous methods. In the comparison experiment with K-means++, PointNet was trained using K-means++ in the same way as HKPS to determine the number of clusters.

PyTorch was used to implement the proposed HKPS. The training was performed on a single NVIDIA 2080Ti GPU. We used the Open3D library [28] to find the normal vectors in the point cloud and implement voxel downsampling.

4.1. Voxel Down Sampling

The point cloud is partially composed of different densities owing to the distance from the sensor, angle, and obstacles, as shown in Figure 4a Because hybrid K-means clustering rearranges the centroids using the center of gravity of each cluster, the centroids tend to be rearranged in the direction with a greater density of the points. To solve this problem, we perform voxel downsampling, and the result is shown in Figure 4b.



(**a**) A raw point cloud (**b**) The point cloud after voxel down sampling

Figure 4. The same point cloud (**a**) before and (**b**) after voxel down sampling. The density of the point cloud becomes uniform and noise is removed.

A voxel downsample filter combines a three-dimensional voxel grid on a point cloud. The points inside each voxel were downsampled to the center of each voxel. The process of voxel downsampling reduces the number of points to align with a uniform density. This process not only benefits from balanced centroid rearrangement for hybrid K-means clustering but also reduces noise and the number of input points to PointNet. Reducing the number of points enables efficient use of memory. After voxel downsampling, we extracted 1600 points from each point cloud for PointNet training.

4.2. Performance Evaluation

Two types of datasets are used to compare the plane segmentation performance of our proposed method, HKPS, with previous works: *with-noise* and *without-noise* datasets. The *with-noise* dataset is the Virtual KITTI dataset after voxel downsampling with uniform density. For the *without-noise* dataset, we deleted small clusters of points, such as parts of trees, vehicles, and unknown objects, from the *with-noise* dataset. PointNet was trained with 70 scenes from the *with-noise* dataset, and the performance evaluation experiment was performed with 20 scenes from each of the *with-noise* and *without-noise* datasets.

We used the performance metrics proposed by Hoover et al. [35] for quantitative comparison with previous works, and the results are shown in Tables 1 and 2. When plane S_i of the ground truth and plane S_j segmented by the evaluation method overlap threshold T% or more, we classify the plane as **correct detection** and use threshold $T = 80$. **Over segmentation** means that a plane is segmented with a greater number than it should be, and **under segmentation** means that it is segmented with a smaller number of planes. **Missed** means that plane S_i of ground truth does not participate in any correct detection, over segmentation, or under segmentation. **Noise** means that plane S_j segmented by evaluation method does not participate in any correct detection, over segmentation, or under segmentation.

Table 1. Average result of *without-noise* Virtual KITTI Dataset. Best results are highlighted in bold.

Method	Correct Detection [%]	Under-Segmentation [%]	Over-Segmentation [%]	Missed [%]	Noise [%]	Avg. Cosine Distance
HKPS (ours)	**84.26**	3.18	11.31	**0.19**	**0.1**	0.0337
K-means++ [4]	70.56	0.76	17.27	9.41	9.40	0.0820
RANSAC [34]	72.36	**0.06**	6.97	13.28	13.69	0.0606
RSPD [28]	72.54	12.35	**0.07**	10.05	14.78	**0.0182**

Table 2. Average result of *with-noise* Virtual KITTI Dataset. Best results are highlighted in bold.

Method	Correct Detection [%]	Under-Segmentation [%]	Over-Segmentation [%]	Missed [%]	Noise [%]	Avg. Cosine Distance
HKPS (ours)	**83.80**	**1.46**	9.50	**2.76**	**2.10**	0.0315
K-means++ [4]	51.87	14.70	17.93	11.68	10.76	0.0916
RANSAC [34]	70.93	2.42	4.14	18.30	16.44	0.1284
RSPD [28]	65.96	9.94	**1.21**	17.11	18.69	**0.0106**

To increase the objective validity of the results, we propose an average cosine distance (ACD) as an additional performance evaluation metric.

$$ACD = \sum_{i=1}^{N} \frac{1}{M} \sum_{x_j \in S_j}^{M} C(x_j, c_i). \tag{11}$$

The average cosine distance represents the angular difference between the normal of each point and the centroid normal of the plane. N is the number of planes extracted from this method.

