

## Article

# Automatic Extraction of Indoor Structural Information from Point Clouds

Dongyang Cheng, Junchao Zhang , Dangjun Zhao , Jianlai Chen  and Di Tian

School of Aeronautics and Astronautics, Central South University, Changsha 410083, China; cheng\_dy@csu.edu.cn (D.C.); zhao\_dj@csu.edu.cn (D.Z.); jianlaichen@csu.edu.cn (J.C.); tiandi\_csu@csu.edu.cn (D.T.)

\* Correspondence: junchaozhang@csu.edu.cn

**Abstract:** We propose an innovative method with which to extract building interior structure information automatically, including ceiling, floor, and wall. Our approach outperforms previous methods in the following respects. First, we propose an approach based on principal component analysis (PCA) to find the ground plane, which is regarded as the new Cartesian plane. Second, to reduce the complexity of data processing, the data are projected into two dimensions and transformed into a binary image via the operation of an improved radius outlier removal (ROR) filter. Third, a traditional thinning algorithm is adopted to extract the image skeleton. Then, we propose a method for calculating slope through the nearest neighbor point. Moreover, the line is represented with the slopes to obtain information pertaining to the interior planes. Finally, the outline of the line is restored to a three-dimensional structure. The proposed method is evaluated in multiple scenarios, and the results show that the method is accurate (the maximum error of 0.03 m was in three scenarios) in indoor environments.



**Citation:** Cheng, D.; Zhang, J.; Zhao, D.; Chen, J.; Tian, D. Automatic Extraction of Indoor Structural Information from Point Clouds. *Remote Sens.* **2021**, *13*, 4930. <https://doi.org/10.3390/rs13234930>

Academic Editors: Mi Wang, Hanwen Yu, Jianlai Chen and Ying Zhu

Received: 9 October 2021

Accepted: 2 December 2021

Published: 4 December 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** interior structure; reconstruction; PCA; projection; ROR; thinning; nearest neighbor; slopes

## 1. Introduction

In recent years, LiDAR (light detection and ranging) technology has developed rapidly due to its high accuracy, low cost, portability, and wide application range such as in autonomous driving [1–5], military fields [6–9], aerospace [10,11], and three-dimensional (3D) reconstruction [12–14]. In terms of 3D modeling, high-precision and high-density point cloud data are provided by LiDAR to accurately restore the surface model of an object that could be a trunk [15], a geological landform [16], or a building [17,18]. With the maturation of indoor navigation [19] technology, it is important to obtain precise building interior structures from the point cloud for accurate navigation. However, it is difficult and time consuming to extract indoor structures from the disorder and high density [20] of point clouds, and the complexity of data processing is greatly increased due to the noise caused by the algorithm and the scanning environment. Therefore, completing interior reconstruction is challenging under the influence of these negative factors.

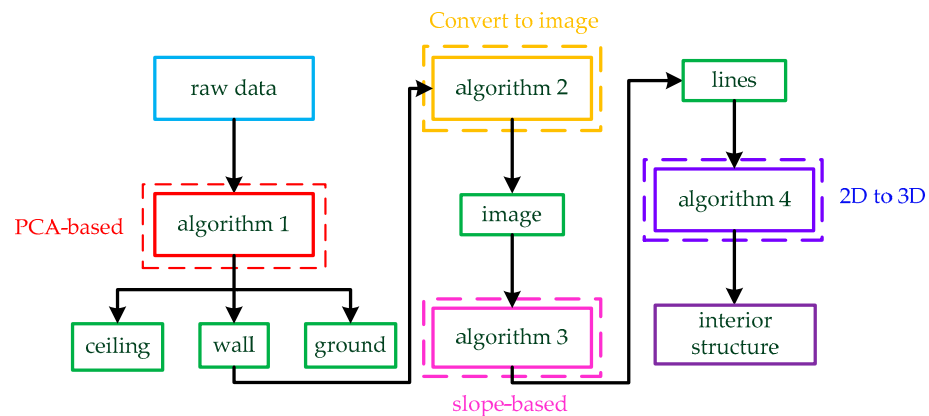
To reconstruct the interior structure accurately, researchers have proposed advanced ideas [21,22] and frameworks. First of all, the raw data of the point cloud mainly come from a laser scanner. Valero et al. [23] used a laser scanner to attain neat point clouds and different algorithms to identify different objects. The point cloud from the laser scanner is precise and the point cloud that makes up the plane is projected vertically as a straight line, which results in great convenience in the subsequent processing. However, it is expensive and takes more time to measure. Therefore, we use LiDAR because the data from cheap and lightweight LiDAR are easily available and convenient. In some large-scale scenes, the data measured by LiDAR are integrated by a scan registration algorithm such as LeGO-LOAM (lightweight and ground-optimized LiDAR odometry and mapping) [24] or

LIO-SAM (lidar inertial odometry via smoothing and mapping) [25], both of which cause point cloud noise.

After obtaining the initial data, indoor structure extraction from a point cloud can be divided into the following two aspects: one is to extract plane information directly from the point cloud, and the other is to extract the indoor plane contour by projecting or transforming it into an image. Ochmann et al. [26] transformed the problem into a linear programming problem of room contour under the prior knowledge that the room was segmented. On this basis, Ochmann et al. [27] also proposed the use of random sample consensus (RANSAC) [28] to automatically segment the room and then transform it into a linear programming problem. The RANSAC is robust and can be modified according to point cloud models of different shapes. Saval-Calvo et al. [29] optimized the RANSAC so that planar point clouds could be effectively extracted in the environment, and the performance was better than the original algorithm. Ambrus et al. [30] used the RANSAC to extract the floor and walls of each room, and then projected the walls onto the floor to extract the outlines of the room. Sanchez et al. [31] proposed a model based on model-fitting and RANSAC, which could effectively extract large-scale buildings and small-scale structures. In addition to this, Wang et al. [32] used the region-growing method to extract all plane frames and extracted the line structure of each plane to automatically generate complete building information models (BIMs). Mura et al. [33] proposed an efficient occlusion-aware process to extract the walls, which were projected onto a two-dimensional plane to extract profiles. Random algorithms often need to traverse every point, which leads to this method being time consuming but robust. Therefore, these algorithms with randomness and growth are effective but inefficient. Some researchers are also looking for other ways to reduce the cost of the algorithms. Oesau et al. [34] divided the initial data into chunks, extracted the outline with a Hough transform [35] in each block, and then integrated them. The time to process the data was greatly reduced by the chunking. Wu et al. [36] sliced the point cloud in different ways and then fitted the line using a RANSAC-based method. The amount of data can be greatly reduced but the loss of indoor data may be caused by the slicing operation. Wang et al. [37] reconstructed the buildings after classifying different objects, which reduced the processing of non-target point clouds. Xiong et al. [38] semantically segmented the interior environment and identified the plane with openings and the shape of openings. Due to the good visual effect and fast processing speed of images, three-dimensional information can also be restored through images [39,40]. Stojanovic et al. [41] used images to aid point cloud data for classifying indoor objects. Jung et al. [42] projected the data onto the ground and turned them into an image, and then detected the corners of the walls to determine their contours. After the data were projected and converted into an image, Jung et al. [43] extracted the two-dimensional skeleton of the wall with the image-thinning algorithm and restored it to a three-dimensional structure. Large point clouds are converted into image skeletons, which greatly reduces computational costs. Moreover, the point cloud changes from irregular points to regular pixels, which is very beneficial for the extraction of structural information. However, the reduction in the amount of data means that some details are lost if nothing is established to optimize the data.

In general, the direct processing of point clouds is time consuming but robust, while dimensionality reduction or image conversion is fast but will reduce accuracy. To inherit the advantages of the above methods and make up for their shortcomings, we present our ideas in this paragraph. In this work, we propose a hypothesis: the walls are perpendicular to the ground and the angles between the walls are only within  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ , and  $135^\circ$ . To reduce time consumption, our data are collected by LiDAR because our target scene is relatively large. In previous studies, it is usually taken as a prior condition that the ground is the x-y plane. Based on a large number of experiments, we find that the prior condition is sometimes untenable. Therefore, we propose a PCA-based framework to find the floor and ceiling plane equations. Then, we project the complete wall onto the ground in the previous step and convert it into images rather than slicing. To remove the sparse noise and

improve the filtering speed, we optimize the radius filtering without changing the effect. Next, coordinates are enlarged and meshed to convert them into pixels. After conversion to an image, the width of the line is several pixels, which is not conducive for us to extract the structure, so we consider extracting the image skeleton using the thinning algorithm. Saeed et al. [44] proposed an image-thinning algorithm that can extract image skeletons effectively; however, this algorithm is not sensitive to corner points. Zhang and Suen [45] proposed a method that could quickly and accurately extract an image skeleton and reflect image corner information by setting several constraint conditions through the relationship between the target pixel and eight-neighborhood pixels. Therefore, the method [43] is more suitable for our data. Due to the error in scan registration, the intersection of two lines produces a radian instead of a right angle. Corner detection in our data is difficult with traditional corner detection operators such as the Harris operator [46] and the SUSAN operator [47]. Chen et al. [48] proposed to use the curvature of each point as a constraint to screen corner points, which could not find corner points very accurately. By observation, we find that the intersection of adjacent lines can be used as the exact value of corner points. As such, we propose a judgment method of a straight line based on slope and select corners from the intersection points of straight lines. Finally, we restore the three-dimensional model according to corner points and corresponding lines. The entire flowchart is shown in Figure 1.



**Figure 1.** The whole process of the method. The raw data are input to the PCA-based Algorithm 1, and the data are divided into three parts: ceiling, wall, and ground. Then, the three-dimensional wall is transformed into an image by Algorithm 2. Next, we use slope-based Algorithm 3 to find lines in the image. Finally, the three-dimensional interior structure is restored from lines by Algorithm 4.

---

#### Algorithm 1 Principal Component Analysis (PCA)

---

**Input:**  $S_I(n \times 3 \text{ points})$

**Output:**  $S_O(n \times 2 \text{ points})$

- 1:  $\mu = \left( \frac{1}{n} \sum_{i=1}^n x_i, \frac{1}{n} \sum_{i=1}^n y_i, \frac{1}{n} \sum_{i=1}^n z_i \right)$
  - 2: **for**  $p_i = (x_i, y_i, z_i) \in S_I, i = 1, 2, \dots, n$  **do**
  - 3:    $p_j = p_i - \mu$
  - 4: **end for**
  - 5:  $p_j = (x_j, y_j, z_j) \in S_P, j = 1, 2, \dots, n$
  - 6:  $\Gamma = \frac{1}{n} S_P^T S_P$
  - 7:  $U \Sigma V^T = \text{SVD}(\Gamma)$
  - 8:  $V = [v_1, v_2, v_3]$
  - 9:  $S_O = S_I[v_1, v_2]$
- 

The rest of this paper is organized as follows. Section 2.1 presents the principle of PCA and the framework that separates the walls based on PCA. In Section 2.2, we introduce

how we can transform data into images, including point cloud projection, improved radius filtering, and data regularization. Section 2.3 presents the principle of the slope-based algorithm and some detailed processing and optimization. In Section 2.4, we organize the expression of straight lines and restore the three-dimensional structure of the interior environment. In Section 3.1, we introduce our equipment and the data extraction process. In Section 3.2, we use three different scenarios to verify our method and give the relevant evaluation parameters of three experimental results. Section 4 shows our discussion of the experimental results, including the advantages, disadvantages, and possible subsequent optimization of the experimental method. Finally, we summarize the whole paper and look ahead to future work in Section 5.

---

**Algorithm 2** Wall extraction method
 

---

**Input:**  $S_r(n \times 3 \text{ points})$   
**Output:**  $S_c(n_1 \times 3 \text{ points}), S_g(n_2 \times 3 \text{ points}), S_w(n_3 \times 3 \text{ points})$   
 1:  $plane0 \leftarrow S_r$  **do** PCA(fit)  
 2:  $S_1 \leftarrow S_r$  on plane0  
 3:  $S_2 \leftarrow S_r$  under plane0  
 4:  $P_1, P_2 \leftarrow S_1, S_2$  **do** PCA(dimension reduction)  
 5:  $P_1, P_2$  **do** mesh( $m \times n$  grid)  
 6: **for** each grid **do**  
 7:      $reserve \leftarrow \text{number of grid} = 1$   
 8:      $remove \leftarrow \text{others}$   
 9: **end for**  
 10:  $P_1, P_2$  **do** SOR  
 11:  $plane1, plane2 \leftarrow P_1, P_2$  **do** PCA  
 12:  $S_c \leftarrow (\text{distance from plane1}) < \text{threshold1}$  in  $S_r$   
 13:  $S_g \leftarrow (\text{distance from plane2}) < \text{threshold2}$  in  $S_r$   
 14:  $S_w \leftarrow \text{others}$

---



---

**Algorithm 3** Data format conversion method
 

---

**Input:**  $S_w(n_3 \times 3 \text{ points})$   
**Output:**  $S_p(n_4 \times 3 \text{ points}), S_{image}(m_0 \times n_0 \text{ image})$   
 1:  $S_{p1} \leftarrow S_w(\text{projected})$   
 2:  $S_{p1}$  **do** mesh  
 3: **for** each grid **do**  
 4:     neighbors search by radius  
 5: **end for**  
 6:  $S_{p2} \leftarrow \text{remove noise by threshold3}$   
 7: **for**  $(x_i, y_i) \in S_{p2}$  **do**  
 8:      $(x_i, y_i) = ((x_i, y_i) - (x_{min}, y_{min})) \times n_a$   
 9: **end for**  
 10:  $S_{p2}$  **do** mesh  
 11: **for**  $(x_i, y_i) \in S_{p2}$  **do**  
 12:      $(x_j, y_j) = \text{round}(x_i, y_i)$   
 13: **end for**  
 14:  $S_p \leftarrow (x_j, y_j), j = 1, 2, \dots, n_{grid}$   
 15:  $m_0 = \max(x_j \in S_p), n_0 = \max(y_j \in S_p)$   
 16:  $S_{image} \leftarrow m_0 \times n_0 \text{ image}$   
 17: **for**  $(x_j, y_j) \in S_p$  **do**  
 18:      $S_{image}(x_j, y_j) = 0 \text{ or } 1(\text{against the background})$   
 19: **end for**

