IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 17, 2020, accepted February 7, 2020, date of publication February 12, 2020, date of current version February 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2973580

Robust Segmentation of Building Planar Features from Unorganized Point Cloud

PENGJU TIAN[1,2], (Member, IEEE), XIANGHONG HUA[1,2], KEGEN YU[3], (Senior Member, IEEE), AND WUYONG TAO[1,2]

[1]School of Geodesy and Geomatics, Wuhan University, Wuhan 430079, China
[2]Hazard Monitoring and Prevention Research Center, Wuhan University, Wuhan 430079, China
[3]School of Environment Science and Spatial Informatics, China University of Mining and Technology, Xuzhou 221116, China

Corresponding author: Xianghong Hua (xhhhua@sgg.whu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 41674005, Grant 41374011, and Grant 41501502, in part by the Open Foundation of Postdoctors Innovation and the Practice Base of Wuhan Geomatics Institute under Grant WGF 2016002, and in part by the Doctoral Fund of Ministry of Education of China under Grant 2018M632909.

ABSTRACT This paper presents a novel method for segmentation of planar feature from unorganized point cloud based on 2D Hough Transform and octree. Given the input point cloud, three steps are performed to segment planar features. Firstly, the original point cloud is sampled and projected to the X-Y plane, and an extended 2D Hough transform algorithm is employed to extract the line segments. The selecting weight iteration method is used to calculate line equations and endpoint coordinates of those line segments. The space geometric equations of the vertical planes are then determined. Secondly, the octree structure of the original point cloud is established, and then the exact endpoint coordinates of the line segment are used to design a cube perpendicular to the X-Y plane and all points held by the cube are extracted. The distance from each of the extracted points inside the cube to its corresponding facade is calculated, if the distance is less than the predefined threshold, the point is regarded as a point inside the facade. Finally, all the facade points are removed from the original point cloud, and remaining point cloud is sampled and projected to the X-Z plane. The above process is repeated to extract horizontal planes. Evaluation experiments are performed by analyzing the performance of our method in four different scenes. The experimental results indicate that the proposed algorithm is suitable for segmentation of building planar features in different scenes. A comparison with competing techniques shows that our approach is considerably faster and scales significantly better than previous ones.

INDEX TERMS Hough transform, unorganized point clouds, planar feature, selecting weight iteration, octree.

I. INTRODUCTION

In recent years, light detection and ranging (LiDAR) technology has been rapidly developed. The popularization of laser scanners has led to an increasing growth in the sizes of the available datasets, and point clouds containing tens of millions of samples are now commonplace. LiDAR can quickly and directly acquire 3D geospatial information, which is generally termed as point cloud [1]. As one of the most common features in man-made objects, plane plays an important role in reverse engineering [2]–[4], object recognition [5], [6], augmented reality [7], and heritage preservation [8], [9]. Hence segmentation of planar features from unorganized point cloud has been a research focus, and a large number of methods have been developed over the last few decades by researchers from different fields, the most important of which can generally be classified into four categories: 3D Hough transform, Random Sample Consensus (RANSAC), Region Growing, and deep learning-based methods. This section discusses these algorithms and their various optimization techniques used to detect planes in point clouds.

The 3D Hough transform [10] is a method for detecting parameterized objects, typically containing lines [11], [12], spheres [13], [14], and planes [15]. Given an object and the type of shape to be identified, the algorithm will perform a voting process in the parameter space to determine the shape

The associate editor coordinating the review of this manuscript and approving it for publication was Yakoub Bazi.

of the object, which is determined by the local maximum in the accumulator space [16]. Currently, the widely used version of the algorithm is generalized Hough transform (GHT) proposed by Duda and Hart [17], making use of angle-radius rather than slope-intercept parameters to simplify the computation further. The Standard Hough transform (SHT) for plane detection iterates over each sample in the point cloud and casts votes in the accumulator for all possible planes passing through the sample. Due to the high computational cost of the SHT, many algorithms have been proposed to accelerate its voting procedure. The Probabilistic Hough transform (PHT) [18] randomly selects a small subset of points in the point cloud and uses them for voting. The selection of optimal subset depends on many characteristics of the point cloud, which is not a simple task. The Adaptive Probabilistic Hough transform (APHT) [19] can monitor changes in the ranks of peaks in the parameter space during the voting procedure. The APHT is sensitive to noise and may lead to the false detection of planes that do not present in the point cloud. The Progressive Probabilistic Hough transform (PPHT) [20] tries to avoid the influence of random noise by only detecting structures whose number of votes exceeds a threshold defined as a percentage of the total number of votes. The Randomized Hough transform(RHT) [21] reduces the SHT's voting processing time by exploiting the fact that a plane can be defined by three non-collinear points. In addition, Ogundana et al. [22] proposed an optimization model of 3D sparse matrix to accelerate the voting process.

Another important algorithm for performing shape detection is the Random Sample Consensus (RANSAC) [23]. The RANSAC algorithm is used for extracting shapes by randomly choosing minimal number of points to construct candidate shape primitives. The candidate shapes are checked against all points in the point cloud to determine the parameters that best fit the shapes [24]. The method has been adapted to segment 3D point clouds. For example, Schnabel *et al.* [25] adapted RANSAC for precise and fast plane extraction, in which a point cloud with normal vectors was used to verify all sampling points before assigning a candidate shape. An octree was then employed to efficiently extract sampling points. Similarly, Chen *et al.* [26] improved the RANSAC algorithm by the localized sampling to segment the polyhedral rooftop primitives and then to separate the coplanar primitives with a region growing based triangulated irregular network (TIN). Awwad *et al.* [27] modifies the RANSAC algorithm by using a clustering technique to divide the point cloud into small clusters based on normal vectors of the points.

The third class of techniques used to identify shapes in point clouds is Region Growing [28], [29]. Compared with the first two methods, the Region Growing algorithm relies on the similarity between points and related attributes. The core is to select the seed points and determine the coplanar conditions between points. Nurunnabi et al. [30] selected the point with smallest curvature as the seed point, and used the relationship between the angles of the tangent planes and the distance between the points as the coplanar condition. Vieira and Shimada [31] firstly removed points along sharp edges using a curvature threshold. Median filtering was then performed to reduce noise, and the remaining points are considered as seed points. The Region Growing algorithm is less efficient due to the requirement of calculating the normal vector and performing neighborhood search based on points. In order to improve the processing efficiency, Vo et al. [32] adopted octree structure to achieve plane extraction using voxel region growth.

Deep learning-based methods have won numerous contests in pattern recognition and machine learning [33]. Boulch *et al.* [34] adopted paired RGB and range images to generate images of 3D scenes. Guerry *et al.* [35] presented an approach to obtain dense 3D point markers for semantic recognition in the context of robotics. Lateef and Ruichek [36] transformed unorganized point cloud into voxels and then carried out feature learning to realize semantic segmentation of point cloud. Graham [37] designed a sparse convolution network for 3D segmentation. Qi *et al.* [38] proposed the PointNet to deal with tasks such as classification and segmentation.

As discussed above, the 3D Hough transform and RANSAC are well established as robust methods for segmenting 3D point clouds. However, these algorithms have some disadvantages. First, many spurious planes that do not exist in reality may be generated. Second, the segmentation quality is sensitive to some point cloud characteristics (e.g., density, positional accuracy, and noise). Third, these algorithms perform poorly on large datasets or those with complex geometries. The 3D Hough transform especially requires significant processing time and high memory consumption for large datasets. To compensate for these defects, most existing planar segmentation methods have to calculate the normal of each point in the point cloud. Besides, many artificial input threshold parameters are also required, which are not always easy to be determined. Although deep learning-based methods are developing rapidly, these methods require a large number of samples and are very time consuming. When faced with the rapid extraction of large-scale unorganized point clouds, they have no significant advantages over traditional methods. Considering that man-made objects are mainly composed of vertical and horizontal planes, this paper presents a novel method for segmentation of building planar features from unorganized point cloud based on 2D Hough transform and octree. The algorithm proposed in this paper has some clear geometric meanings, which are convenient to implement. The main contributions of the paper are summarized as follows:

(1) A novel method for segmentation of building planar feature from unorganized point cloud is proposed that does not require connectivity information or information about normal vector.