It is necessary to define the ground truth suitable for plane segmentation to evaluate the proposed plane segmentation technique. Because the ground truth was not available for the Virtual KITTI dataset, we created it by labeling 3D point clouds using RGB 2D images provided by the dataset. The ground truth is shown in the last lines of Figures 5 and 6.

K-means++ clustering achieved the lowest correct detection in both the *with-noise* and *without-noise* datasets, and it achieved the largest average cosine distance in the *without-noise* dataset and the second largest in the *with-noise* dataset. As shown in Figure 5, K-means++ fails to segment the corners of buildings and falsely segments buildings and roads into the same cluster if the distance between the building and the road is small. This is because K-means++ clustering only considers the Euclidean distance when segmenting clusters; therefore, it is not suitable for clustering point clouds into planes.

RANSAC showed large values of missed and noise in both datasets, and the average cosine distance was the largest in the *with-noise* dataset, which was significantly higher than that in the *without-noise* dataset. This is because the points in the *with-noise* dataset are widely distributed in the horizontal direction of the surface owing to noise. As shown in the *with-noise* scene in Figure 5, RANSAC creates planes perpendicular to the buildings because the method segments planes without considering their normals. The performance depends greatly on the initial seed selection. Because of these disadvantages, RANSAC cannot extract small planes, and the planes that are scanned only partially by the sensor are estimated as noise.

RSPD has the second highest correct detection in the *without-noise* dataset, but it drops significantly in the *with-noise* dataset. Despite the low value of correct detection, RSPD has the smallest average cosine distance compared to the other methods. RSPD determines outliers using angle difference to extract planes from various datasets without adjusting parameters; however, the allowable angle difference is tightly set. As shown in Figure 6, the planes composed of a few points in the *with-noise* dataset and the corners that are difficult to be segmented into planes using angle differences were classified as outliers.

HKPS has the highest correct detection in both datasets, and its average cosine distance is the second best. This means that the proposed HKPS segments the point cloud into appropriate planes. The values of missed and noise are significantly lower than those of other methods, as the clustering-based method of HKPS can segment the planes with a small number of points. It is shown that HKPS is suitable for segmenting planes in urban scenes, where many parts of planes are obscured by obstacles, such as vehicles and trees.