---



**Algorithm 4** Line segment extraction method

---

Input:  $S_{image}(m_0 \times n_0)$  (background is 0 and others are 1)  
Output: lines

- 1:  $S_{con} \leftarrow$  convolve the  $S_{image}$  with kernel1
- 2: **for** each pixel  $(x_p, y_p) \in S_{con}$  **do**
- 3:    $S_{image}(x_p, y_p) = 0$  **if**  $S_{con}(x_p, y_p) < 4$
- 4:    $S_{image}(x_p, y_p) = 1$  **if**  $S_{con}(x_p, y_p) > 4$
- 5: **end for**
- 6:  $S_s \leftarrow S_{image}$  **do** skeleton extracting
- 7: **for** random  $(x_i, y_i) \in S_s$  **do**
- 8:   Calculate the nearest neighbor  $(x_{i1}, y_{i1})$
- 9:    $k(x_i, y_i) = \frac{y_{i1} - y_i}{x_{i1} - x_i}$ ,  $k(x_{i1}, y_{i1}) = k(x_i, y_i)$
- 10:    $k(x_{i1}, y_{i1})$  is the next seed
- 11: **end for**
- 12: **if** two neighbors of  $k(x_i, y_i)$  is satisfied with  $k_1 = k_2$
- 13:    $k(x_i, y_i) = k_1$
- 14: **end if**
- 15: **for**  $(x_j, y_j) \in S_s$  **do**
- 16:   **if**  $(x_j, y_j)$  is (two neighbors &  $k_1 \neq k_2$ ) or (one neighbor) or (no neighbor)
- 17:     $(x_j, y_j)$  is the seed<sub>0</sub>
- 18:   **end if**
- 19:   **for** seed<sub>1</sub>  $(x_j, y_j) \in line_j$  **do**
- 20:     $(x_{j0}, y_{j0})$  is the neighbor of  $(x_j, y_j)$  &  $k(x_{j0}, y_{j0}) = k(x_j, y_j)$
- 21:     $line_j \leftarrow (x_{j0}, y_{j0})$
- 22:     $(x_{j0}, y_{j0})$  is the new seed<sub>1</sub>
- 23:   **end for**
- 24: **end for**
- 25: **for** each  $line_{j1}, line_{j2} \in lines$  **do**
- 26:   endpoint  $p_{j1}, p_{j2} \in line_{j1}, line_{j2}$
- 27:   **if** only one endpoint is satisfied with distance  $(p_{j1}, p_{j2})$  &  $k_{line_{j1}} = k_{line_{j2}}$
- 28:    same line  $\leftarrow line_{j1}, line_{j2}$
- 29:   **end if**
- 30: **end for**
- 31: **for** each  $line_j$  **do**
- 32:   endpoint  $(x_1, y_1), (x_2, y_2)$ , midpoint  $x_0 = \sum_{i=1}^n x_i$ ,  $y_0 = \sum_{i=1}^n y_i$ , slope  $k_j$
- 33:   new endpoint  $(x'_1, y'_1) = \left( \frac{y_1 + k_j x_0 - y_0 + x_1 k_j}{2k_j}, \frac{x_1 k_j - k_j x_0 + 2y_0}{2} \right)$
- 34:   new endpoint  $(x'_2, y'_2) = \left( \frac{y_2 + k_j x_0 - y_0 + x_2 k_j}{2k_j}, \frac{x_2 k_j - k_j x_0 + 2y_0}{2} \right)$
- 35: **end for**
- 36: lines  $\leftarrow$  each endpoint  $(x'_{i1}, y'_{i1}), (x'_{i2}, y'_{i2})$

---

**2. Methods**

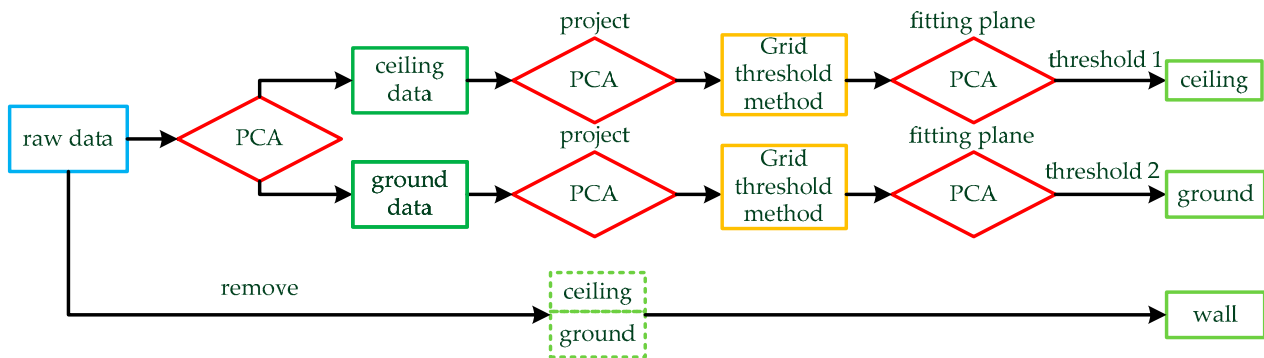
In this section, we detail the principle of our approach in four steps: (1) A wall extraction method based on PCA; the data are divided into three sections: floor, ceiling, and other data. (2) Conversion from 3D to 2D data, including projection, filtering, and voxelization. (3) Extraction of line segments based on the image; the image is skeletonized and we propose a line segment-extraction algorithm based on the slope of adjacent points. (4) Reconstruction of the indoor structure; we reconstruct the interior structure using linear information.

### 2.1. Extraction of Wall Based on PCA

PCA can be used to extract the principal components of data, reduce the dimensionality of data, or fit planes or lines. In this section, we use the theory of PCA to find the floor and ceiling of the data and solve the problem that the floor may not be the x-y plane in the x-y-z frame.

The principle of PCA is shown in Algorithm 1. The input  $S_I \in \mathbb{R}^{n \times 3}$  is the three-dimensional point cloud. Then, the average  $\mu$  of the point clouds is calculated. For the point  $p_i \in S_I$ , the deviation  $p_j$  of  $p_i$  and  $\mu$  is calculated, and the deviation matrix  $S_P \in \mathbb{R}^{n \times 3}$  consists of  $p_j$ . Next, the covariance matrix  $\Gamma$  is calculated by  $S_P$  and we obtain the transformation matrix  $V = [v_1, v_2, v_3] \in \mathbb{R}^{3 \times 3}$  by singular value decomposition (SVD).  $v_1$  and  $v_2$  are taken as vectors of the fitting plane, and  $v_3$  is taken as a normal vector to that plane. Finally, we project the raw 3D data to obtain the 2D data by matrix operation  $S_O = S_I[v_1, v_2]$ .

Our first step is the process of finding the plane based on PCA. The main flow of this method is shown in Figure 2 and Algorithm 2.  $S_r \in \mathbb{R}^{n \times 3}$  is used as the raw data for Algorithm 2.



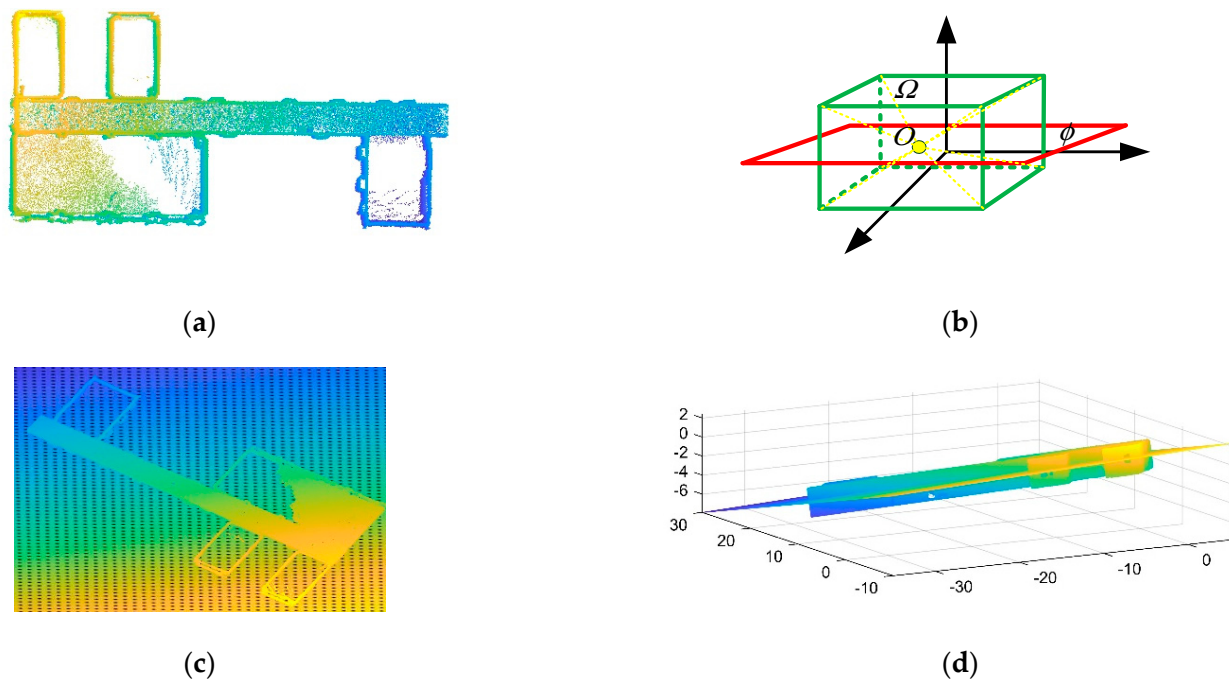
**Figure 2.** Wall extraction method based on PCA: PCA is used to fit the plane for the raw data, and the data are divided into two parts by the plane: ceiling data and ground data. The two parts of data are projected into two-dimensional data by PCA, and then the grid threshold method is used to find the feature points of the ground and ceiling. PCA is used to fit the feature points to obtain the level of the ground and ceiling. Finally, thresholds are set for the two planes and the point clouds of the floor and ceiling are removed.

$S_r \in \mathbb{R}^{n \times 3}$  is used as the raw data in Algorithm 2. To better represent the effect of the algorithm, we use test data to verify our algorithm, as shown in Figure 3a. The sum  $s$  of the distances of all points  $(x_i, y_i, z_i)$  from the plane  $Ax + By + Cz + D = 0$  is minimum by fitting regular data as follows:

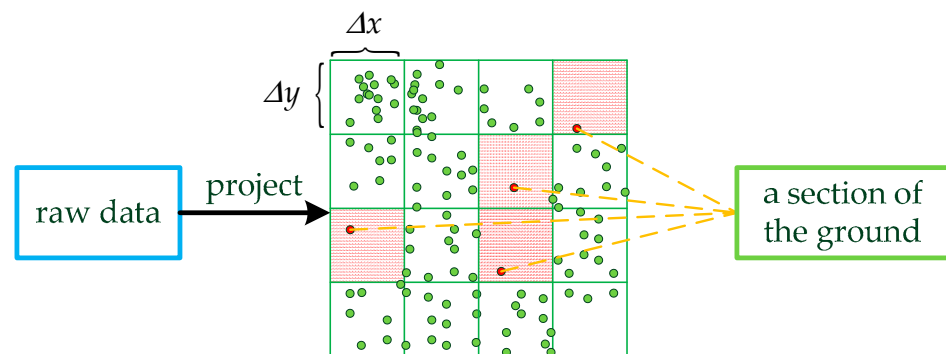
$$s = \min \left\{ \sum_{i=1}^n \frac{|Ax_i + By_i + Cz_i + D|}{\sqrt{A^2 + B^2 + C^2}} \right\} \quad (1)$$

Therefore, the plane fitted by the symmetric point cloud is in the middle of the data, as shown in Figure 3b. Due to indoor noise and other objects, the position of the plane is changed, but the data are also divided into upper and lower parts. The upper part  $S_1$  contains the ceiling and the lower part  $S_2$  contains the floor, as shown in Figure 3c,d.

Then,  $S_1$  and  $S_2$  are projected on planes and converted into two-dimensional  $P_1 \in \mathbb{R}^{n_1 \times 2}$  and  $P_2 \in \mathbb{R}^{n_2 \times 2}$ .  $P_1$  and  $P_2$  are processed by the grid threshold method and we introduce the method using  $P_1$  as an example. The method is shown in Figure 4.



**Figure 3.** PCA is used to find cross-sections of the data: (a) raw point cloud data; (b) suppose that  $\Omega$  is a cuboid composed of point clouds and  $O$  is the average value of data;  $\phi$  is the plane determined by the normal vector and  $O$  obtained from PCA; (c,d) the plane fitted in the raw data by PCA.



**Figure 4.** The grid threshold method. The original data are projected two-dimensionally and meshed. Part of the ground data consists of these points whose quantity is one in the grid.

We set the number  $n_{grid}$  of grids, and then calculate the horizontal resolution  $\Delta x$  and vertical resolution  $\Delta y$  of the grid according to the maximum values  $x_{max}$  and  $y_{max}$  of  $P_1$  in two directions as follows:

$$\Delta x = \frac{x_{max}}{n_{grid}}, \Delta y = \frac{y_{max}}{n_{grid}} \quad (2)$$

Points  $(x_i, y_i)$  in the same grid  $(i_{grid}, j_{grid})$  are assigned the same number:

$$i_{grid} = \text{round}\left(\frac{x_i - x_{min}}{\Delta x}\right), j_{grid} = \text{round}\left(\frac{y_i - y_{min}}{\Delta y}\right) \quad (3)$$

Some edge problems are caused by rounding the number of points. Therefore, we take some measures as follows:

$$i_{grid}, j_{grid} = \begin{cases} 1, & i_{grid}, j_{grid} = 0 \\ n_{grid}, & i_{grid}, j_{grid} > n_{grid} \end{cases} \quad (4)$$

For the number  $(i_{grid}, j_{grid})$  of each point, we replace 0 with 1 and replace the points that are larger than  $n_{grid}$  with  $n_{grid}$ . The data whose number corresponds to one point are taken as the plane points. However, due to the interference of noise, we use the SOR filter to obtain the exact plane points. These points are fitted into a plane by PCA, and  $P_1$  and  $P_2$  correspond to the plane:

$$\begin{aligned} plane1 : A_1x + B_1y + C_1z + D_1 &= 0 \\ plane2 : A_2x + B_2y + C_2z + D_2 &= 0 \end{aligned} \quad (5)$$

Since the ceiling and floor are parallel, we take the average of the normal vectors of the two planes as our new normal vector:

$$\begin{aligned} plane1 : \frac{A_1+A_2}{2}x + \frac{B_1+B_2}{2}y + \frac{C_1+C_2}{2}z + D_1 &= 0 \\ plane2 : \frac{A_1+A_2}{2}x + \frac{B_1+B_2}{2}y + \frac{C_1+C_2}{2}z + D_2 &= 0 \end{aligned} \quad (6)$$

Finally, we set *threshold1* and *threshold2* of the distance from the point to the planes to determine the ceiling  $S_c$  and the floor  $S_g$ , and the wall  $S_w$  is the data  $S_r$  except for  $S_c$  and  $S_g$ . The entire process is shown in Table 1.