(2) The proposed method can effectively segment coplanar planes that are not connected and avoid the problem of misidentification of planar pseudo-intersection regions.

(3) The proposed method scales well with the size of the datasets and is robust to the presence of noise. Besides, the parameters in the algorithm are adaptively estimated.

The remainder of this paper is organized as follows: the proposed method is presented with details in section II. In section III, the experiments are described, experimental results are demonstrated, performance comparison is performed and discussions are provided. The conclusions are summarized in section IV.

II. THE PROPOSED SEGMENTATION METHOD

The method proposed in this paper is based on a very intuitive phenomenon in the real world, namely most human-made objects have flat surfaces and those flat surfaces are basically perpendicular or parallel to the ground. In other words, they are axis-aligned. As shown in Fig.1, if we project point cloud to the ground (the X-Y plane), those facades will exhibit a dense linear distribution, which is significantly different from other objects. In order to achieve better segmentation, some pre-processing operations are performed and described in detail in the subsequent content. In the passing decade, 2D line segment extraction has been well studied and several efficient algorithms are developed including the 2D Hough Transform [12], the CANNAY [39] and the widely used LSD [40]. With these techniques we can easily extract 2D segments and once the spatial equation and endpoint coordinates of the line segment are determined, the spatial equation of the vertical plane will be determined accordingly. Then an octree of the original point cloud is employed to obtain points near the plane. Finally, the points on vertical planes are removed from the original point cloud, and the remaining point cloud is sampled and then projected to the X-Z plane to extract horizontal planes.

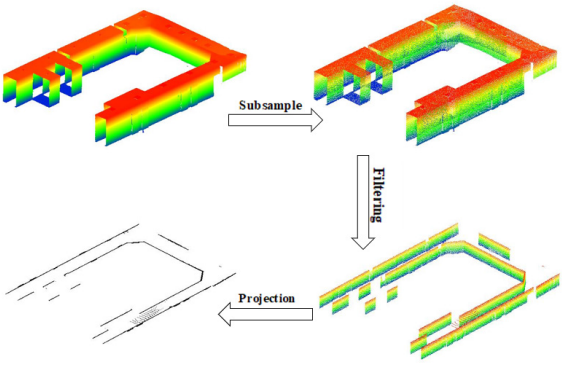


FIGURE 1. A demonstration of the data pre-processing process. Point cloud is colored in the Z-axis direction.

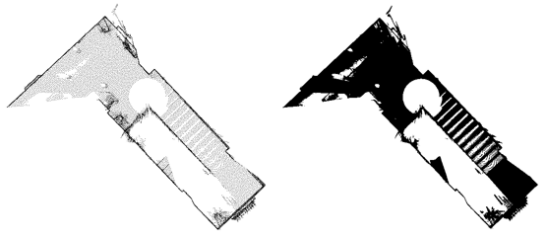
The algorithm proposed in this paper is mainly aimed at the segmentation of vertical and horizontal planes. Considering that the two processes are very similar, the former is taken as an example, and we will describe the method in detail in the following subsections.

A. DATA PRE-PROCESSING

The building point clouds are usually very dense and data processing can be quite time consuming. Based on the fact that the normal vectors of most planes in building point clouds are axis-aligned, the raw data is pre-processed in the following sequences (Fig.1).

1) SUBSAMPLING

Subsampling is performed to reduce the point number of the original point cloud set $\mathbf{S_{original}}$ and maintain the shape characteristics of the point cloud by employing the spatial subsample method, which is very efficient in improving the speed of registration, surface reconstruction, shape recognition and other functions. We set a distance $\boldsymbol{\varepsilon_1}$ between two points empirically, then pick points from the original cloud $\mathbf{S_{original}}$ to ensure that the distance between any two points is longer than $\boldsymbol{\varepsilon_1}$, and the output point cloud is denoted as $\mathbf{S_{subsample}}$. As shown in Fig.2, we subsample the original point cloud data and the number of points is reduced from 2,354,332 to 37,528. Although the data volume is greatly reduced, the facade part projected onto the X-Y plane still exhibits a dense linear distribution and the average point spacing is 0.015m.



(a) (b)

FIGURE 2. An example of projection results. (a) The projection result with subsampling. (b) The projection result without subsampling.

2) FILTERING

Next, we calculate the distribution of z coordinates in $\mathbf{S_{subsample}}$ and only keep the middle part (denoted as $\mathbf{S_{filtering}}$) to filter out unrelated features and improve line segment extraction efficiency.

3) PROJECTION

We project all points in $\mathbf{S_{filtering}}$ onto the plane, the equation of which is given by:

$$ax + by + cz + d = 0 \tag{1}$$

where $a = b = d = 0$, and $c = 1$. That is, $\mathbf{S_{filtering}}$ is projected onto the X-Y plane, and the output point cloud is denoted as $\mathbf{S_{projection}}$. Note that $\mathbf{S_{projection}}$ only contains x and y coordinates. In the stage of horizontal extraction, we specify $a = c = d = 0$ and $b = 1$ to project point cloud onto the X-Z plane.

B. LINE SEGMENT EXTRACTION

1) COLLINEAR POINT EXTRACTION

After the original point cloud set $\mathbf{S_{original}}$ is pre-processed, the line segments in $\mathbf{S_{projection}}$ need to be further extracted. An extended 2D Hough transform algorithm is employed to address this problem.

The currently universally used version of the HT is proposed by Duda and Hart [17], which replaces the slope-intercept with an angle-radius parameterization based on the normal equation of the line (Fig.3):

$$\rho = xcos(\theta) + ysin(\theta) \tag{2}$$

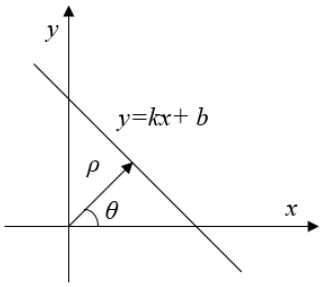


FIGURE 3. The normal parameters for a line. x and y are the coordinates of a point in $S_{projection}$. ρ is the distance from a line to the origin of coordinate system and θ is the angle between the normal of the line and the x-axis.

Given a 2D point cloud set $\mathbf{S_{projection}}$, we calculate the lower left corner coordinate $(\mathbf{x_l}, \mathbf{y_l})$ of $\mathbf{S_{projection}}$, which is then shifted as a whole, so that the lower left corner coincides with the origin and the 2D point cloud set after translation is denoted as $\mathbf{S_{shift}}$. Since the accumulator matrix is in units of discretized integers, it will inevitably cause loss of precision when the polar radius is calculated. To address this problem, the point coordinates of $\mathbf{S_{shift}}$ are enlarged by $\boldsymbol{\varepsilon_2}$ ($\boldsymbol{\varepsilon_2}$ represents an integer multiple of 10, usually set to 1000) to obtain a 2D point cloud set $\mathbf{S_{enlarge}}$. Note that the distance-related thresholds involved in the calculation should also be enlarged by the corresponding multiples. Then the top right corner coordinate $(\mathbf{x_r}, \mathbf{y_r})$ of $\mathbf{S_{enlarge}}$ is calculated so that the maximum polar radius $\boldsymbol{\rho_{max}}$ can be obtained as follows:

$$\rho_{max} = \sqrt{x_r^2 + y_r^2} \tag{3}$$

For practical applications, we need to discretize the parameter $\boldsymbol{\rho}$ and $\boldsymbol{\theta}$ into many small cells. A data structure called accumulator is employed to store all these cells. We set the accumulator size to $\boldsymbol{\varepsilon_3} \times \boldsymbol{\rho_{max}}$. For each point $\mathbf{p_i}$ in $\mathbf{S_{enlarge}}$, the polar radius is calculated as follows:

$$\rho_{ij} = x_i cos\theta_j + y_i sin\theta_j \tag{4}$$

where $0 \leq j \leq \boldsymbol{\varepsilon_3}$, $\theta_j = j*180/\boldsymbol{\varepsilon_3}$ and $(\mathbf{x_i}, \mathbf{y_i})$ is the coordinate of $\mathbf{p_i}$.