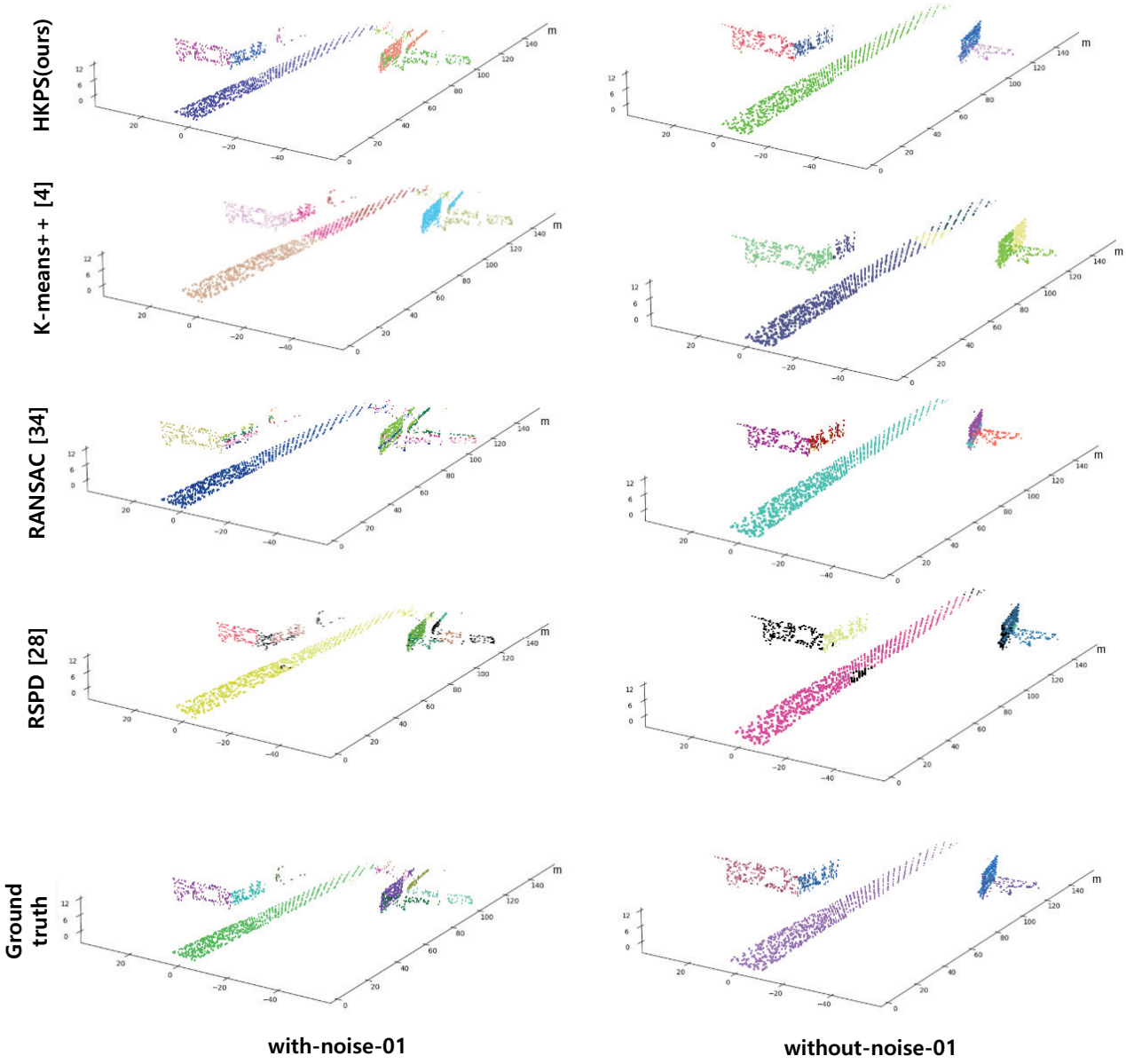


Figure 5. Performance comparison results using scene 01 from *with-noise* Virtual KITTI Dataset and *without-noise* Virtual KITTI Dataset. Different planes are displayed with different colors. The outliers of RANSAC and RSPD are colored in black.