**Table 1.** The process of extracting the wall: Project the initial data onto a plane, and then the ground data are screened by the grid threshold method. The ground is restored to a three-dimensional form and treated with the SOR filter. The ground and ceiling are fitted into a plane; then, the wall data are extracted by setting thresholds.

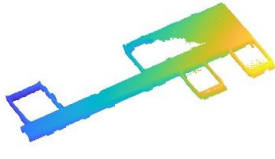
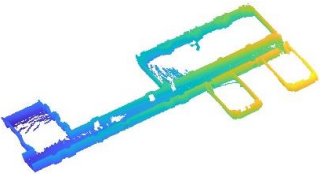
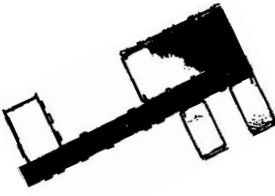
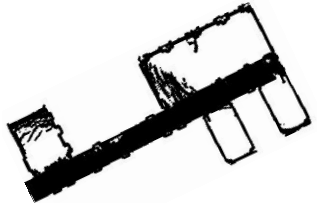
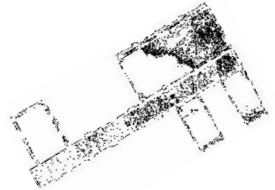
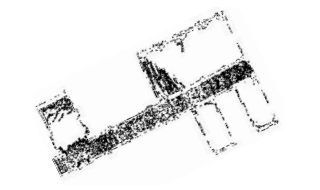

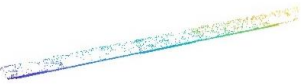


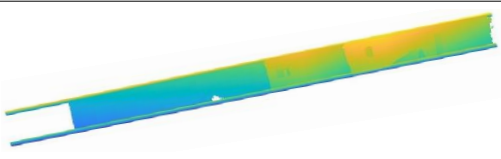
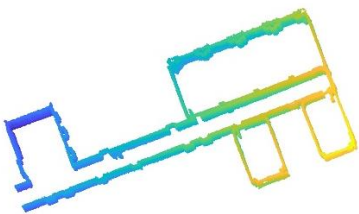
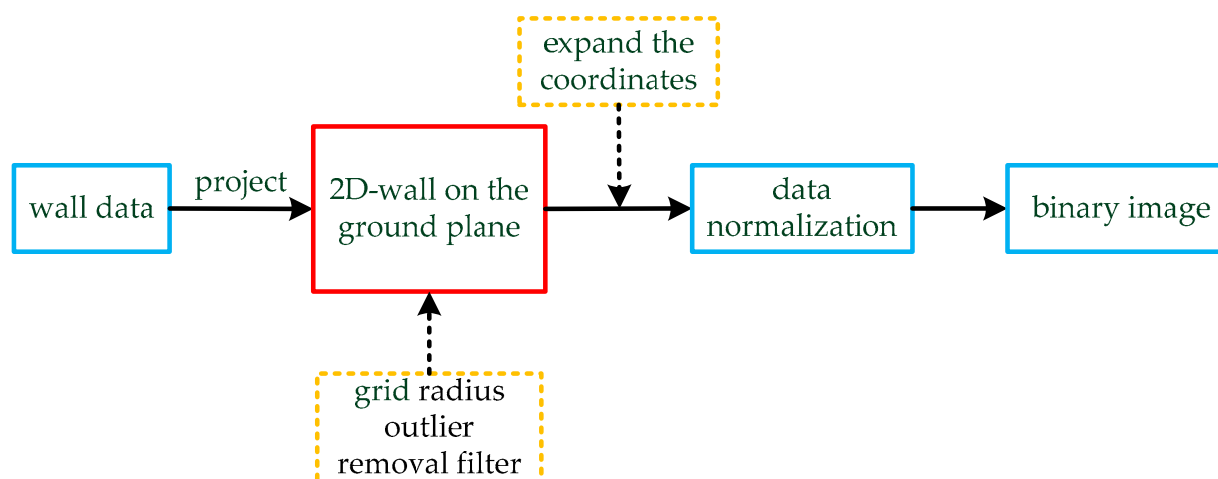
|                       | Ceiling  | Ground  |
|-----------------------|--|---|
| raw data              |  |  |
| 2D project            |  |  |
| grid threshold method |  |  |

Table 1. Cont.

|                  | Ceiling  | Ground  |
|------------------|--|---|
| 3D recovery      |  |  |
| SOR              |  |  |
| fit plane        |  |   |
| extract the wall |  |   |

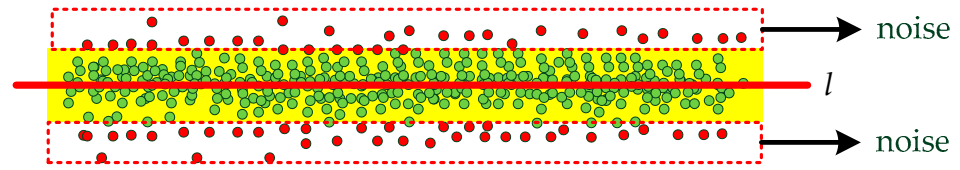
## 2.2. Data Format Conversion

The point cloud data are so dense and disordered that it is difficult to extract information from them. However, point cloud slicing leads to the loss of the contour, which increases the difficulty of subsequent processing. Therefore, to improve the efficiency of the algorithm, we projected the wall onto the ground and meshed it. The entire process is shown in Figure 5 and Algorithm 3.



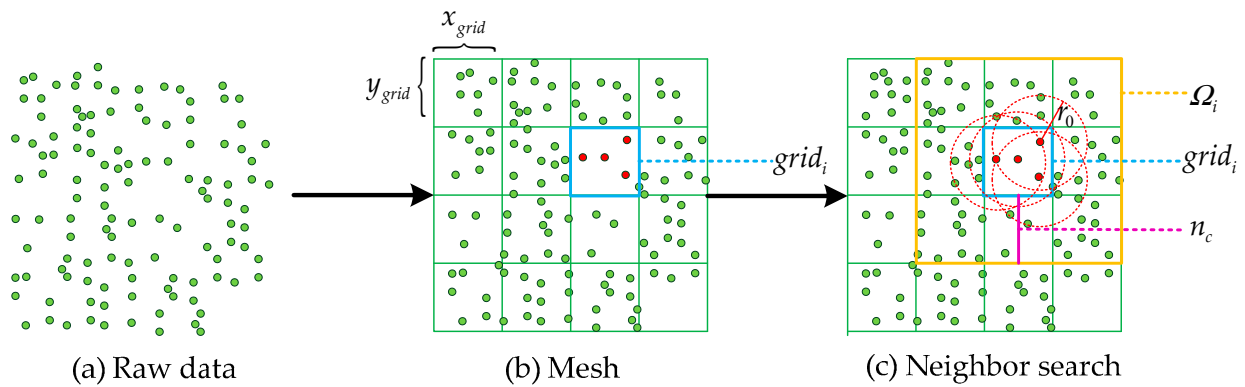
**Figure 5.** The process of data conversion. The wall data are projected on the ground plane, which was fitted in the previous section. The two-dimensional data are processed by a grid radius outlier removal (GROR) filter, and the coordinate values are enlarged by appropriate multiples. Finally, the data are normalized and transformed into a binary image.

$S_w \in \mathbb{R}^{n_3 \times 3}$  is used as the raw data in Algorithm 3, and  $S_w$  is projected into  $S_{p_1}$  on the ground plane. Due to the influence of noise, the points of a line have a certain width, and we extract the line features from these points. First, we need to remove the noise of the two-dimensional points, as shown in Figure 6.



**Figure 6.** The noise in the points. We need to extract the line  $l$  and remove the noise.

The radius outlier removal (ROR) [49] filter is an effective tool for noise removal. If the ROR filter is not optimized in any way, this method is very inefficient. Therefore, we use the GROR (grid radius outlier removal) filter to optimize the ROR filter. We mesh the data and then search for the nearest neighbor points in a specific way. The results show that the efficiency of this method is greatly improved without affecting the accuracy of the ROR filter. At the same time, due to the high density of the vertical wall point and the low density of other sundries points after projection, some sundries points are removed by the GROR filter. The principle of this method is shown in Figure 7.



**Figure 7.** Grid radius outlier removal (GROR) filter. The resolutions of the divided grid are  $x_{grid}$  and  $y_{grid}$ . The grid to be processed is  $grid_i$ .  $n_c$  is the number of adjacent grids, and the search area  $\Omega_i$  is determined by  $n_c$ .  $r_0$  is the radius of the neighbor search.

First, the raw data  $S_{p_1}$  are divided into grids, and the resolutions in the two directions are, respectively,  $x_{grid}$  and  $y_{grid}$ . The points in each grid are searched for their nearest neighbors. If only the points in the grid are used as the search scope, some of the nearest neighbors will be lost at the edge of the grid. Therefore, we have expanded the area to ensure the accuracy of the search. For the grid  $(i_{grid}, j_{grid})$ , we use the number  $n_c$  of adjacent grids to expand the region.  $n_c$  is determined by the search radius  $r_0$  and resolutions:

$$n_c = \text{ceil} \left( \frac{r_0}{\min(x_{grid}, y_{grid})} \right) \quad (7)$$

The radius  $r_0$  divided by the minimum resolution and ceiled is called  $n_c$ . Therefore, the search area  $\Omega_i$  for the grid  $(i_{grid}, j_{grid})$  is as follows:

$$\Omega = \begin{bmatrix} (i_{grid} - n_c, j_{grid} - n_c) & \dots & (i_{grid} - n_c, j_{grid} + n_c) \\ \dots & (i_{grid}, j_{grid}) & \dots \\ (i_{grid} + n_c, j_{grid} - n_c) & \dots & (i_{grid} + n_c, j_{grid} + n_c) \end{bmatrix} \quad (8)$$



Since the mesh size  $n_{grid} \times n_{grid}$  is fixed, edge suppression is added as follows:

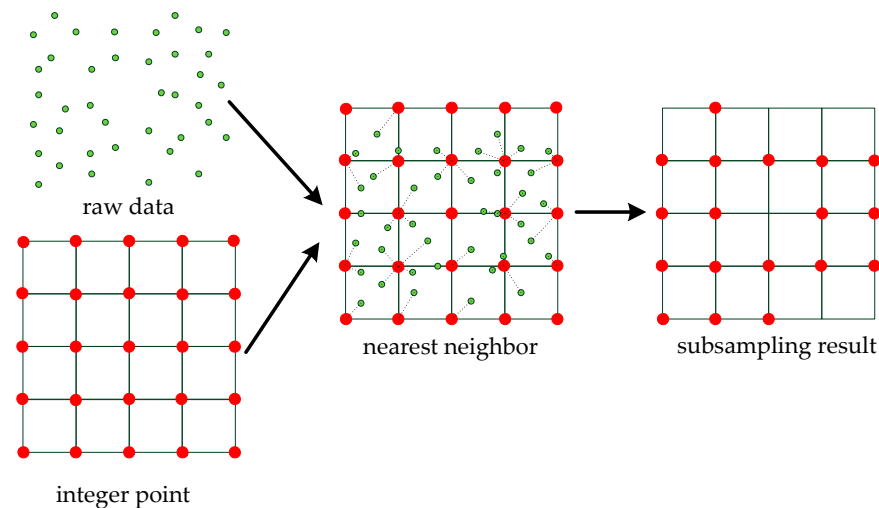
$$\begin{cases} i_{grid} - n_c = 1 \text{ if } (i_{grid} - n_c) < 1 \\ i_{grid} + n_c = n_{grid} \text{ if } (i_{grid} + n_c) > n_{grid} \\ j_{grid} - n_c = 1 \text{ if } (j_{grid} - n_c) < 1 \\ j_{grid} + n_c = n_{grid} \text{ if } (j_{grid} + n_c) > n_{grid} \end{cases} \quad (9)$$

Finally, the points whose number of neighbors is less than *threshold3* are removed. After removing the noise, we need to expand the coordinates of the points  $S_{p_2}$ . We chose a large multiple  $n_a$  to make the details of the data more obvious as follows:

$$(x_i, y_i) = ((x_i, y_i) - (x_{min}, y_{min})) \times n_a \quad (10)$$

For each point  $(x_i, y_i) \in S_{p_2}$ , we subtract the minimum value  $(x_{min}, y_{min})$  of the point to make sure the points are positive and use the multiple  $n_a$  to expand the coordinate of these points. Since the coordinates need to be rounded after expanding, and then restored after processing, the details can be better reflected and the error can be reduced by a larger multiple. Assume a point is  $x_s = a_0.a_1a_2a_3$ , and when it expands 100 times it becomes  $x_s = a_0a_1a_2.a_3$ . We round it to  $x_s = a_0a_1a_2$  and then restore it to  $x_s = a_0.a_1a_2$ . The error before and after treatment was less than 0.01.

Then, we use the grid subsampling method to round the coordinates and eliminate the repeated coordinates of the same position. The benefit of this method is that the data become regular and the amount of data is greatly reduced. The principle of the method is shown in Figure 8.