In the following, each point is sued to cast a vote at the corresponding accumulator cell. If the calculation result falls into a cell of the parameter space, the accumulator cell will be incremented by+1.

Assuming that the angle and radius corresponding to the cell with the largest cumulative value are $\boldsymbol{\theta_k}$ and $\boldsymbol{\rho_k}$, respectively. For each point $\mathbf{p_i}$ in $\mathbf{S_{enlarge}}$, the polar radius $\boldsymbol{\rho_{ik}}$ corresponding to angle $\boldsymbol{\theta_k}$ is calculated as follows:

$$\rho_{ik} = x_i cos\theta_k + y_i sin\theta_k \tag{5}$$

Due to the discretization of the Hough space and the noise in the input data, it is advisable to search not only for one cell with a maximal score but for the maximum sum in a small region of the accumulator. In order to classify those approximate collinear points into one class, we set a threshold $\boldsymbol{\varepsilon_4}$ ($\boldsymbol{\varepsilon_4} > 0$) to perform this task and store collinear point if the following condition is satisfied:

$$-\varepsilon_4 \leq \rho_{ik} - \rho_k \leq \varepsilon_4 \tag{6}$$

2) COLLINEAR POINT SEGMENTATION

The extracted collinear points are denoted as $\mathbf{S_{col}}$. As shown in Fig.4, the collinear points $\mathbf{S_{col}}$ may contain some collinear but discontinuous points, so it is necessary to split those points into separated segments. The point coordinates in $\mathbf{S_{col}}$ are sorted by the angle $\boldsymbol{\theta_k}$ obtained in the Hough transform. If $0° \leq \boldsymbol{\theta_k} \leq 45°$ or $145° \leq \boldsymbol{\theta_k} < 180°$, the x coordinates are sorted ascendantly and the y coordinates change accordingly; if $45° < \boldsymbol{\theta_k} < 145°$, the y coordinates are sorted ascendantly and the x coordinates change accordingly. The sorted result is denoted as $\mathbf{S_{sort}}$.

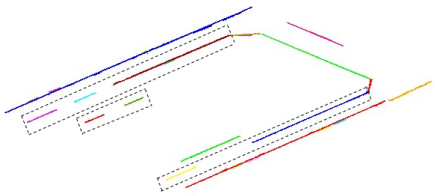


FIGURE 4. An illustration of the extracted line segments. The points in the black rectangles are collinear but not connected.

We calculate the distance $\mathbf{d_i}$ between two adjacent points in $\mathbf{S_{sort}}$, and set a threshold value $\boldsymbol{\varepsilon_5}$ ($\boldsymbol{\varepsilon_5} > 0$) as the farthest distance between two adjacent points. If $\mathbf{d_i} \leq \boldsymbol{\varepsilon_5}$, it indicates that the two adjacent points are on the same segment, otherwise, it means that the last point is not on the same segment with the previous ones, and then we store the previous points in an array, and continue to deal with the remaining points until the collinear points are segmented. The result after collinear point segmentation is denoted as $\mathbf{L_{seg}}$.

For each subset in $\mathbf{L_{seg}}$, we define a threshold value $\boldsymbol{\varepsilon_6}$ ($\boldsymbol{\varepsilon_6} > 0$) as the minimum number of points to form a line segment, if the number of points in the subset exceeds $\boldsymbol{\varepsilon_6}$, it is sorted as a qualifying line segment. The qualifying line segment $\mathbf{L_{subset}}$ are stored in $\mathbf{L_{result}}$. Then points in $\mathbf{L_{subset}}$ are removed from $\mathbf{S_{enlarge}}$ and the next cycle is continued until all the line segments are detected. The point coordinates

of $\mathbf{L_{result}}$ are in the coordinate system of $\mathbf{S_{enlarge}}$. Since $\mathbf{S_{enlarge}}$ is based on a series of transformations on $\mathbf{S_{projection}}$, the original point coordinates stored in $\mathbf{L_{Result}}$ in the coordinate system of $\mathbf{S_{projection}}$ are computed as:

$$\begin{cases} x_{Result} = x_l + x_{reslut}/\varepsilon_1 \\ y_{Result} = y_l + y_{reslut}/\varepsilon_1 \end{cases} \tag{7}$$

For the convenience of distinction, the extracted line segments are marked with different random colors (Fig.4).

3) LINE SEGMENT FITTING

The Hough transform algorithm can suppress noise points in point cloud data well, but its accuracy is difficult to control, which is not suitable for precise applications. The fitting error of least squares method is large when there are too many noise points in the data. In general, noise points are inevitable, so we have to seek ways to get accurate parameters in the presence of noise points. The selecting weight iteration method in robust estimation is widely used, and it is the most commonly used method for eliminating gross errors or suppressing gross errors in data processing.

For a line segment in $\mathbf{L_{Result}}$, the linear equation can be expressed as:

$$\mathrm{y} = a_i * x + b_i \tag{8}$$

where $\mathbf{a}$ is the slope of the line, $\mathbf{b}$ is the intercept of the y-axis, and $\mathbf{a}$ and $\mathbf{b}$ are the parameters to be estimated.

We calculate the approximate values $\mathbf{a_0}$ and $\mathbf{b_0}$ of $\mathbf{a}$ and $\mathbf{b}$ by least squares method as follows:

$$X = (B^T B)^{-1} B^T L \tag{9}$$

where

$$\underset{2,1}{\boldsymbol{X}} = \begin{pmatrix} a_0 \\ b_0 \end{pmatrix}, \quad \underset{n,2}{\mathbf{B}} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}, \quad \underset{n,1}{\mathbf{L}} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

The error equation is computed as:

$$v_{y_i} = \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_b \end{bmatrix} + (a_0 x_i + b_0 - y_i) \tag{10}$$

Equation (10) can be further expressed as:

$$V_k = B\delta X_k - l_k \tag{11}$$

where

$$\underset{n,1}{\mathrm{V}_k} = \begin{bmatrix} v_{y_1} \\ v_{y_2} \\ \vdots \\ v_{y_n} \end{bmatrix} \quad \underset{n,2}{\mathbf{B}} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}$$

$$\underset{2,1}{\delta\mathbf{X}_\mathrm{k}} = \begin{pmatrix} \delta\mathrm{a} \\ \delta\mathrm{b} \end{pmatrix} \quad \underset{n,1}{\mathbf{l}_k} = \begin{pmatrix} y_1 - a_0 x_1 - b_0 \\ y_2 - a_0 x_2 - b_0 \\ \vdots \\ y_\mathrm{n} - a_0 x_n - b_0 \end{pmatrix}$$

According to the least squares criterion:

$$V_k^T P_k V_k = min \tag{12}$$

where

$$\underset{n,n}{\mathbf{P}_k} = \begin{bmatrix} \mathrm{v}_{\mathrm{y}_1}^2 & 0 & \dots & 0 \\ 0 & \mathrm{v}_{\mathrm{y}_2}^2 & & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & \mathrm{v}_{y_n}^2 \end{bmatrix}$$

when k = 1, the parameters in Equation (11) are calculated as follows:

$$\begin{cases} \delta \mathrm{X}_1 = (B^T \mathrm{P}_1 B)^{-1} B^T P_1 \mathrm{l}_1 \\ \mathrm{V}_1 = B\delta \mathrm{X}_1 - l_1 \\ \sigma = \sqrt{\dfrac{\mathrm{V}_1^T P_1 V_1}{n-2}} \end{cases} \tag{13}$$

where

$$\underset{n,n}{\mathbf{P}_k} = \begin{bmatrix} \mathrm{v}_{\mathrm{y}_1}^2 & 0 & \dots & 0 \\ 0 & \mathrm{v}_{\mathrm{y}_2}^2 & & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & \mathrm{v}_{y_n}^2 \end{bmatrix} \quad \underset{2,1}{\delta\mathbf{X}_1} = \begin{pmatrix} \delta\mathrm{a}_1 \\ \delta\mathrm{b}_1 \end{pmatrix}$$