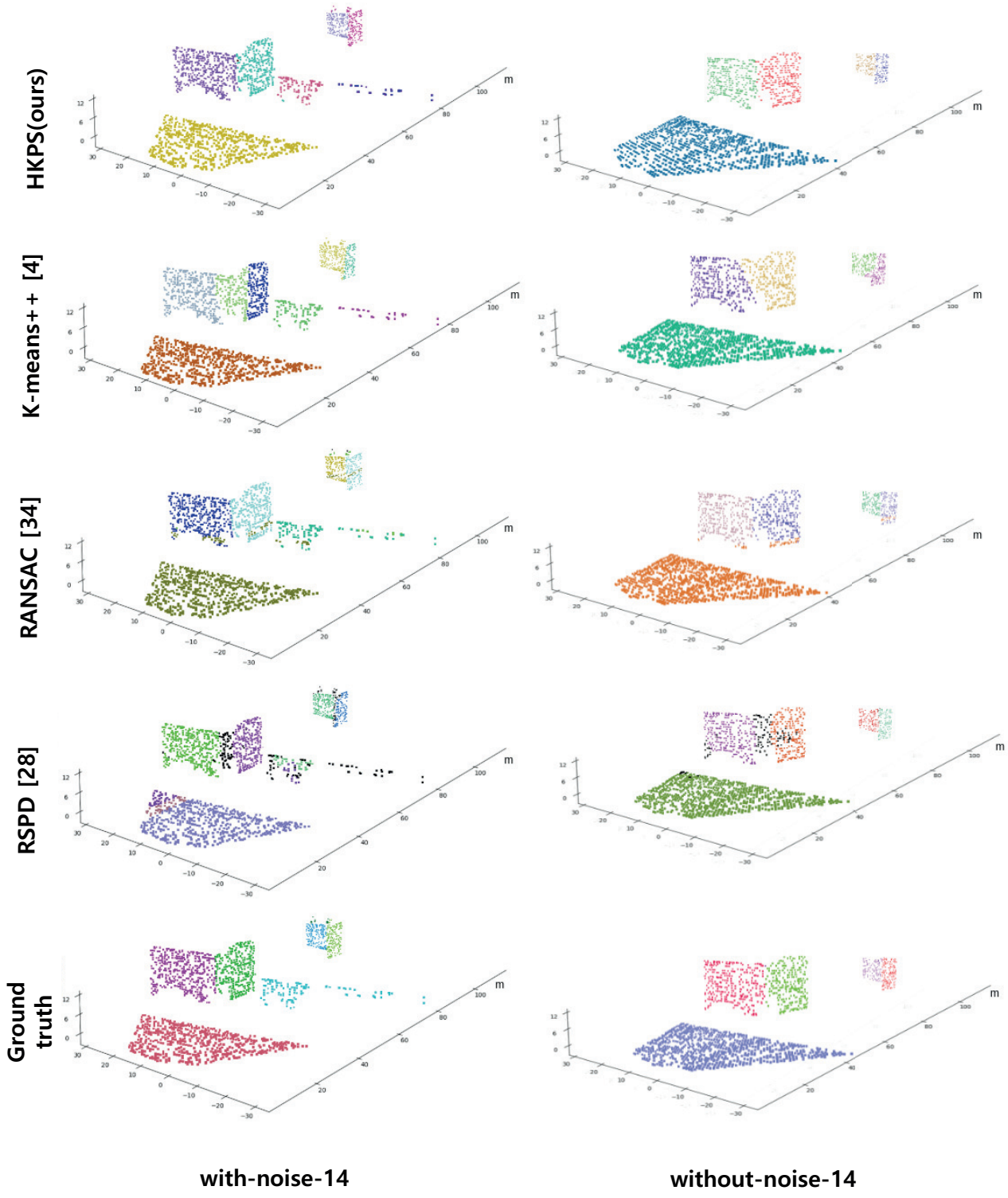


Figure 6. Performance comparison results using scene 14 from *with-noise* Virtual KITTI Dataset and *without-noise* Virtual KITTI Dataset. Different planes are displayed with different colors. The outliers of RANSAC and RSPD are colored in black.

4.3. Scalability

We measured the performance change with the number of points in the input point cloud. Figure 7 shows a graph of the correct detection change according to the change in the number of input points. There may be more points in the urban scene, but we measured performance using 70,000 to 200,000 points.

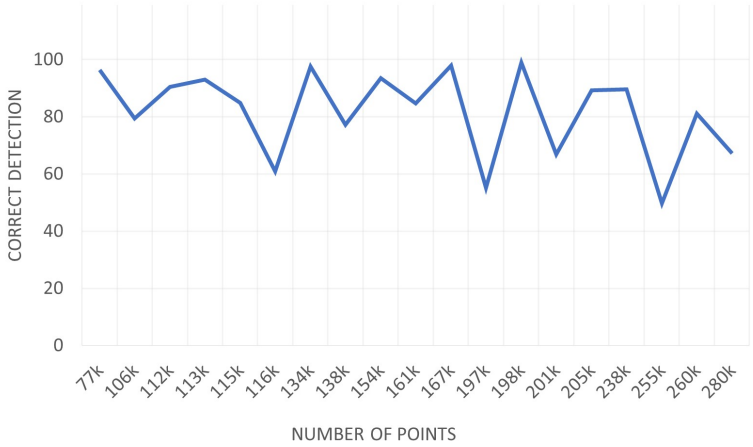


Figure 7. Correct detection metric difference according to the number of points. It can be seen that there is not much change in performance depending on the number of points.