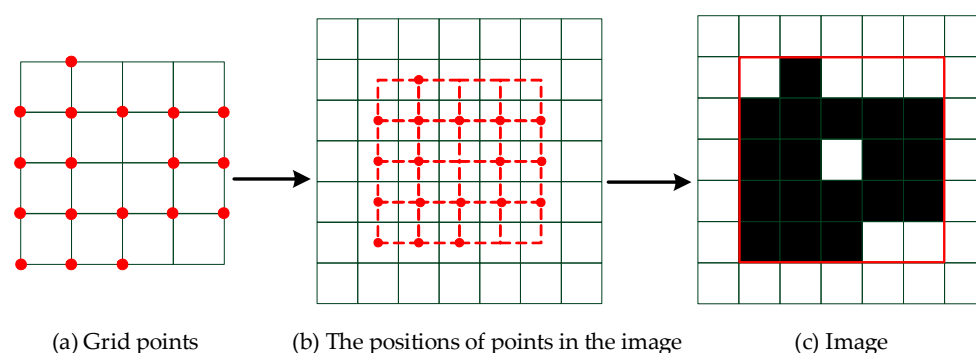
**Figure 8.** The grid subsampling method. Given the coordinates of integer points, each point in the raw data is updated to the nearest integer point. The shape of the data is retained and the amount of data is reduced in the subsampling result  $S_p$ .

For the raw data  $(x_i, y_i) \in S_{p_2}$  and integer points  $(x_i^{int}, y_j^{int}) \in S_{integer}$ ,  $(x_i, y_i)$  is replaced by the point  $(x_i^{int}, y_j^{int})$  nearest to  $(x_i, y_i)$ . Then, the overlapping points are removed. Finally, we obtain the data  $S_p \in \mathbb{R}^{n_4 \times 3}$  after the subsampling. For each point  $(x_i, y_i) \in S_p$ , we need to replace 0 with 1 as follows:

$$\begin{cases} x_i = 1 \text{ if } x_i = 0 \\ y_i = 1 \text{ if } y_i = 0 \end{cases}, i = 1, 2, \dots, n_4 \quad (11)$$

Since the coordinates of the data  $S_p$  are all positive integers, the data can be converted into images to prepare for the next step. This process is shown in Figure 9. The size  $m_0 \times n_0$  of the image  $S_{image}$  is determined by the maximum value of the coordinates in the two directions of the point  $(x_i, y_i) \in S_p$  as follows:

$$\begin{aligned} m_0 &= \max(x_j \in S_p) \\ n_0 &= \max(y_j \in S_p) \end{aligned} \quad (12)$$



**Figure 9.** The process of converting grid points to the image: the pixel corresponding to the positions of the grid points in the image is given the value  $\alpha$ . The image is made up of pixels in the red box in the final step.

Additionally, the pixel of the position corresponding to  $S_p$  in the image  $S_{image}$  is replaced by  $\alpha$ :

$$S_{image}(x_j, y_j) = \alpha \quad (13)$$

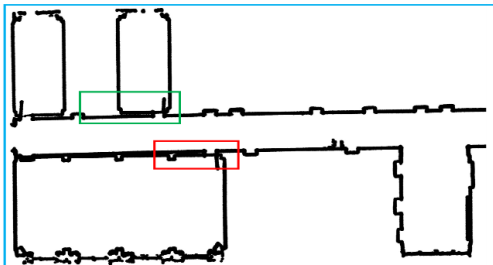
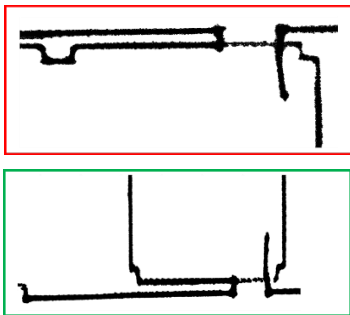

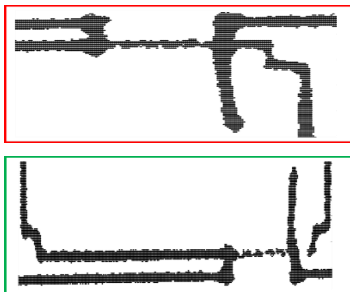
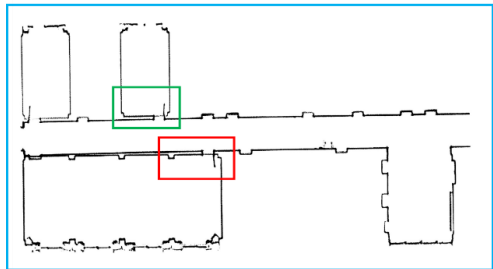
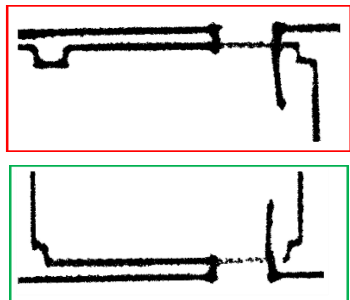
where the parameter  $\alpha$  is against the background; for example, the parameter  $\alpha$  is 0 and the background is 1 in the image.

Therefore, we obtain two sets of data: the two-dimensional points with a grid structure and the corresponding images of the two-dimensional points. The shape of the raw data is retained and the scattered points are regularized by the two data. The experimental procedure in this section is shown in Table 2.

**Table 2.** Projection and transformation of data. First, the raw data are projected onto the ground plane. Then, the noise is removed by the GROR filter. We enlarge the coordinates 70 times and the grid subsampling method is used to make the data regular. The regular data can be converted to an image, which is prepared for the next section.

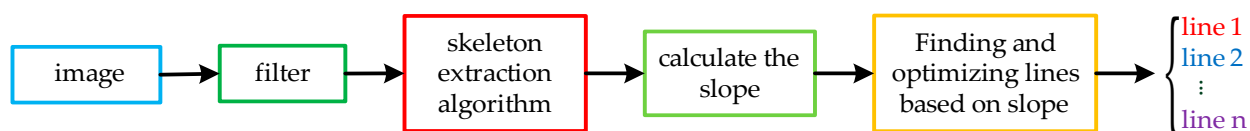
|  | Result | Details |
|--|--------|---------|
| the raw data are projected to the ground |        | <br>    |

Table 2. Cont.

|  | Result  | Details   |
|--|---|---|
| GROR filter  |    |    |
| grid subsampling method (the coordinates are magnified 70 times) |    |    |
| convert to the image   |  |  |

### 2.3. Extraction of Line Segments

For the image in the previous section, we need to extract line segment information from it. Since straight lines are composed of points with a certain thickness in the data, it is difficult to extract multiple lines directly from the data. Therefore, we first use the thinning algorithm to extract the skeleton of the image. Then, the corresponding slope of each point is calculated based on the nearest neighbor point. Finally, we propose a slope-based search method to extract the line segments. We give the flow of the algorithm as shown in Figure 10 and Algorithm 4.

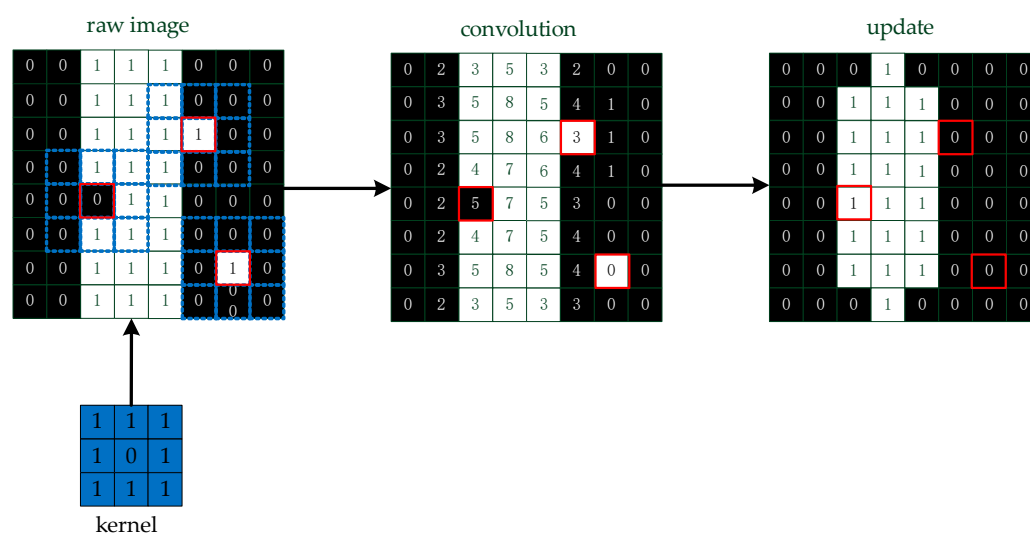


**Figure 10.** The process of line segment extraction. First, filtering is used to remove the burrs from the image, and existing skeleton extraction algorithms are used to extract the image algorithm. Then, we calculate the slopes of the points and optimize the slopes accordingly. Finally, we judge all lines according to the specified rules.

The image  $S_{image}$  is the input in Algorithm 4. We assume that the background is 0 and the points are 1 in the image, and we convolve the image  $S_{image}$  with a kernel to remove the burr, which is influential in extracting the skeleton. Then, the image  $S_{image}$  is updated by the convolution value  $S_{con}$ :

$$\begin{cases} S_{image}(x_p, y_p) = 0 & \text{if } S_{con}(x_p, y_p) < 4 \\ S_{image}(x_p, y_p) = 1 & \text{if } S_{con}(x_p, y_p) > 4 \end{cases}, (x_p, y_p) \in S_{image} \quad (14)$$

The pixel value is updated by comparing the convolution value  $S_{con}$  with the number 4, as shown in Figure 11.



**Figure 11.** A simple method to remove the burr. We convolve the image and update the pixels of the image based on the convolution values.

To better reflect the contour features of the wall, we consider refining the image. If the pixel representing the line is refined into a single pixel, the feature of the line is more obvious in the skeleton. Therefore, we use a suitable thinning algorithm to extract the two-dimensional skeleton information of the wall through experiments. The results are shown in Table 3.

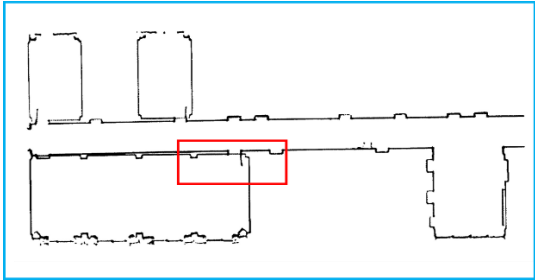
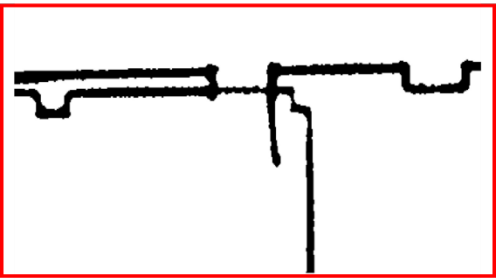

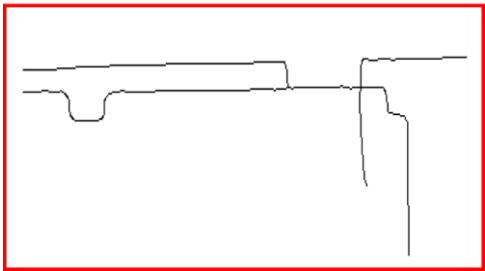
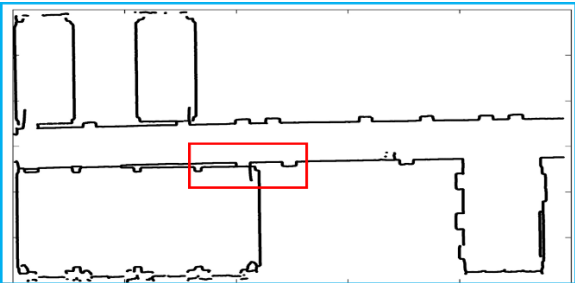
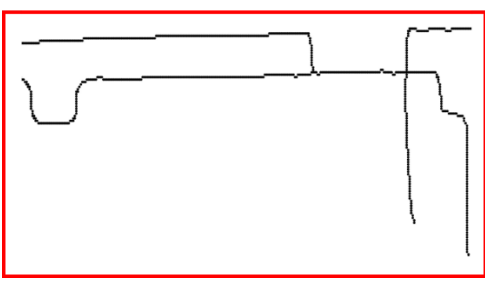
The image skeleton  $S_s$  is a single-pixel structure; we find that the characteristics of the lines are obvious, and the corners of the data affected by the error have a certain radian. Therefore, it is very difficult to extract the exact corners from the data; thus, we consider extracting the lines and determining the corners from the lines. We propose a line extraction method based on the nearest neighbor slope, and then introduce this method in detail.

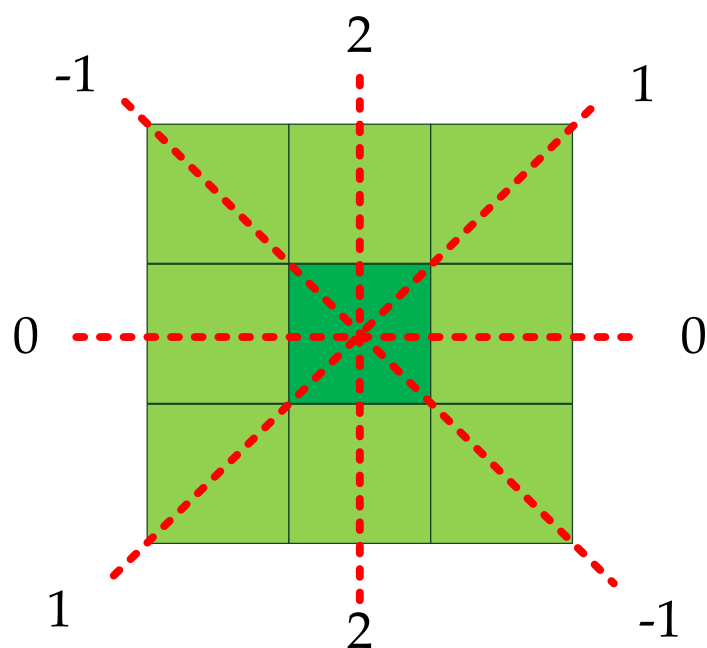
First, the random point  $(x_i, y_i) \in S_s$  is the starting point; its nearest neighbor  $(x_{i1}, y_{i1})$  is found and the slope of the two points is calculated. In this section, the neighbor points are all found in the 8 neighborhoods. If there are multiple nearest neighbor points, select one as the nearest neighbor point at random. Then, their slope  $k$  is calculated as follows:

$$k = \frac{y_{i1} - y_i}{x_{i1} - x_i} \quad (15)$$

The slope of these two points is represented by the slope  $k$ . Since the pixel positions in the image are all integers, there are only four cases for the slope of two adjacent points: 0, 1,  $-1$ , and the slope that does not exist is replaced by 2, as shown in Figure 12.