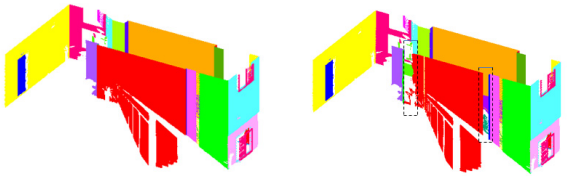
Next, each point is re-weighted, and the Danish weight function is adopted to reduce the influence of noise points on the adjustment result. The Danish weight function has many forms. Here, we adopt the following one:

$$\begin{cases} v_{y_i}^2 = 1 & |\mathrm{v}_{\mathrm{y}_i}| < 1.5\sigma \\ v_{y_i}^2 = v_{y_i}^2 e^{1-\left(\frac{v_{y_i}}{1.5\sigma}\right)^2} & |\mathrm{v}_{\mathrm{y}_i}| \geq 1.5\sigma \\ v_{y_i}^2 = 0 & |\mathrm{v}_{\mathrm{y}_i}| > 5\sigma \end{cases} \tag{14}$$

Then we get the weight matrix $\mathbf{P_2}$ after the re- weighting and substitute $\mathbf{P_2}$ into Equation (12) to calculate the value of $\boldsymbol{\delta X_2}$, $\mathbf{V_2}$ and $\boldsymbol{\sigma}$. In practice, in order to save computation time and improve efficiency, we often set a certain number $\boldsymbol{\varepsilon_7}$ of iterations empirically. The final fitting parameters are calculated as follows:

$$\begin{cases} a = a_0 + \delta a_k \\ b = b_0 + \delta b_k \end{cases} \tag{15}$$

For each line segment in $\mathbf{L_{Result}}$, we get its exact endpoint coordinates ($\mathbf{x_{start}}$, $\mathbf{y_{start}}$) and ($\mathbf{x_{end}}$, $\mathbf{y_{end}}$). Accurate endpoint coordinates are important in the planar segmentation process because some planes are large enough and susceptible to noise. If the endpoint coordinates are not accurate, some of the planar points may be excluded from the cube in the next operation. As shown in Fig.5, the result on the left is based on the coordinates extracted accurately by the selecting weight iteration method, while the result on the right is based on the coordinates directly extracted by the Hough transform. We can easily observe that some of the points are missing (in the black rectangle) because there are pots and other debris near the façade inside the room. Affected by these noise points, the line parameters extracted by the Hough transform are inaccurate.



(a) (b)

FIGURE 5. The effects of accurate and inaccurate endpoint coordinates on the plane segmentation results. (a) Coordinates extracted by the selecting weight iteration method. (b) Coordinates extracted by the Hough transform.

III. VERTICAL PLANE SEGMENTATION

Considering that the normal vector of a vertical plane in $\mathbf{S_{original}}$ is parallel to X-Y plane, the normal vector $\vec{\mathbf{n}}$ is computed as:

$$\vec{n} = \left(1, \frac{x_{end} - x_{start}}{y_{end} - y_{start}}, 0\right) \tag{16}$$

Then the spatial geometric equation of the vertical plane is defined as:

$$A * x + B * y + D = 0 \tag{17}$$

The distance from a space point ($\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$) to its corresponding vertical plane is computed as:

$$d = \frac{|A * x + B * y + D|}{\sqrt{A^2 + B^2}} \tag{18}$$

where

$$\begin{cases} A = 1 \\ B = -\dfrac{x_{end} - x_{start}}{y_{end} - y_{start}} \\ D = \dfrac{x_{end} * y_{start} - x_{start} * y_{end}}{y_{end} - y_{start}} \end{cases}$$

Considering that the size of point cloud data is generally very large, if all point cloud data are involved in the operation to extract a plane, it will be similar to the RANSAC algorithm, which is undoubtedly a time-consuming process. To address this problem, we manage to limit the points involved in the operation to a local range via the following steps.

Firstly, we construct the octree of $\mathbf{S_{original}}$ to organize the point data and define groups of neighboring points for feature estimation. In this octree form, all nodes are split into eight identical children, resulting in that all voxels at a single level with the same size. Additionally, points are stored in every non-leaf node, thereby explicitly linking the node to its children to facilitate tree transversal.

Then for a vertical plane in $\mathbf{S_{original}}$, we design a cube to wrap it and calculate the distribution of the point cloud along the Z-axis direction in $\mathbf{S_{original}}$. Next, we calculate the left front lower corner coordinate ($\mathbf{X_{min}}$, $\mathbf{Y_{min}}$, $\mathbf{Z_{min}}$) and right rear upper corner coordinate ($\mathbf{X_{max}}$, $\mathbf{Y_{max}}$, $\mathbf{Z_{max}}$) of $\mathbf{S_{original}}$. if $0° \leq \boldsymbol{\theta_k} \leq 90°$, the left front lower corner coordinate is ($\mathbf{x_{start}}$, $\mathbf{y_{start}}$, $\mathbf{Z_{min}}+\boldsymbol{\varepsilon_4}$) and right rear upper corner coordinate is($\mathbf{x_{end}}$, $\mathbf{y_{end}}$, $\mathbf{Z_{max}}-\boldsymbol{\varepsilon_4}$). If $90° \leq \boldsymbol{\theta_k} \leq 180°$, the left front lower corner coordinate is ($\mathbf{x_{start}}$, $\mathbf{y_{end}}$, $\mathbf{Z_{min}}+\boldsymbol{\varepsilon_4}$) and right rear upper corner coordinate is ($\mathbf{x_{end}}$, $\mathbf{y_{start}}$, $\mathbf{Z_{max}}-\boldsymbol{\varepsilon_4}$). We extract the points $\mathbf{S_{Cube}}$ inside the cube. The purpose of designing this cube is to confine the points participating in the segmentation process to the vicinity of the vertical plane.

For each point $\mathbf{p_i}$ in $\mathbf{S_{Cube}}$, we calculate its orthophoto distance $\mathbf{d_i}$ to its target façade according to Equation (18) and set a threshold value $\boldsymbol{\varepsilon_8}$($\boldsymbol{\varepsilon_8} > 0$) as the farthest distance from a point to the facade. If $\mathbf{d_i} \leq \boldsymbol{\varepsilon_8}$, the point is kept, otherwise it will be abandoned. All extracted vertical planes are stored in $\mathbf{S_{vertical}}$. As shown in Fig.6, the extracted vertical planes are marked with different random colors.

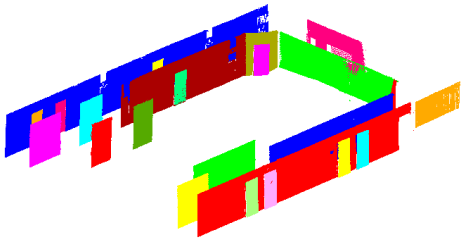


FIGURE 6. A demonstration of the extracted vertical planes.

Since we have established the octree structure of the original point cloud data set $\mathbf{S_{original}}$ in advance, each point in $\mathbf{S_{vertical}}$ has an explicit index and it is convenient to remove $\mathbf{S_{vertical}}$ from $\mathbf{S_{original}}$. The remaining point cloud is stored in $\mathbf{S_{remain}}$. As shown in Fig.7, most vertical planes in the original point cloud are extracted and the remaining part is mainly composed of horizontal planes and other debris points.

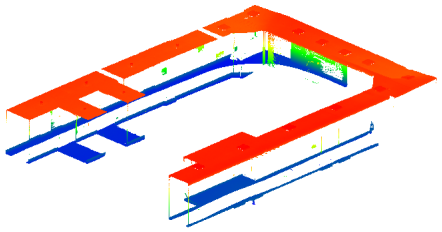


FIGURE 7. A demonstration of the remaining point cloud.