The number of input points was the number before voxel downsampling, and 1600 points were extracted for training PointNet after voxel downsampling. As the number of points increases, it does not appear to be a significant drop in performance. This proves that our method is suitable for plane segmentation in an urban scene composed of a large number of points.

5. Conclusions

We presented an effective neural network for segmenting an urban scene point cloud into planes. Our plane segmentation method has two major differences compared to the previous methods. First, we combine two types of coordinate differences: Euclidean and cosine differences. Our experiments demonstrate that combining the two coordinate differences translates into superior accuracy plane clustering results. The proposed method segments the planes composed of a small number of points that are far from the sensor or partly obscured by obstacles. Second, we use PointNet as a method to automatically determine the number of planes in the point cloud. Previous K-means clustering-based methods applied various methods to find the appropriate number of clusters K. In addition, to find an appropriate K, it was necessary to perform clustering several times while changing K. However, the proposed method uses PointNet to estimate the number of clusters so that the plane can be segmented by performing clustering only once. Because of this, parameter estimation using PointNet has an advantage in terms of computational time.

Owing to the effective results, we suggest several extensions of the proposed plane segmentation method, HKPS. We plan to extend PointNet to enable parameter estimation that is suitable for various datasets. In addition, we are working on extending HKPS to a network that can extract other primitive types, such as spheres and cones.

Author Contributions: Conceptualization, J.J.; methodology, H.L.; software, H.L.; validation, H.L.; formal analysis, H.L. and J.J.; investigation, H.L. and J.J.; resources, J.J.; data curation, H.L.; writing—original draft preparation, H.L.; writing—review and editing, J.J.; visualization, H.L.; supervision, J.J.; project administration, J.J.; funding acquisition, J.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a grant from National Research Foundation of Korea (NRF-2020R1C1C1008726).