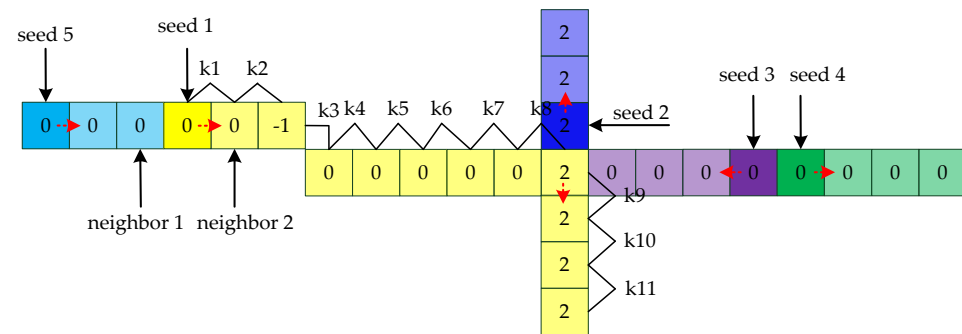
**Table 3.** Image skeleton extraction. The contour of the image is extracted by the skeleton extraction algorithm and lines are refined into single pixels. Similarly, the skeleton made up of two-dimensional points is reduced by the image skeleton.

|                 | Data  | Details  |
|-----------------|---|--|
| raw image       |    |    |
| image skeleton  |    |    |
| points skeleton |  |  |



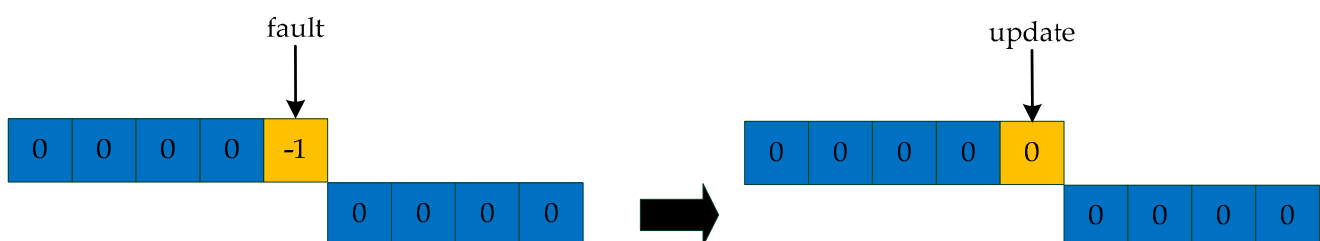
**Figure 12.** The slope of the nearest neighbor. The slope is given four values: 0, 1, −1, 2.

Then, the point  $(x_{i1}, y_{i1})$  is the next seed and its nearest neighbor  $(x_{i2}, y_{i2})$  is found to calculate the next slope, and the points whose slopes have already been calculated are not considered next neighbor search objects. The starting point  $(x_i, y_i) \in S_s$  is randomly selected until the seed point  $(x_{in}, y_{in})$  has no neighbors. If the slopes of all points are calculated, the algorithm stops. The process of calculating the slopes is shown in Figure 13.



**Figure 13.** The process of calculating the slope. First, seed 1 is randomly selected in the image. Seed 1 has multiple nearest neighbor points and picks one at random to calculate the slope value 0. The values of seed 1 and neighbor 2 are 0. Then, neighbor 2 is the next seed 1. The points for which the slope has been calculated are not searched later. Seed 1 stops growing if the current point has no neighbors. Then, seed n is randomly selected and the same growth process is carried out until all the pixels of the image are calculated.

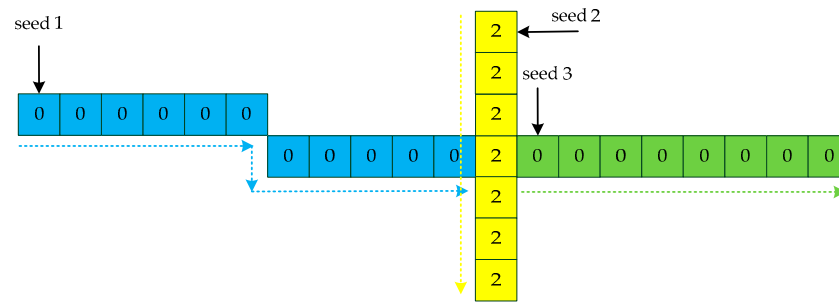
After calculating the slope at all points, we need to eliminate the fault in the line, as shown in Figure 14. For each point, if it only has two neighbors and the two neighbors have the same slope  $k$ , update the slope of that point to  $k$ .



**Figure 14.** Eliminate faults in lines. The fault is caused by noise and algorithm error, and we update the slope of the current point using the slope of the point's nearest neighbor.

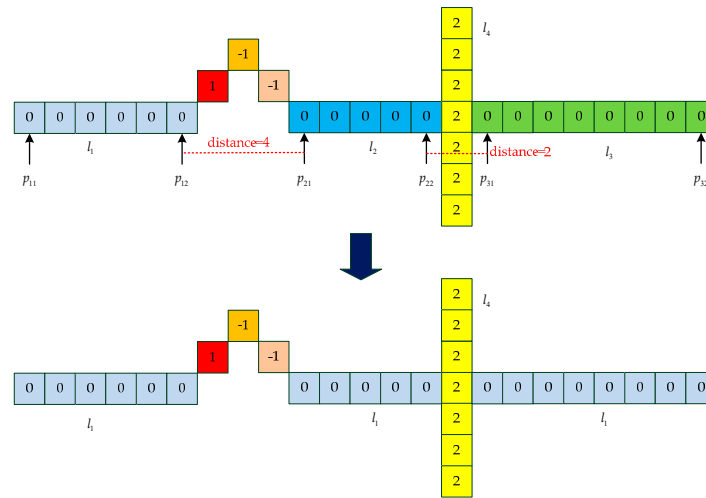
When the slope of all the points is calculated, we look for lines based on the slope. The idea is to start at one endpoint of each line as a seed point and grow until the line is completely extracted. Therefore, we first need to look for endpoints as seed points. We use the following three conditions to determine the seed point  $(x_j, y_j)$ : (1) the number of neighbors is two and the slopes of the neighbors are satisfied with  $k_1 \neq k_2$ ; (2) the number of neighbors is one; (3) no neighbors. The point  $(x_j, y_j)$  that satisfies any of these conditions is considered a seed. Next, we extract points on the same line from the seed, as shown in Figure 15. For a straight line, look for a point  $(x_{j0}, y_{j0})$  within the 8 neighborhoods of the seed  $(x_j, y_j)$  that has the same slope as the seed. In these data, the number of the point  $(x_{j0}, y_{j0})$  is only 1 or 0. If the number is 1, the point  $(x_{j0}, y_{j0})$  is the next new seed, and the raw point  $(x_j, y_j)$  is removed. If the number is 0, the points we calculated before form a line. Keep picking seed points until all the points are divided into lines.





**Figure 15.** Line searching method based on slopes. Determine the seed point according to the judgment condition, and then find the line according to the slope of the nearest neighbor point until traversing all the points.

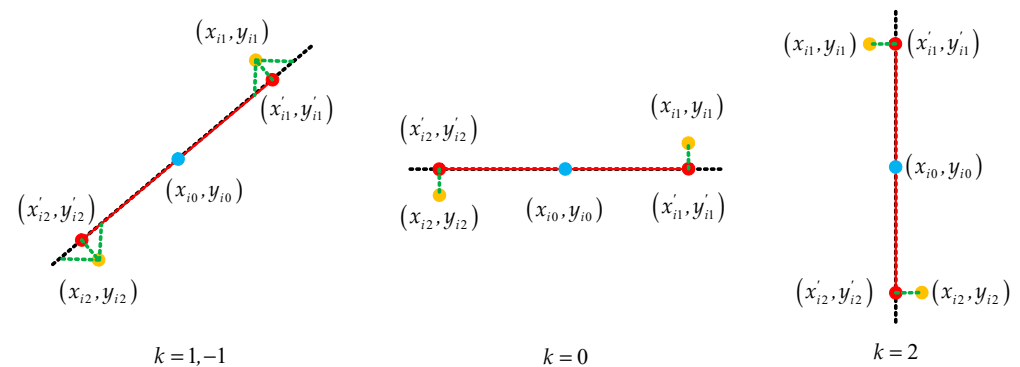
When all the lines are categorized, we need to set thresholds to eliminate non-line points. Due to our conditions applied to individual points and short line segments, lines containing fewer points are removed. Since the same line is divided into sections by noise, we propose a method to solve this problem. As shown in Figure 16, for two endpoints  $(p_{j1}, p_{j2})$  of each line, two lines are the same line only if the distance of two endpoints is less than the threshold and the two lines have the same slope. For the line  $l_i$ , we compare the line  $l_1$  to the lines  $l_2, l_3, \dots, l_n$ , then compare the line  $l_2$  to the lines  $l_3, l_4, \dots, l_n$ , until, finally, we compare the line  $l_{n-1}$  to the line  $l_n$ . When all lines have been identified, repeat the steps until the number of lines is changeless.



**Figure 16.** An example for updating lines. We set the distance threshold to 5. For the line  $l_1$  and the line  $l_2$ , the distance between the endpoint  $p_{12}$  and the endpoint  $p_{21}$  is less than the threshold. Therefore, the two lines are classified as the same line. In the same way, the line  $l_2$  and the line  $l_3$  are classified as the same line.

For these lines, due to the points that make them up not being in a line, we need to figure out their slopes and endpoints to update the lines. The principle of updating the endpoints is shown in Figure 17. We know the parameter of the line  $l_i$ : two endpoints  $(x_{i1}, y_{i1}), (x_{i2}, y_{i2})$ , midpoint  $\left(x_{i0} = \sum_{j=1}^n x_{ij}, y_{i0} = \sum_{j=1}^n y_{ij}\right)$ , and slope  $k_i$ . If the slope is 1 or  $-1$ , we can obtain new endpoints based on these parameters as follows:

$$\begin{cases} (x'_{i1}, y'_{i1}) = \left( \frac{y_{i1} + k_i x_{i0} - y_{i0} + x_{i1} k_i}{2k_i}, \frac{x_{i1} k_i - k_i x_{i0} + 2y_{i0}}{2} \right) \\ (x'_{i2}, y'_{i2}) = \left( \frac{y_{i2} + k_i x_{i0} - y_{i0} + x_{i2} k_i}{2k_i}, \frac{x_{i2} k_i - k_i x_{i0} + 2y_{i0}}{2} \right) \end{cases} \quad (16)$$



**Figure 17.** The process of determining new endpoints. The dotted line is defined by the midpoint  $(x_{i0}, y_{i0})$  and the slope  $k$ . If the slope is 1 or  $-1$ , the vertical point of the raw endpoints on the line is the new endpoint. We calculate the points in which the original endpoint intersects the line in both the horizontal and vertical directions; the midpoint of the points is a new endpoint. If the slope is 0, the abscissa of the new endpoints is the same as the raw endpoints, and the ordinate is the same as the midpoint. If the slope is 2, the abscissa of the new endpoints is the same as the midpoint, and the ordinate is the same as the raw endpoints.

If the slope is 0, we obtain new endpoints as follows:

$$\begin{cases} (x'_{i1}, y'_{i1}) = (x_{i1}, y_{i0}) \\ (x'_{i2}, y'_{i2}) = (x_{i2}, y_{i0}) \end{cases} \quad (17)$$

If the slope is 2, we obtain new endpoints as follows:


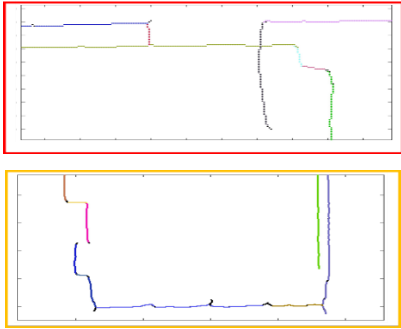
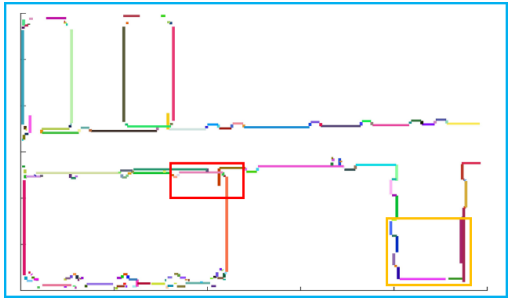
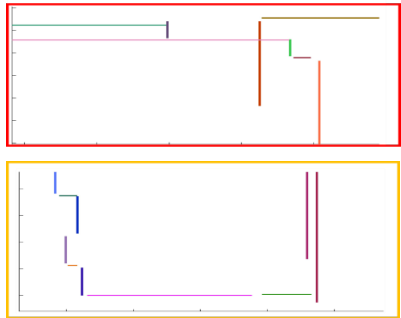
$$\begin{cases} (x'_{i1}, y'_{i1}) = (x_{i0}, y_{i1}) \\ (x'_{i2}, y'_{i2}) = (x_{i0}, y_{i2}) \end{cases} \quad (18)$$

Therefore, the line is determined by the midpoint and the slope, and the range of the line is determined by the new endpoints. Finally, we show the process of finding the straight lines and drawing them all, as shown in Table 4.