Then we use $\mathbf{S_{remain}}$ as the original input point cloud and perform a process similar to the above to extract the

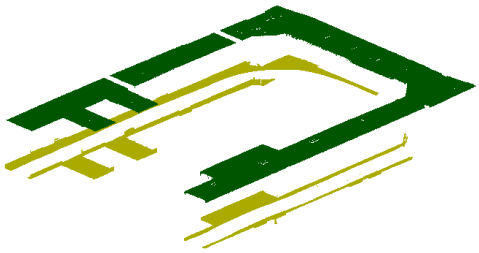


FIGURE 8. A demonstration of the extracted horizontal planes.

horizontal plane (Fig.8). Note that (1) the filtering operation in the facade segmentation process is not used here; (2) point cloud is projected onto the X-Z plane rather than the X-Y plane, the parameters in Equation (1) are a = c = d = 0 and b = 1; (3) only line segments with the angle $\boldsymbol{\theta_k}$ obtained in the Hough transform between 0° and 5° are considered as candidates; (4) the spatial geometric equation of the horizontal plane is also slightly different from the vertical planes. Finally, we merge the vertical and horizontal planes to get the segmentation result (Fig.9).

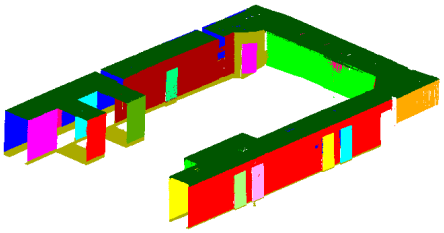


FIGURE 9. A demonstration of the point cloud segmentation result.

IV. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed building planar feature segmentation method, we tested it on four different scenes. As shown in Fig.10, scene 1 is an interior stair corner point cloud and scene 2 is a point cloud of an outdoor plaza consisting of a number of modern buildings and some trees at Wuhan University, both of them were scanned using a Faro Focus ^{S}150 laser scanner. For the convenience of displaying the interior of scene 1, the ceiling part is removed. Scene 3 is a point cloud captured of the hallways of the 5th floor of Cory Hall on the UC Berkeley campus using a pushcart system consisting of 3 laser scanners, 2 cameras, and 1 IMU. Scene 4 is a point cloud from the public point cloud dataset: Semantic3D. All algorithms in this paper were implemented using C++ and point cloud library 1.8.0, which were performed on a PC with Intel Core i5-830H 2.3GHz CPU and 8GB RAM.

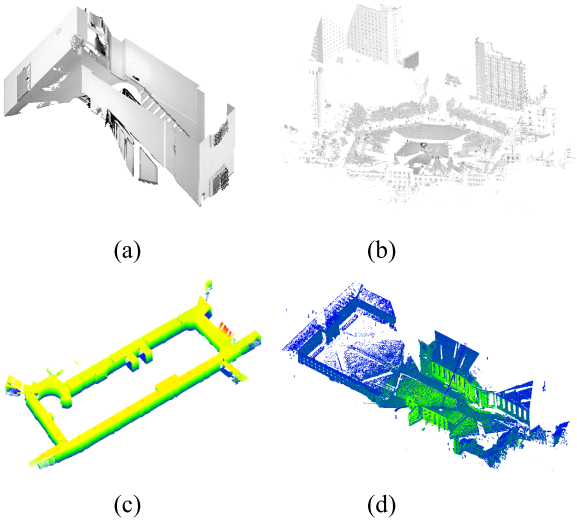


FIGURE 10. Input point clouds. (a) An interior stair corner point cloud. (b) A point cloud of an outdoor plaza consisting of a number of modern buildings and some trees. (c) A point cloud captured of the hallways of the 5th floor of Cory Hall on the UC Berkeley campus. (d) A point cloud from the public point cloud dataset: Semantic3D.

A. PARAMETERS SETTING

As we can see from the above sections, there are generally two categories of parameters in our algorithm: number-related and distance-related. Below, we describe how to set the parameters.

The subsampling parameter $\boldsymbol{\varepsilon_1}$ denotes the closest distance between any two points. Generally, $\boldsymbol{\varepsilon_1}$ cannot be determined by a fixed value; rather, it depends on the scale of the data and the noise level. For indoor dense point cloud like scene 1, $\boldsymbol{\varepsilon_1}$ is set to 0.05 m, while for large scale point clouds like scene 2, scene 3, and scene 4 where points are relatively sparse, $\boldsymbol{\varepsilon_1}$ is set to 0.1 m. $\boldsymbol{\varepsilon_2}$ denotes the multiple of the coordinate magnification and usually set to 1000, which is suitable for most applications. $\boldsymbol{\varepsilon_3}$ is the number that divides 180° equally and usually set to 500. $\boldsymbol{\varepsilon_4}$ denotes the farthest orthophoto distance to determine whether a point lies on a line. Here, we estimate $\boldsymbol{\varepsilon_4}$ through statistical analysis, i.e., set $\boldsymbol{\varepsilon_4}$ to 5 times the average spacing of $\mathbf{S_{projection}}$. This value works well on a broad spectrum of datasets. As to the remaining parameters, $\boldsymbol{\varepsilon_5}$ is the farthest distance that determines whether two adjacent points lie on the same line segment, $\boldsymbol{\varepsilon_6}$ is the minimum number of points to form a line segment, $\boldsymbol{\varepsilon_7}$ is the number of iterations in line segment fitting, and $\boldsymbol{\varepsilon_8}$ is the farthest distance from a point to its corresponding facade. We used $\boldsymbol{\varepsilon_5} = 15\,\boldsymbol{\varepsilon_4}$, $\boldsymbol{\varepsilon_6} = 20$, $\boldsymbol{\varepsilon_7} = 10$, and $\boldsymbol{\varepsilon_8} = \boldsymbol{\varepsilon_4}$ in experiments. Note that all the above parameters are used for vertical plane segmentation. As to the horizontal plane segmentation, the value of $\boldsymbol{\varepsilon_4}$ depends on the situation, if $\mathbf{S_{remain}}$ consists mainly of small range of horizontal planes, e.g. for scene 1, $\boldsymbol{\varepsilon_4}$ remains unchanged, otherwise we set $\boldsymbol{\varepsilon_4}$ to 10 times the average spacing of $\mathbf{S_{projection}}$. These parameter values were not tuned throughout all of the following experiments. It is worth mentioning that fine-tuning these parameters for different types of point clouds would obtain better results.

B. PERFORMANCE

Fig.11 shows the details of the 3D plane segmentation results. As shown in Fig.11(a) and Fig.11(c), we can see that not only the large structures, like the walls, floors, and ceilings, but also the small details, like the windows, doors, and stairs, are well segmented. Those coplanar but discontinuous planes are also well separated. As shown in Fig.11(b) and Fig.11(d), our algorithm preserves most of the important geometric contents, while filtering out the trees, bush, and other sundries. However, due to the characteristics of the method, those inclined planes, such as roofs, cannot be extracted. Table 1 demonstrates the statistic results about the experiments performed with our technique on each test data, from which we can get the following observations: in general, the proposed method successfully extracts the major planes of the test data within acceptable time; the total