Data Availability Statement: The data presented in this study are openly available at reference number [32]. Our code is openly available at reference number [7].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Feng, C.; Taguchi, Y.; Kamat, V.R. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 6218–6225.
2. Torr, P.H.; Zisserman, A. MLESAC: A new robust estimator with application to estimating image geometry. *Comput. Vis. Image Underst.* **2000**, *78*, 138–156. [CrossRef]
3. Schaefer, A.; Vertens, J.; Büscher, D.; Burgard, W. A maximum likelihood approach to extract finite planes from 3-D laser scans. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 72–78.
4. Arthur, D.; Vassilvitskii, S. *k-Means++: The Advantages of Careful Seeding*; Technical report; Stanford University: Stanford, CA, USA, 2006.
5. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [CrossRef]
6. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep learning on point sets for 3D classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
7. Lee, H. HKPS. 2021. Available online: https://www.github.com/jimmy9704/plane-segmentation-network (accessed on 1 December 2021).
8. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
9. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]
10. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 834–848. [CrossRef] [PubMed]
11. Su, H.; Maji, S.; Kalogerakis, E.; Learned-Miller, E. Multi-view convolutional neural networks for 3D shape recognition. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 945–953.
12. Li, L.; Sung, M.; Dubrovina, A.; Yi, L.; Guibas, L.J. Supervised fitting of geometric primitives to 3D point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 2652–2660.
13. Hůlková, M.; Pavelka, K.; Matoušková, E. Automatic classification of point clouds for highway documentation. *Acta Polytech.* **2018**, *53*, 165–170. [CrossRef]
14. Alonso, I.; Riazuelo, L.; Montesano, L.; Murillo, A.C. 3D-mininet: Learning a 2D representation from point clouds for fast and efficient 3D lidar semantic segmentation. *IEEE Robot. Autom. Lett.* **2020**, *5*, 5432–5439. [CrossRef]
15. Zhu, X.; Zhou, H.; Wang, T.; Hong, F.; Ma, Y.; Li, W.; Li, H.; Lin, D. Cylindrical and asymmetrical 3D convolution networks for lidar segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 9939–9948.
16. Paigwar, A.; Erkent, Ö.; Sierra-Gonzalez, D.; Laugier, C. Gndnet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2021; pp. 2150–2156.
17. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space; In Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017.
18. Hough, P.V. Method and Means for Recognizing Complex Patterns. US Patent 3069654, 18 December 1962.
19. Vosselman, G.; Gorte, B.G.; Sithole, G.; Rabbani, T. Recognising structure in laser scanner point clouds. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2004**, *46*, 33–38.
20. Limberger, F.A.; Oliveira, M.M. Real-time detection of planar regions in unorganized point clouds. *Pattern Recognit.* **2015**, *48*, 2043–2053. [CrossRef]
21. Fernandes, L.A.; Oliveira, M.M. Real-time line detection through an improved Hough transform voting scheme. *Pattern Recognit.* **2008**, *41*, 299–314. [CrossRef]
22. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
23. Gotardo, P.F.; Bellon, O.R.P.; Silva, L. Range image segmentation by surface extraction using an improved robust estimator. In Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Madison, WI, USA, 18–20 June 2003; Volume 2, pp. 33–38.
24. Schnabel, R.; Wahl, R.; Klein, R. Efficient RANSAC for point-cloud shape detection. In *Computer Graphics Forum*; Wiley Online Library: Hoboken, NJ, USA, 2007; Volume 26, pp. 214–226.
25. Gallo, O.; Manduchi, R.; Rafii, A. CC-RANSAC: Fitting planes in the presence of multiple surfaces in range data. *Pattern Recognit. Lett.* **2011**, *32*, 403–410. [CrossRef]
26. Nurunnabi, A.; Belton, D.; West, G. Robust segmentation in laser scanning 3D point cloud data. In Proceedings of the 2012 International Conference on Digital Image Computing Techniques and Applications (DICTA), Fremantle, WA, Australia, 3–5 December 2012; pp. 1–8.

27. Vo, A.V.; Truong-Hong, L.; Laefer, D.F.; Bertolotto, M. Octree-based region growing for point cloud segmentation. *ISPRS J. Photogramm. Remote Sens.* **2015**, *104*, 88–100. [CrossRef]
28. Araújo, A.M.; Oliveira, M.M. A robust statistics approach for plane detection in unorganized point clouds. *Pattern Recognit.* **2020**, *100*, 107115. [CrossRef]
29. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the kdd, Portland, OR, USA, 2–4 August 1996; Volume 96, pp. 226–231.
30. Dhillon, I.S.; Modha, D.S. Concept decompositions for large sparse text data using clustering. *Mach. Learn.* **2001**, *42*, 143–175. [CrossRef]
31. Zhou, Q.Y.; Park, J.; Koltun, V. Open3D: A Modern Library for 3D Data Processing. *arXiv* **2018**, arXiv:1801.09847.
32. Gaidon, A.; Wang, Q.; Cabon, Y.; Vig, E. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In Proceedings of the CVPR, Las Vegas, NV, USA, 27–30 June 2016. Available online: https://europe.naverlabs.com/research/computer-vision/proxy-virtual-worlds-vkitti-1/ (accessed on 1 October 2021).
33. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of the IEEE International Conference on Computer Vision (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
34. Mariga, L. pyRANSAC-3D. 2021. Available online: https://github.com/leomariga/pyRANSAC-3D (accessed on 1 October 2021).
35. Hoover, A.; Jean-Baptiste, G.; Jiang, X.; Flynn, P.J.; Bunke, H.; Goldgof, D.B.; Bowyer, K.; Eggert, D.W.; Fitzgibbon, A.; Fisher, R.B. An experimental comparison of range image segmentation algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.* **1996**, *18*, 673–689. [CrossRef]