**Table 4.** The process of finding lines. In these images, each line segment is represented by a separate color except black. First, we find points that belong to the same line by using neighboring points. In detail, we can see that points that originally belong to the same line are divided into multiple lines due to noise and algorithm. Then, by optimizing the line based on endpoint and slope, we solve the problem. Finally, we draw the lines by slope, midpoint, and new endpoints.

|   | Result | Details |
|---|--------|---------|
| find the line according to the nearest neighbor |        |         |

Table 4. Cont.

|   | Result   | Details  |
|---|--|--|
| update lines based on endpoint and slope        |   |   |
| update the endpoints and draw the line segments |  |  |

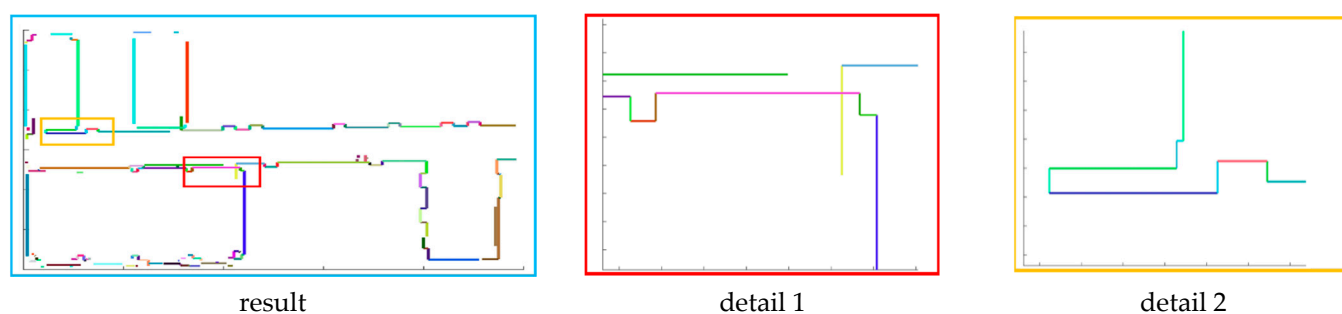
#### 2.4. Structure Reconstruction

In this part, we optimize the lines and reduce them to three-dimensional planes. In the last section, due to the error of scan registration, we removed the smooth intersection of the adjacent segments. Therefore, we propose a method to connect the lines. The basic steps of this method are as follows:

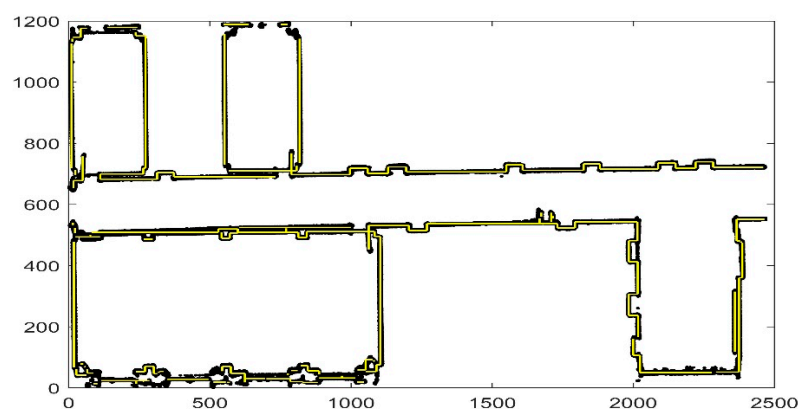
1. Assuming that the known condition is the endpoint  $p_1$ , the nearest endpoint  $p_2$  of the other lines is calculated. Additionally, we then calculate the distance  $d$  between the two endpoints.
2. The distance threshold is set to avoid two lines that are far apart from each other being connected. Then, we compare the distance  $d$  to the threshold.
3. If the distance is less than the threshold and the two lines are perpendicular, the lines corresponding to the two endpoints are connected and the endpoints  $p_1$  and  $p_2$  are replaced by the intersection. For the slopes  $k_1$  and  $k_2$  of the two lines, we think these two lines are perpendicular if they satisfy the condition:

$$|k_1 - k_2| = 2 \text{ or } \|k_1\| - \|k_2\| = 1 \quad (19)$$

The results are shown in Figure 18. To better demonstrate the accuracy of the experimental results, we spliced and compared the lines with the raw data, as shown in Figure 19.

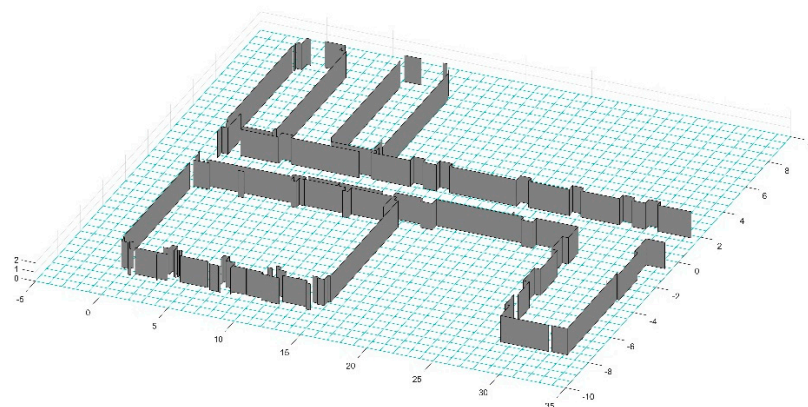


**Figure 18.** The result of line correction. All the lines that satisfy the condition are connected and we show them in detail.



**Figure 19.** Comparison of experimental results. The yellow line is the result that we obtain and the black points are the raw data. We obtain the basic outline of the data, and the reason why some parts are not considered lines is that the points are discontinuous or the slope changes are complicated.

Finally, we reconstruct the three-dimensional plane from the resulting lines. Now, we need to figure out the height of each line. In the previous section, we calculated the ground plane and calculated the height of each point above the ground. Before we achieve this, the endpoints are restored by Equation (10). We calculate the nearest neighbors of the midpoints of each line segment and choose the height of the line as the maximum distance between these neighbors and the ground. If the height is close to the distance between the ceiling and the floor, the height changes to that distance. The result is shown in Figure 20. At this point, we have introduced the principle of the whole algorithm in detail. In the next section, we will evaluate the performance of the algorithm with multiple sets of data.

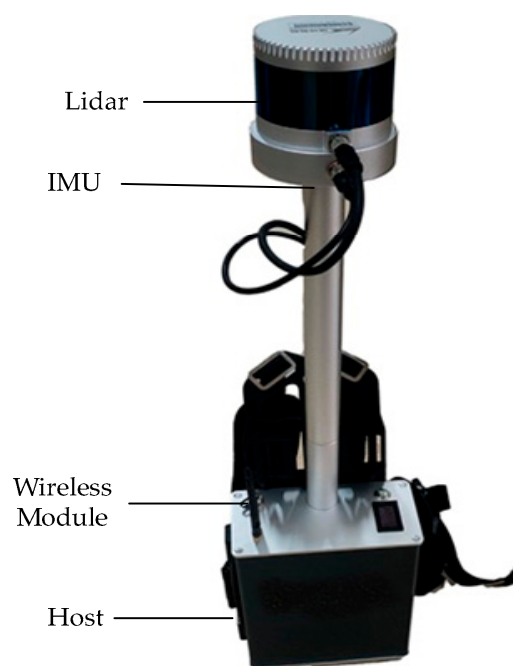


**Figure 20.** The result of three-dimensional reconstruction. To make the result clearer, we removed the ceiling and painted all the vertical planes on the ground.

### 3. Experimental Results

#### 3.1. Preparation of Experimental Data

The equipment used to collect data is shown in Figure 21. The experimental equipment was a backpack system integrating LiDAR, an inertial measurement unit (IMU), a wireless module, and an upper computer. The surroundings are sensed by a 16-line mechanical LiDAR at a frequency of 0.5 s and surface information about surrounding objects is scanned by the LiDAR. The IMU is used to measure the attitude information of LiDAR and we fuse it with each frame point cloud by the robotic operating system (ROS) in the upper computer. The processed data are transmitted to a computer for display through the wireless module.



**Figure 21.** The experimental equipment. It is a backpack system integrating LiDAR, an inertial measurement unit (IMU), a wireless module, and an upper computer.

We adopted LeGO-LOAM to implement the scan registration of data. In the algorithm, the attitude  $[t_z, \theta_{roll}, \theta_{pitch}]$  is obtained from the ground features, and the attitude  $[t_x, t_y, \theta_{yaw}]$  is calculated according to the remaining point cloud features. Therefore, the matching error is caused by too few ground features, and another error is caused by the Euler angles turning too fast in a short time. In the following experimental results, these two aspects may be the causes of the error. Finally, we present some important parameters for the next three scenarios: the number  $n_{fac}$  of grids for the floor and ceiling is extracted in Section 2.1. Radius  $r_{GROR}$  and threshold  $n_{GROR}$  of the GROR filter, magnification  $n_{moc}$  of the coordinates, the distance  $d_p$  that connects parallel lines, and the distance  $d_v$  between the vertical lines are shown in Table 5.

**Table 5.** Some important parameter settings for the scenarios.

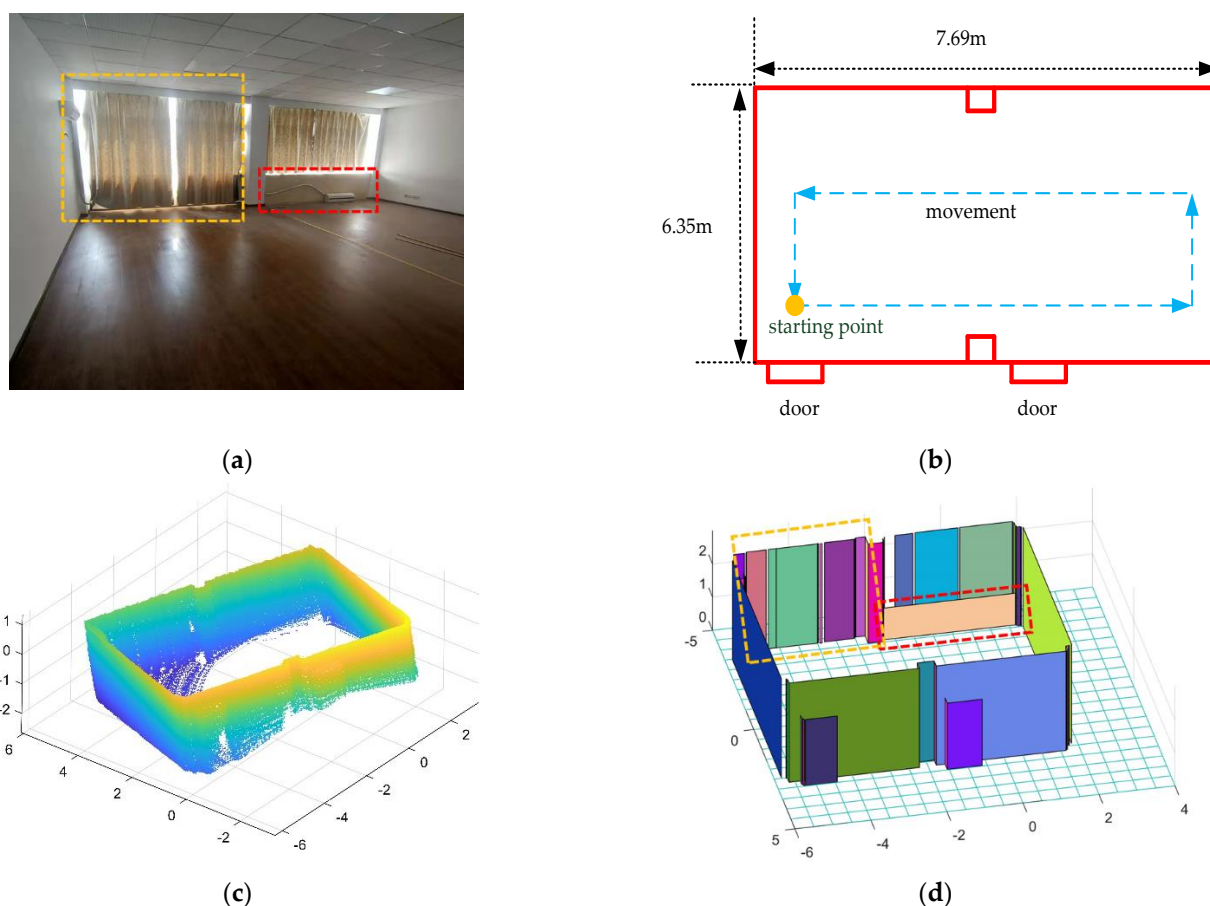
|         | $n_{fac}$ | $r_{GROR}$ | $n_{GROR}$ | $n_{moc}$ | $d_p$ | $d_v$ |
|---------|-----------|------------|------------|-----------|-------|-------|
| Scene 1 | 200       | 0.01       | 10         | 100       | 10    | 40    |
| Scene 2 | 600       | 0.03       | 10         | 10        | 5     | 8     |
| Scene 3 | 300       | 0.03       | 10         | 10        | 15    | 5     |

### 3.2. Results

In this section, we use three real scenarios to test our algorithm. We carried the equipment on our backs and the LiDAR was about 1.8 m away from the ground. We followed a fixed trajectory and tried to keep the posture of the device stable as we moved. Finally, three data points are prepared: a room, a floor, and some rooms.

#### 3.2.1. Experiment Scene 1

We extracted the structure of a single room in the first data point; the room was about  $7.69\text{ m} \times 6.35\text{ m} \times 3\text{ m}$  (length  $\times$  width  $\times$  height). The experimental scene is shown in Figure 22a, and the two-dimensional plan of the scene is shown in Figure 22b. To better show the interior structure, we removed the ceiling in the second section. The measured point cloud data are shown in Figure 22c, and the final result is shown in Figure 22d.



**Figure 22.** The resulting presentation of scene 1: (a) photo of the real scene; (b) the two-dimensional plane of the scene; we measured the dimensions of the room and marked the starting point and trajectory; (c) point cloud data after removing the ceiling; (d) the result of the reconstruction of the interior structure, and each color represents a plane.

It can be seen from the experimental results that the basic structure of the room has been restored, but we can see that there are some flaws in the results. There are gaps between some adjacent planes. In the previous sections, we joined planes if they were adjacent and the distance between them was less than the threshold. In real scene 1, due to the glass between the curtains not being sensed by the LiDAR, we kept the gaps in the result. At the same time, the large gaps due to measurement errors were preserved.