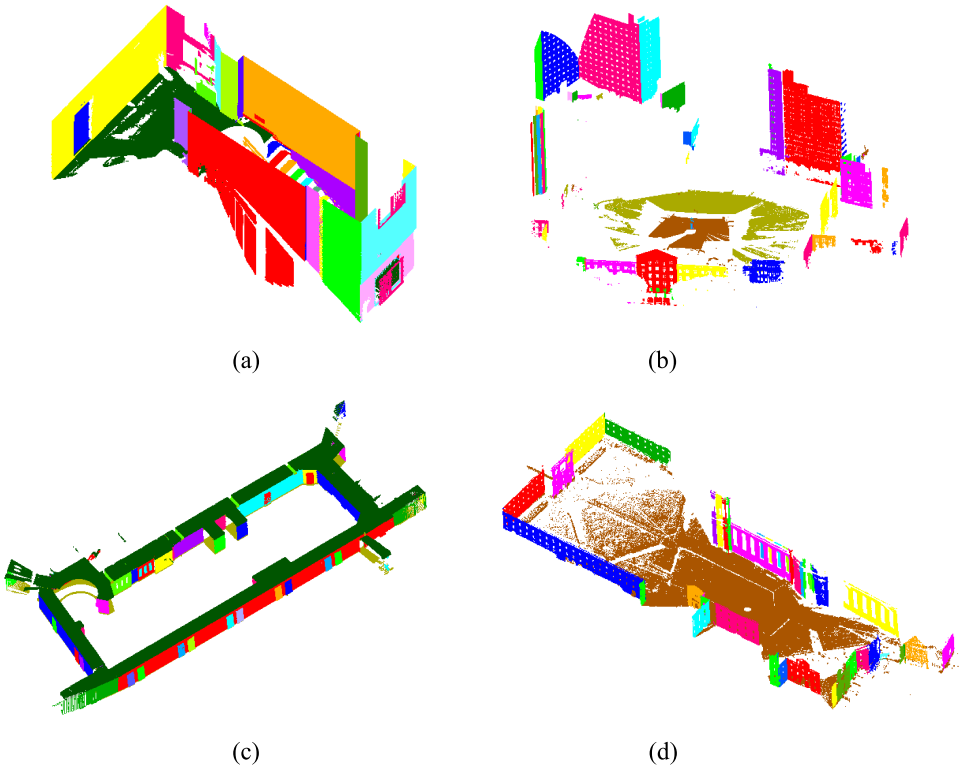


FIGURE 11. Segmentation results of the proposed method. (a) Scene 1 plane segmentation results by the proposed method. (b) Scene 2 plane segmentation results by the proposed method. (c) Scene 3 plane segmentation results by the proposed method. (d) Scene 4 plane segmentation results by the proposed method.

TABLE 1. Computing results in four different point clouds. Size, Spacing, Bounding Box, N_p , T_v, T_h, T_{all} denote the point number, the average distance between points, the length, width and height of a point cloud, total number of planes, the time for segmentation of vertical planes, the time for segmentation of horizontal planes, the total time, respectively.

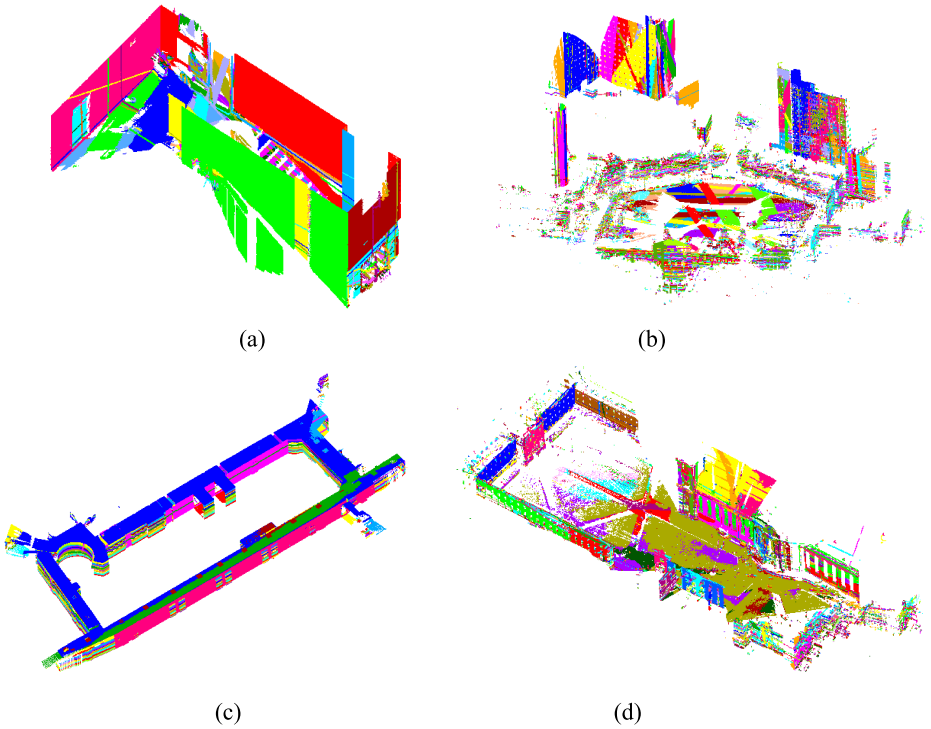
Name	Size	Spacing(m)	Bounding Box (m)	N_p	T_v(s)	T_h (s)	T_{all} (s)
Scene 1	2,354,332	0.005	8.3×9.9×5.7	46	215.63	86.82	302.45
Scene 2	4,391,833	0.05	264.4×285.9×77.7	73	301.52	162.61	464.12
Scene 3	4,218,838	0.01	54.9×29.2×4.6	89	291.94	134.39	426.33
Scene 4	7,384,510	0.02	320.3×156.4×73.6	68	362.15	270.56	632.71

segmentation time is linearly proportional to the number of input points; the size of bounding box has less impact on total segmentation time. Note that these time periods were generated by averaging the time period of 10 executions, including the time for reading the original point cloud data and for saving the segmentation result.

C. COMPARISON WITH OTHER METHODS

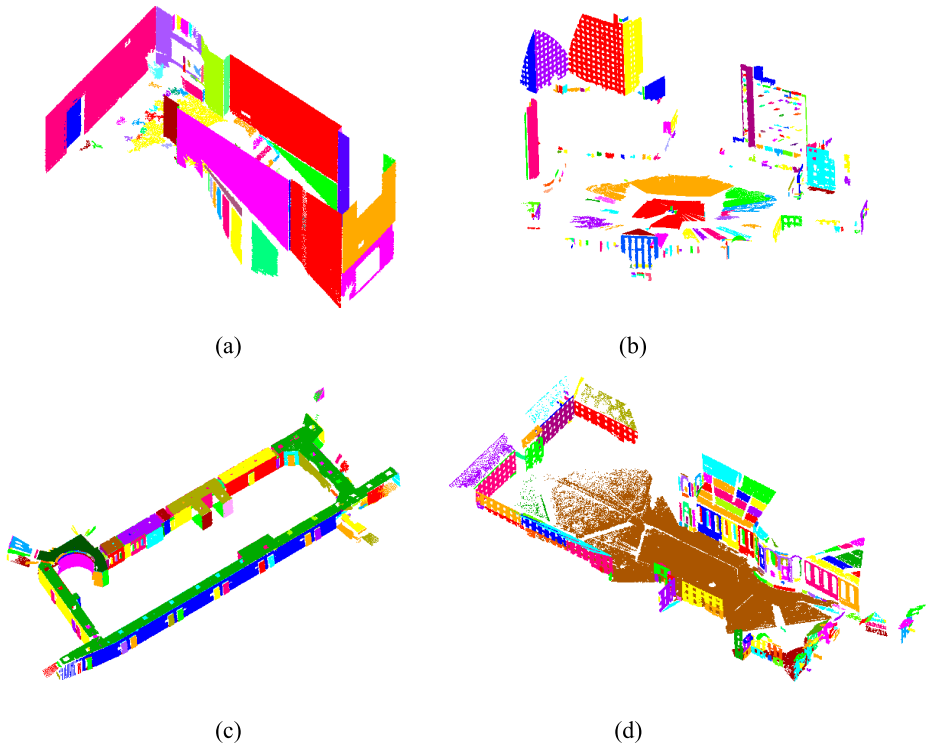
Further evaluation is performed by comparing our method with the standard RANSAC, and the Region Growing for plane detection that are implemented in the PCL library [41], a modern C++ library for 3D point cloud processing. The PCL's RANSAC method is highly optimized and it is further extended by successive least squares refining applied after random sample consensus fitting. Exact definition of the implemented RANSAC algorithm is described in [42]. The inlier threshold distance is set to ε_4 that is applied in vertical plane segmentation. The maximum number of iterations for the detection of one plane is set to 50 for scene 1 and 100 for scene 2, scene 3, and scene 4. The parameters in Region Growing algorithm are also optimally set for comparison with other methods.