#### 3.2.2. Experiment Scene 2

We chose the floor of the school building as the second scenario to test the method. The point cloud we collected is shown in Figure 23a and the result of algorithm pro-



cessing is shown in Figure 23b. Similarly, we present the two-dimensional plan of the floor in Figure 23c, and we provide real photos of the two positions on the floor and the corresponding reconstruction results, as shown in Figure 23d–g.

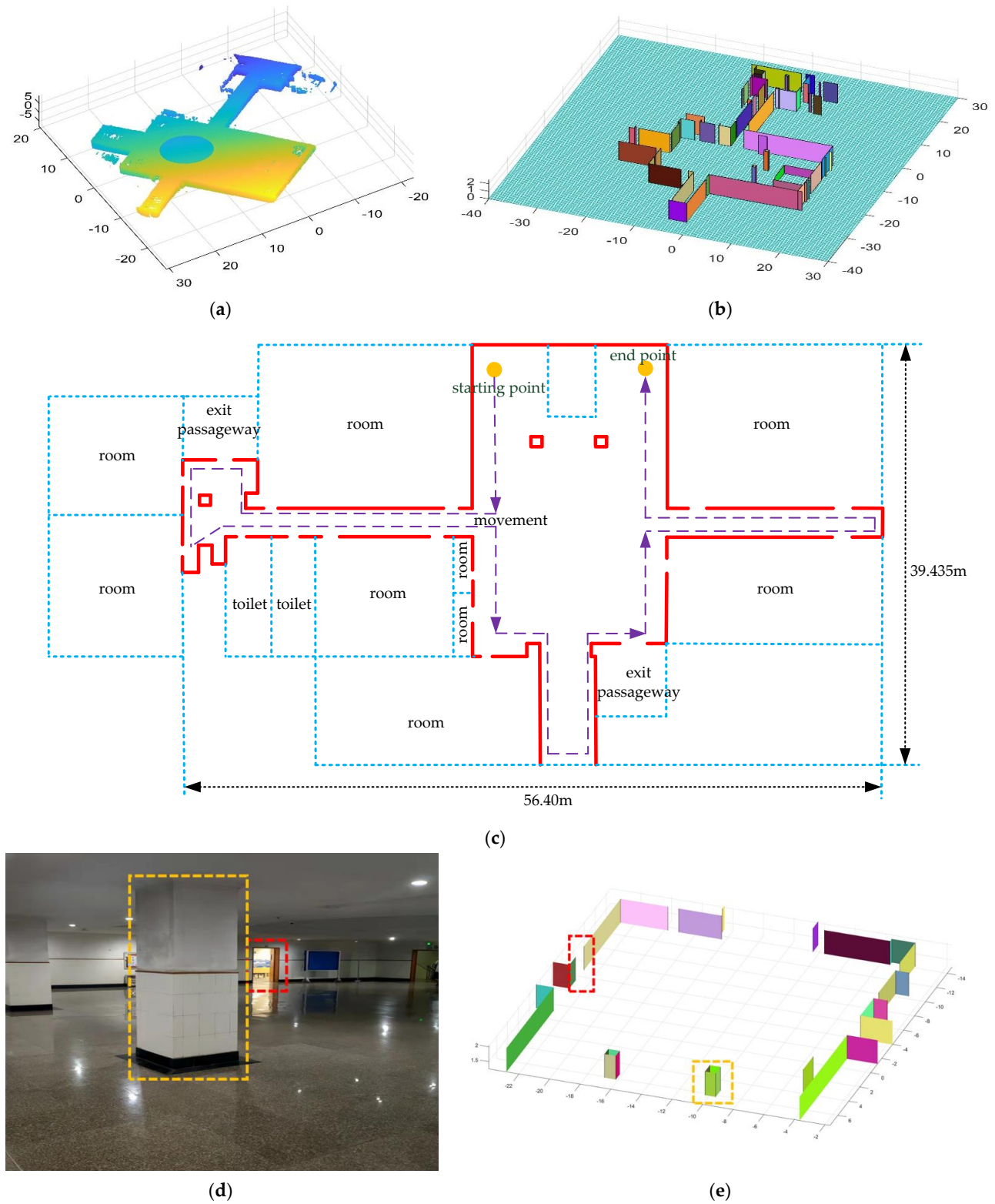
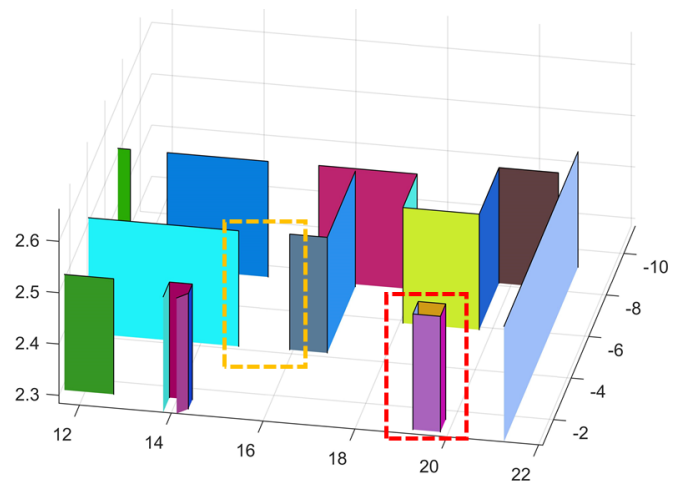


Figure 23. Cont.



(f)



(g)

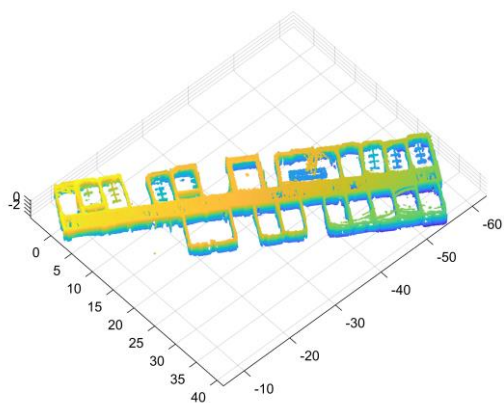
**Figure 23.** The resulting presentation of scene 2: (a) point cloud data of scene 2; (b) the experimental results obtained by our algorithm; to better show the interior, we removed the ceiling; (c) we provide the two-dimensional plan of the floor and mark out the rooms, the toilets, the safe passage, and our movements; the gaps between the lines are the doors; (d,e) are the first part of the real scene and the comparison of experimental results; (f,g) are the second part of the real scene and the comparison of experimental results.

Comparing the results with the real scene, we can see that all the walls are well restored. At the time of measurement, we did not enter any rooms. Some of the doors were open and others were closed, which is why some of the doors were empty in our results. At the same time, due to the attention to detail in our method, uneven sampling and the error of the matching algorithm may result in faults in the same plane.

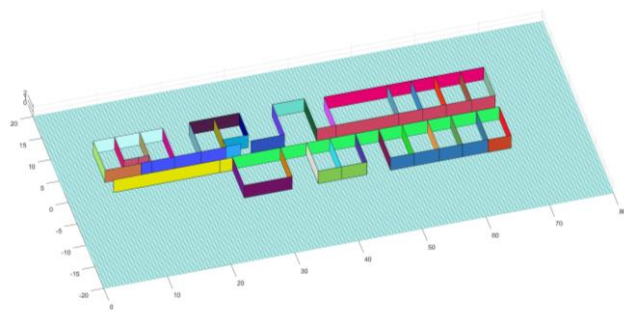
### 3.2.3. Experiment Scene 3

In scene 3, we chose an interior environment with multiple rooms. The raw data are shown in Figure 24a, and the result is shown in Figure 24b. We drew the two-dimensional plan of the data as shown in Figure 24c and a comparison of the result with the real world as shown in Figure 24d,e.

We entered most of the rooms during the measurement and all the doors were closed. Due to the complexity of the scene corresponding to the data, there was a major difference between the two frames when the device entered the room, which led to a major error in scan registration. Therefore, the error of our results is larger than that of the first two data points. However, all the rooms were restored well, so the result was satisfactory.

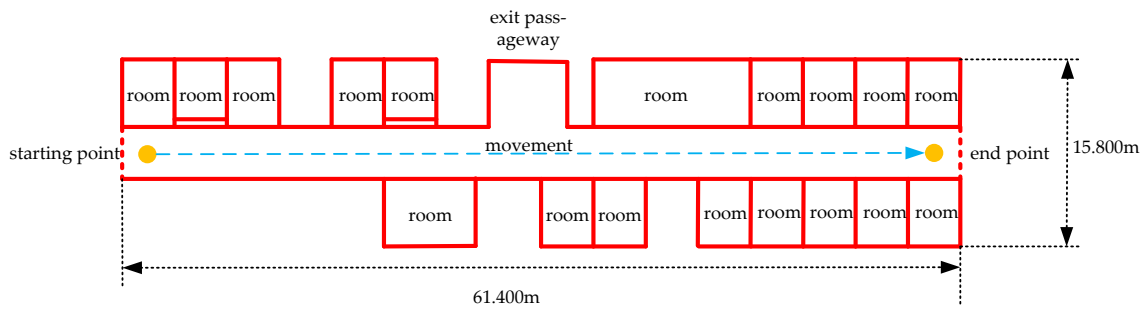


(a)



(b)

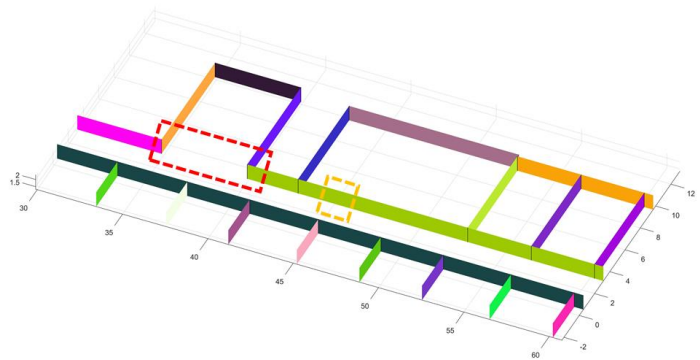
**Figure 24.** Cont.



(c)



(d)



(e)

**Figure 24.** The resulting presentation of scene 3: (a) point cloud data of scene 3; (b) the experimental results; (c) the two-dimensional plan of the data; we moved and scanned each room for a few seconds as we passed it; (d,e) we marked two locations in the simulation results that corresponded to the real world.

### 3.2.4. Relevant Parameters of the Experiment

We aim to obtain indoor structural information in a fast and efficient way, so we list running time, data volume, and distance parameters. In our approach, we employed a series of ways to reduce the amount of data. Therefore, on the premise of ensuring the authenticity and validity of parameters, and to demonstrate the efficiency of our method, we calculated the time of the following parts in Table 6: the time  $t_{c-g}$  of fitting the ceiling and ground, the time  $t_{f-v}$  of projection, filtering, and voxelization, and the time  $t_{i-r}$  from image to three-dimensional reconstruction. To reflect the changes in the amount of data in the process, we counted the number  $n_{raw}$  of initial points, the number  $n_{wall}$  of wall points, the number  $n_{voxel}$  of two-dimensional points after projection and voxelization, and the size  $i_{image} \times j_{image}$  of the image, as shown in Table 7. To verify the accuracy of the algorithm, we compare it with the actual data using the following aspects in Tables 8 and 9: ceiling equation  $A_c x + B_c y + C_c z + D_c = 0$  and ground equation  $A_g x + B_g y + C_g z + D_g = 0$  after first calculation, the number  $n_{plane}$  of planes, the length  $l_s$ , width  $w_s$ , height  $h_s$  of the building simulation data, the length  $l_r$ , width  $w_r$ , height  $h_r$  of the building real data, the error between the actual value and the predicted value of the three aspects  $\sigma_l$ ,  $\sigma_w$ ,  $\sigma_h$ , and the mean precision  $\zeta$ . In Table 10, we list the characteristics of different reconstruction methods in recent years. The function and purpose of each method are not completely the same. To reflect the innovation of our method and solve the problem, we evaluate it while considering the following aspects: (1) sources of data; the effect of the algorithm can be affected by different data; (2) ground calculation; the ground is found as the base by our method without any conditions such as LiDAR attitude; (3) filtering; if no filtering is used, the algorithm has strong robustness; (4) data optimization; data optimization shows that the algorithm has high processing performance; (5) the manifestation of doors and windows; the details show the precision of the algorithm; (6) accuracy of the number of planes; if all planes are found, the accuracy of the algorithm is high.