Fig.12 shows the details of the RANSAC segmentation results and Fig.13 shows the segmentation results of the region growing algorithm. As shown in Fig.12, the PCL's RANSAC algorithm is prone to over-segmentation and under-segmentation. Taking scene 1 as an example, the floor is divided into multiple planes and continuous stairs are mistakenly identified as one plane. Besides, the algorithm cannot



(a) (b) (c) (d)

FIGURE 12. Segmentation results of PCL's RANSAC. (a) Scene 1 plane segmentation results by PCL's RANSAC. (b) Scene 2 plane segmentation results by PCL's RANSAC. (c) Scene 3 plane segmentation results by PCL's RANSAC. (d) Scene 4 plane segmentation results by PCL's RANSAC.



(a) (b) (c) (d)

FIGURE 13. Segmentation results of PCL's Region Growing. (a) Scene 1 plane segmentation results by PCL's Region Growing. (b) Scene 2 plane segmentation results by PCL's Region Growing. (c) Scene 3 plane segmentation results by PCL's Region Growing. (d) Scene 4 plane segmentation results by PCL's Region Growing.

effectively distinguish among coplanar but discontinuous planes. While being robust to noise, the random nature of RANSAC makes it non-deterministic. Depending on the choice of its parameter values, the algorithm may detect planes that are not representative of the original data, which is illustrated in scene 2, scene 3, and scene 4. As shown in Fig.13, over-segmentation and under-segmentation still inevitably exist in the Region Growing algorithm, but the

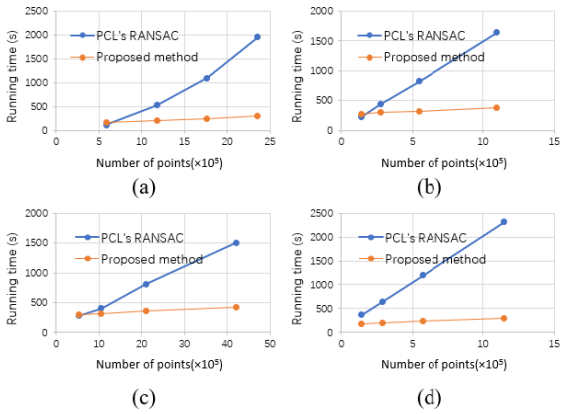


FIGURE 14. Comparative experiments of the proposed method and the PCL's RANSAC. (a) Comparative experiments for Scene 1. (b) Comparative experiments for Scene 2. (c) Comparative experiments for Scene 3. (d) Comparative experiments for Scene 4.

algorithm effectively avoids the problem of misidentification in RANSAC. Note that since Region Growing algorithm is not as fast as RANSAC and our method, it requires significant processing time to process the raw data. In order to obtain the segmentation results within acceptable time, we have to sample the original point cloud data before running the Region Growing algorithm.

To investigate the influence of data size (i.e. number of points) on the running time, the four original point clouds were sub-sampled at four other density levels. Each point cloud was sequentially segmented using RANSAC and the proposed method. The running time of the two methods are shown in Fig.14. We can clearly see that the running time of the PCL's RANSAC increases drastically with the increase of the size of data. By contrast, the calculation time of our method grows slowly.

To show the robustness of the proposed method, planes were extracted from point clouds with noise. We study the robustness of the proposed method by adding Gaussian noise to a synthetic point cloud. As shown in Fig.15, the original point cloud is uniformly sampled from the surface of a 1m × 1m × 1m cube. We can observe that as Gaussian noise increases, the planes can still be accurately extracted.

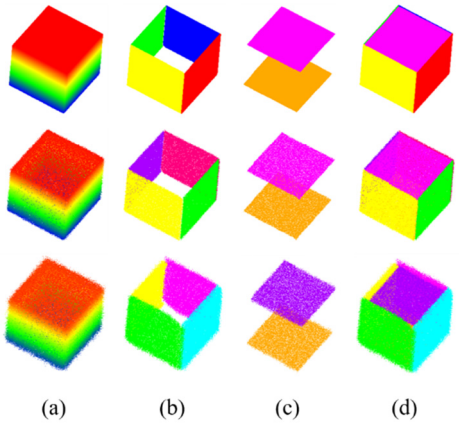


FIGURE 15. Extracted planes in the presence of different levels of Gaussian noise. From top to bottom: 0 m Gaussian noise, 0.02 m Gaussian noise and 0.05 m Gaussian noise. (a) Input point cloud. (b) Extracted vertical planes. (c) Extracted horizontal planes. (d) the point cloud segmentation results.

D. FAILURE CASES

As we have introduced above, our method is projection-based, which is suitable for the segmentation of vertical and horizontal planes but may also lead to failure in the case of inclined planes. Despite these failure cases, our method is very efficient and effective in the man-made scene with lots of structural planes.

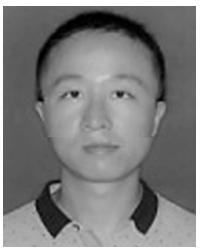
V. CONLUSION

In this paper we present and testify a simple but efficient algorithm for segmentation of building planar features from unorganized point cloud. We can find applications for this technology in building construction, object recognition, and other fields. The proposed method is based on 2D Hough transform and octree. Compared with the traditional classic method, (1) it does not require connectivity information nor information about normal vector; (2) it can effectively segment coplanar planes that are not connected or parallel but non-coplanar planes; (3) it scales well with the size of the datasets and is robust to the presence of noise, and we do not use denoising operations throughout the algorithm; (4) unique design makes it possible to avoid the occurrence of pseudo-planar phenomenon. The proposed method has been tested on various point cloud data, including the point clouds obtained by laser devices, and the synthetic point clouds with noise. We also compared the proposed method with two other methods and provided visual comparison results. The experimental results show that our algorithm is efficient, insensitive to noise and outliers, and produces more accurate and complete planes than the compared methods. Since our method is projection-based, one failure case of our method is on the inclined planes. As for those non-horizontal and non-vertical walls, first, we remove the extracted horizontal and vertical planes from the original point cloud. Next, we use traditional method such as Region Growing for plane extraction. Since the remaining valuable planes are mainly outdoor sloping roofs, in order to save calculation time, only those points with a height within a certain range are considered as candidate points to participate in the segmentation calculation.

REFERENCES

[1] 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.
[2] H. Huang, C. Brenner, and M. Sester, "3D building roof reconstruction from point clouds via generative models," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. (GIS)*, Chicago, IL, USA, 2011, pp. 16–24.
[3] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, St. Louis, MO, USA, Oct. 2009, pp. 1–6.

[4] H.-W. Lin, C.-L. Tai, and G.-J. Wang, "A mesh reconstruction algorithm driven by an intrinsic property of a point cloud," *Comput.-Aided Des.*, vol. 36, no. 1, pp. 1–9, Jan. 2004.
[5] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. Mcgann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. Gerkey, and E. Berger, "Autonomous door opening and plugging in with a personal robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, May 2010, pp. 729–736.
[6] R. B. Rusu, W. Meeussen, S. Chitta, and M. Beetz, "Laser-based perception for door and handle identification," in *Proc. Int. Conf. Adv. Robot.*, Munich, Germany, 2009, pp. 1–8.
[7] D. Chekhlov, A. P. Gee, A. Calway, and W. Mayol-Cuevas, "Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual SLAM," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nara, Japan, Nov. 2007, pp. 153–156.
[8] R. Zlot, M. Bosse, K. Greenop, Z. Jarzab, E. Juckes, and J. Roberts, "Efficiently capturing large, complex cultural heritage sites with a handheld mobile 3D laser mapping system," *J. Cultural Heritage*, vol. 15, no. 6, pp. 670–678, Nov. 2014.
[9] N. Okamoto, K. Hasegawa, L. Li, A. Okamoto, and S. Tanaka, "Highlighting feature regions combined with see-through visualization of laser-Scanned cultural heritage," in *Proc. Int. Conf. Culture Comput. (Culture Comput.)*, Kyoto, Japan, 2017, pp. 7–12.
[10] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognit.*, vol. 13, no. 2, pp. 111–122, Jan. 1981.
[11] M. H. Nasseri, H. Moradi, S. M. Nasiri, and R. Hosseini, "Power line detection and tracking using Hough transform and particle filter," in *Proc. 6th RSI Int. Conf. Robot. Mechatronics (IcRoM)*, Tehran, Iran, 2018, pp. 130–134.
[12] L. A. F. Fernandes and M. M. Oliveira, "Real-time line detection through an improved Hough transform voting scheme," *Pattern Recognit.*, vol. 41, no. 1, pp. 299–314, Jan. 2008.
[13] M. Camurri, R. Vezzani, and R. Cucchiara, "3D Hough transform for sphere recognition on point clouds," *Mach. Vis. Appl.*, vol. 25, no. 7, pp. 1877–1891, Oct. 2014.
[14] A. Abuzaina, M. S. Nixon, and J. N. Carter, "Sphere detection in Kinect point clouds via the 3D Hough transform," in *Proc. 15th Int. Conf. Comput. Anal. Images Patterns*, Berlin, Germany, 2013, pp. 290–297.
[15] R. Hulik, M. Spanel, P. Smrz, and Z. Materna, "Continuous plane detection in point-cloud data based on 3D Hough transform," *J. Vis. Commun. Image Represent.*, vol. 25, no. 1, pp. 86–97, Jan. 2014.
[16] F. A. Limberger and M. M. Oliveira, "Real-time detection of planar regions in unorganized point clouds," *Pattern Recognit.*, vol. 48, no. 6, pp. 2043–2053, Jun. 2015.
[17] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.
[18] N. Kiryati, Y. Eldar, and A. M. Bruckstein, "A probabilistic Hough transform," *Pattern Recognit.*, vol. 24, no. 4, pp. 303–316, Jan. 1991.
[19] A. Yla-Jaaski and N. Kiryati, "Adaptive termination of voting in the probabilistic circular Hough transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 9, pp. 911–915, Sep. 1994.
[20] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic Hough transform," *Comput. Vis. Image Understand.*, vol. 78, no. 1, pp. 119–137, Apr. 2000.
[21] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: Randomized Hough transform," *Pattern Recognit.*, vol. 11, no. 5, pp. 331–338, 1990.
[22] O. O. Ogundana, C. R. Coggrave, R. L. Burguete, and J. M. Huntley, "Automated detection of planes in 3D point clouds using fast Hough transforms," *Opt. Eng.*, vol. 50, no. 5, pp. 053609-1–053609-11, 1981.
[23] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM.*, vol. 24, pp. 381–395, Jan. 2011.
[24] D. P. Chen, X. N. Chu, X. W. Sun, and Y. P. Li, "A new product service system concept evaluation approach based on information axiom in a fuzzy-stochastic environment," *Int. J. Comput. Integr. Manuf.*, vol. 28, no. 11, pp. 1123–1141, 2015.
[25] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," *Comput. Graph. Forum*, vol. 26, no. 2, pp. 214–226, 2007.
[26] D. Chen, L. Zhang, P. T. Mathiopoulos, and X. Huang, "A methodology for automated segmentation and reconstruction of urban 3-D buildings from ALS point clouds," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 10, pp. 4199–4217, Oct. 2014.
[27] T. M. Awwad, Q. Zhu, Z. Q. Du, and Y. T. Zhang, "An improved segmentation approach for planar surfaces from unstructured 3D point clouds," *Photogramm. Rec.*, vol. 25, no. 129, pp. 5–23, 2010.
[28] D. S. Chen, "A data-driven intermediate level feature extraction algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 749–758, Jul. 1989.
[29] P. J. Besl and R. C. Jain, "Segmentation through variable-order surface fitting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-10, no. 2, pp. 167–192, Mar. 1988.
[30] A. Nurunnabi, D. Belton, and G. West, "Robust segmentation in laser scanning 3D point cloud data," in *Proc. Int. Conf. Digit. Image Comput. Techn. Appl. (DICTA)*, Fremantle, WA, Australia, 2012, pp. 1–8.
[31] M. Vieira and K. Shimada, "Surface mesh segmentation and smooth surface extraction through region growing," *Comput.-Aided Geometric Des.*, vol. 22, no. 8, pp. 771–792, 2005.
[32] A. V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto, "Octree-based region growing for point cloud segmentation," *ISPRS J. Photogramm. Remote Sens.*, vol. 104, pp. 88–100, Jun. 2015.
[33] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[34] A. Boulch, B. L. Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks," in *Proc. Eurograph. Workshop 3D Object Retr.*, vol. 2, 2017, pp. 17–24.
[35] J. Guerry, A. Boulch, B. L. Saux, J. Moras, A. Plyer, and D. Filliat, "SnapNet-R: Consistent 3D multi-view semantic labeling for robotics," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Venice, Italy, Jun. 2017, pp. 669–678.
[36] F. Lateef and Y. Ruichek, "Survey on semantic segmentation using deep learning techniques," *Neurocomputing*, vol. 338, pp. 321–348, Apr. 2019.
[37] B. Graham, "Spatially-sparse convolutional neural networks," *Comput. Sci.*, vol. 34, no. 6, pp. 864–867, 2014.
[38] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 77–85.
[39] X. Lu, J. Yao, K. Li, and L. Li, "CannyLines: A parameter-free line segment detector," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Quebec City, QC, Canada, Sep. 2015, pp. 507–511.
[40] R. G. V. Gioi, J. Jakubowicz, J. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *Trans. Softw. Eng.*, vol. 32, no. 4, pp. 722–732, 2010.
[41] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 1–4.
[42] O. Chum and J. Matas, "Matching with PROSAC-progressive sample consensus," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Washington, DC, USA, Jul. 2005, pp. 220–226.



PENGJU TIAN (Member, IEEE) received the B.S. and M.S. degrees from Wuhan University, Wuhan, China, in 2014 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the School of Geodesy and Geomatics. His research interests include laser scanning and data processing.

XIANGHONG HUA received the B.S., M.S., and Ph.D. degrees from Wuhan University, Wuhan, China, in 1984, 1996, and 2006, respectively. He is currently a Professor with the School of Geodesy and Geomatics, Wuhan University. His research interests include laser scanning and indoor position.

KEGEN YU (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from The University of Sydney, Sydney, NSW, Australia, in 2003. He was with Jiangxi Geological and Mineral Bureau, Nanchang, China, Nanchang University, Nanchang, the University of Oulu, Oulu, Finland, the CSIRO ICT Center, Sydney, Macquarie University, Sydney, the University of New South Wales, Sydney, and Wuhan University, Wuhan, China. Since 2011, he has been an Adjunct Professor with Macquarie University. He is currently a Professor with the School of Environment Science and Spatial informatics, China University of Mining and Technology, Xuzhou, China. He has coauthored the book *Ground-Based Wireless Positioning* (Wiley and IEEE Press, 2009), a Chinese version of the book is also available, and the book *Wireless Positioning: Principles and Practice* (Springer, 2018). He edited a book *Positioning and Navigation in Complex Environments* (IGI Global, 2018). He has authored or coauthored over 100 refereed journal and conference papers. His research interests include global navigation satellite systems reflectometry, ground-based and satellite-based positioning, and remote sensing.

Dr. Yu has recently served five years on the editorial boards of *EURASIP Journal on Advances in Signal Processing*, the IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS, and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He was the Lead Guest Editor for a special issue of *Physical Communication* on Indoor Navigation and Tracking and for a special issue of *EURASIP Journal on Advances in Signal Processing* on GNSS remote sensing.

WUYONG TAO received the master's degree from the East China Institute of Technology, Nanchang, China, in 2015. He is currently pursuing the Ph.D. degree with the School of Geodesy and Geomatics, Wuhan University, Wuhan, China. His research interests include laser scanning and data processing.