**Table 6.** The running time of several algorithms.

|         | $t_{c-g}$ | $t_{f-v}$ | $t_{i-r}$ |
|---------|-----------|-----------|-----------|
| Scene 1 | 4.064 s   | 45.432 s  | 1.995 s   |
| Scene 2 | 7.735 s   | 57.559 s  | 2.037 s   |
| Scene 3 | 5.644 s   | 78.373 s  | 2.034 s   |

**Table 7.** The data volume statistics of several steps algorithm.

|         | $n_{raw}$            | $n_{wall}$         | $n_{voxel}$       | $i_{image} \times j_{image}$ |
|---------|----------------------|--------------------|-------------------|------------------------------|
| Scene 1 | $720,224 \times 3$   | $537,721 \times 3$ | $26,349 \times 2$ | $701 \times 776$             |
| Scene 2 | $1,804,574 \times 3$ | $768,450 \times 3$ | $7858 \times 2$   | $406 \times 571$             |
| Scene 3 | $1,568,974 \times 3$ | $756,038 \times 3$ | $2143 \times 2$   | $187 \times 52$              |

**Table 8.** Fitting the parameters of the ceiling and floor equations.

|         | $A_c$   | $B_c$   | $C_c$  | $D_c$   | $A_g$   | $B_g$   | $C_g$  | $D_g$  |
|---------|---------|---------|--------|---------|---------|---------|--------|--------|
| Scene 1 | −0.015  | 0.1674  | 0.9858 | −1.3243 | −0.0181 | 0.1523  | 0.9882 | 1.6697 |
| Scene 2 | 0.2833  | −0.0893 | 0.9549 | −0.7801 | 0.2807  | −0.0877 | 0.9558 | 1.7295 |
| Scene 3 | −0.0039 | 0.0277  | 0.9996 | −1.1052 | −0.0125 | 0.0325  | 0.9994 | 1.4702 |

**Table 9.** Parameter comparison between the actual scenario and the predicted scenario.

|         | $n_{plane}$ | $l_s$    | $w_s$    | $h_s$   | $l_r$    | $w_r$    | $h_r$   | $\sigma_l$ | $\sigma_w$ | $\sigma_h$ | $\xi$   |
|---------|-------------|----------|----------|---------|----------|----------|---------|------------|------------|------------|---------|
| Scene 1 | 53          | 7.652 m  | 6.337 m  | 2.994 m | 7.690 m  | 6.350 m  | 3.000 m | 0.038 m    | 0.013 m    | 0.006 m    | 0.019 m |
| Scene 2 | 86          | 56.360 m | 39.390 m | 2.510 m | 56.400 m | 39.435 m | 2.500 m | 0.040 m    | 0.045 m    | 0.010 m    | 0.030 m |
| Scene 3 | 46          | 61.444 m | 15.815 m | 2.575 m | 61.400 m | 15.800 m | 2.560 m | 0.044 m    | 0.015 m    | 0.015 m    | 0.025 m |

**Table 10.** Comparison of the characteristics of several different methods. Aspect 1: sources of data; Aspect 2: ground calculation; Aspect 3: filtering; Aspect 4: data optimization; Aspect 5: the manifestation of doors and windows; Aspect 6: accuracy of the number of planes.

|                | Aspect 1      | Aspect 2 | Aspect 3 | Aspect 4 | Aspect 5 | Aspect 6 |
|----------------|---------------|----------|----------|----------|----------|----------|
| Ochmann et al. | laser scanner | ✗        | ✓        | ✗        | ✗        | ✓        |
| Ambrus et al.  | laser scanner | ✗        | ✗        | ✓        | ✓        | ✓        |
| Jung et al.    | laser scanner | ✗        | ✓        | ✓        | ✓        | ✓        |
| Wang et al.    | laser scanner | ✗        | ✓        | ✓        | ✓        | ✓        |
| Xiong et al.   | laser scanner | ✗        | ✓        | ✗        | ✓        | ✓        |
| Sanchez et al. | laser scanner | ✗        | ✓        | ✗        | ✓        | ✗        |
| Oesau et al.   | laser scanner | ✗        | ✗        | ✓        | ✓        | ✓        |
| Mura et al.    | laser scanner | ✗        | ✓        | ✗        | ✓        | ✓        |
| ours           | LiDAR         | ✓        | ✓        | ✓        | ✓        | ✓        |

#### 4. Discussion

First, we evaluated the experimental results of the three scenarios. In scenario 1, all vertical planes of the room are created and the basic structure is restored. Because of the curtain, the curved part is removed by our algorithm, which leads to gaps in the plane formed by the curtain. In our algorithm, we expand the coordinates of the data, and the uneven sampling areas and the walls with windows contain fewer points. These areas are removed by our method. Therefore, our method can better restore the details of the room by enlarging the coordinates, while the room is restored to a rectangle if normal processing is used. In scenario 2, due to the large area of the scene, we stretched the result vertically for better detail. We can see that all the open doors are identified as gaps between the planes, and by comparing the coordinates in the result, we can ensure that the plane precision is high; for example, the width of the door is about 1 m. The whole floor is well restored, although there are still a few errors in the plan. This is where our method will need to be optimized at a later stage. In scenario 3, we went into some rooms selectively, and the results show that the reconstruction is better. All the rooms are perfectly reproduced, though there are some errors due to scan registration.

Second, we analyze and discuss the parameters in five tables. In Table 6, although we optimized the filtering, we can see that most of the time is still spent on filtering and voxelization. However, in terms of the time and effect of restoring the image to a three-dimensional plane, the time  $t_{i-r}$  of about 2 s shows that our slope-based algorithm is efficient and fast. Therefore, we will try other filtering approaches to improve the performance of the program in subsequent work. In Table 7, we can see a gradual decrease in the amount of data, such as in scene 3, the number of points is reduced to  $187 \times 52$  from  $1,568,974 \times 3$ , which is why our approach is efficient. In Table 8, by comparing the parameters of the ceiling and ground planes, we note that they are parallel. The error of the room height calculated from these two planes is small (the maximum error does not exceed 0.015), so our method of finding the plane is accurate. We want to restore a model from the point cloud that is the same size as in the actual scene. In Table 9, by comparing the calculated length, width, and height with the actual value, the error between the predicted value and the actual value is, respectively, less than 0.044 m, 0.045 m, and 0.015 m. The mean error of 0.024 m proves that our algorithm is accurate within the error tolerance. In Table 10, we overcome the difficulty of LiDAR data processing and achieve similar results compared with some current methods. The data from LiDAR are easy to obtain but difficult to process, laser scanner data are difficult to obtain but high precision. We summarize the advantages and disadvantages of some algorithms. Then, we propose some methods to optimize and solve these problems in terms of prior knowledge, noise removal, data optimization, structure extraction, algorithm efficiency, and accuracy. Compared with existing methods, our method is superior to current algorithms in some respects.

Finally, there are still some shortcomings in our experiment. To make the initial data tidier, we will test other algorithms to replace the current algorithm for scan registration, such as LIO-SAM. Our method is currently only applicable to the reconstruction of interior vertical walls, and circles are recognized as polygons by the method. Therefore, we will modify the algorithm to accommodate more complex cases and make it more robust.

#### 5. Conclusions

The method proposed in this paper can restore basic interior structure on an equal scale. In this method, the ceiling and the floor of the room are segmented and fitted, and the ground plane is used as the reference plane. Then, the wall is projected onto the reference plane. We filter, regularize, and transform the resulting two-dimensional data into an image. We extract the skeleton of the image and calculate the slope for each point. The slope that characterizes a line expands by growing until all points of the line are found. These lines are optimized and corrected by our method to produce more accurate lines. Finally, these lines are restored to three-dimensional vertical planes by applying height information. From the experimental results, the reconstruction effect and parameter analysis all show



that our method is accurate (mean error of 0.024 m at an average measuring range of 40 m). In the whole algorithm, we pay attention to the improvement of computing speed and innovation of the algorithm. Therefore, we made a lot of data optimization and algorithm improvement, and we proposed a new framework to solve the problem of indoor reconstruction. However, there are some limitations and errors in this method. For example, parameters need to be adjusted to obtain the best results, and indoor sundries cannot be reconstructed. Therefore, we will focus on algorithm optimization and detail processing to make the algorithm more practical in the future.

**Author Contributions:** Conceptualization, J.Z.; methodology, J.Z. and D.C.; software, D.C.; validation, D.Z. and J.C.; formal analysis, J.Z. and D.C.; resources, D.T.; data curation, D.T.; writing—original draft preparation, D.C.; writing—review and editing, J.Z., D.C. and D.Z.; visualization, D.C.; supervision, J.Z. and J.C.; project administration, J.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** National Natural Science Foundation of China (62105372), Foundation of Key Laboratory of National Defense Science and Technology (6142401200301), Natural Science Foundation of Hunan Province (2021JJ40794).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

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

## References

- Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [\[CrossRef\]](#)
- Jo, K.; Kim, J.; Kim, D.; Jang, C.; Sunwoo, M. Development of Autonomous Car—Part II: A Case Study on the Implementation of an Autonomous Driving System Based on Distributed Architecture. *IEEE Trans. Ind. Electron.* **2015**, *62*, 5119–5132. [\[CrossRef\]](#)
- Yao, S.; Zhang, J.; Hu, Z.; Wang, Y.; Zhou, X. Autonomous-driving vehicle test technology based on virtual reality. *J. Eng.* **2018**, *2018*, 1768–1771. [\[CrossRef\]](#)
- Yaqoob, I.; Khan, L.U.; Kazmi, S.M.A.; Imran, M.; Guizani, N.; Hong, C.S. Autonomous Driving Cars in Smart Cities: Recent Advances, Requirements, and Challenges. *IEEE Netw.* **2020**, *34*, 174–181. [\[CrossRef\]](#)
- Urmson, C.; Anhalt, J.; Bagnell, D.; Baker, C.; Bittner, R.; Clark, M.N.; Dolan, J.; Duggins, D.; Galatali, T.; Geyer, C.; et al. Autonomous driving in urban environments: Boss and the Urban Challenge. *J. Field Robot.* **2008**, *25*, 425–466. [\[CrossRef\]](#)
- Gelbart, A.; Weber, C.; Bybee-Driscoll, S.; Freeman, J.; Fetzer, G.J.; Seales, T.; McCarley, K.A.; Wright, J. FLASH lidar data collections in terrestrial and ocean environments. In Proceedings of the SPIE 5086, Laser Radar Technology and Applications VIII, Orlando, FL, USA, 21 August 2003; pp. 27–38. [\[CrossRef\]](#)
- Kopp, F. Wake-vortex characteristics of military-type aircraft measured at Airport Oberpfaffenhofen using the DLR Laser Doppler Anemometer. *Aerosp. Sci. Technol.* **1999**, *3*, 191–199. [\[CrossRef\]](#)
- Zhou, L.F.; Yu, H.W.; Lan, Y. Deep Convolutional Neural Network-Based Robust Phase Gradient Estimation for Two-Dimensional Phase Unwrapping Using SAR Interferograms. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 4653–4665. [\[CrossRef\]](#)
- Zhou, L.F.; Yu, H.W.; Lan, Y.; Xing, M.D. Artificial Intelligence In Interferometric Synthetic Aperture Radar Phase Unwrapping: A Review. *IEEE Geosci. Remote Sens. Mag.* **2021**, *9*, 10–28. [\[CrossRef\]](#)
- Matayoshi, N.; Asaka, K.; Okuno, Y. Flight Test Evaluation of a Helicopter Airborne Lidar. *J. Aircr.* **2007**, *44*, 1712–1720. [\[CrossRef\]](#)
- Rabadan, G.J.; Schmitt, N.P.; Pistner, T.; Rehm, W. Airborne Lidar for Automatic Feedforward Control of Turbulent In-Flight Phenomena. *J. Aircr.* **2010**, *47*, 392–403. [\[CrossRef\]](#)
- Tachella, J.; Altmann, Y.; Mellado, N.; McCarthy, A.; Tobin, R.; Buller, G.S.; Tournier, J.Y.; McLaughlin, S. Real-time 3D reconstruction from single-photon lidar data using plug-and-play point cloud denoisers. *Nat. Commun.* **2019**, *10*, 4984. [\[CrossRef\]](#)
- Yang, L.; Sheng, Y.; Wang, B. LiDAR data reduction assisted by optical image for 3D building reconstruction. *Optik* **2014**, *125*, 6282–6286. [\[CrossRef\]](#)
- Li, H.Y.; Yang, C.; Wang, Z.; Wu, G.L.; Li, W.H.; Liu, C. A Hierarchical Contour Method for Automatic 3d City Reconstruction From Lidar Data. In Proceedings of the 2012 IEEE International Geoscience and Remote Sensing Symposium, Munich, Germany, 22–27 July 2012; pp. 463–466. [\[CrossRef\]](#)
- Guiyun, Z.; Bin, W.; Ji, Z. Automatic Registration of Tree Point Clouds From Terrestrial LiDAR Scanning for Reconstructing the Ground Scene of Vegetated Surfaces. *IEEE Geosci. Remote Sens. Lett.* **2014**, *11*, 1654–1658. [\[CrossRef\]](#)
- Hu, H.; Fernandez-Steege, T.M.; Dong, M.; Azzam, R. Numerical modeling of LiDAR-based geological model for landslide analysis. *Autom. Constr.* **2012**, *24*, 184–193. [\[CrossRef\]](#)



17. Kim, C.; Habib, A. Object-Based Integration of Photogrammetric and LiDAR Data for Automated Generation of Complex Polyhedral Building Models. *Sensors* **2009**, *9*, 5679–5701. [[CrossRef](#)] [[PubMed](#)]
18. Kwak, E.; Habib, A. Automatic representation and reconstruction of DBM from LiDAR data using Recursive Minimum Bounding Rectangle. *ISPRS J. Photogramm. Remote Sens.* **2014**, *93*, 171–191. [[CrossRef](#)]
19. Liu, S.; Atia, M.M.; Karamat, T.B.; Noureldin, A. A LiDAR-Aided Indoor Navigation System for UGVs. *J. Navig.* **2014**, *68*, 253–273. [[CrossRef](#)]
20. Liu, Y.; Fan, B.; Meng, G.; Lu, J.; Xiang, S.; Pan, C. DensePoint: Learning Densely Contextual Representation for Efficient Point Cloud Processing. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 5238–5247.
21. Khatamian, A.; Arabnia, H.R. Survey on 3D Surface Reconstruction. *J. Inf. Process. Syst.* **2016**, *12*, 338–357. [[CrossRef](#)]
22. Matiukas, V.; Paulinas, M.; Usinskas, A.; Adaskevicius, R.; Meskauskas, R.; Valincius, D. A survey of point cloud reconstruction methods. *Electr. Control Technol.* **2008**, *3*, 150–153.
23. Valero, E.; Adan, A.; Cerrada, C. Automatic construction of 3D basic-semantic models of inhabited interiors using laser scanners and RFID sensors. *Sensors* **2012**, *12*, 5705–5724. [[CrossRef](#)]
24. Shan, T.X.; Englot, B. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 4758–4765.
25. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Rus, D. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2020; pp. 5135–5142.
26. Ochmann, S.; Vock, R.; Wessel, R.; Klein, R. Automatic reconstruction of parametric building models from indoor point clouds. *Comput. Graph.* **2016**, *54*, 94–103. [[CrossRef](#)]
27. Ochmann, S.; Vock, R.; Klein, R. Automatic reconstruction of fully volumetric 3D building models from oriented point clouds. *ISPRS J. Photogramm. Remote Sens.* **2019**, *151*, 251–262. [[CrossRef](#)]
28. Schnabel, R.; Wahl, R.; Klein, R. Efficient RANSAC for point-cloud shape detection. *Comput. Graph. Forum.* **2007**, *26*, 214–226. [[CrossRef](#)]
29. Saval-Calvo, M.; Azorin-Lopez, J.; Fuster-Guillo, A.; Garcia-Rodriguez, J. Three-dimensional planar model estimation using multi-constraint knowledge based on k-means and RANSAC. *Appl. Soft Comput.* **2015**, *34*, 572–586. [[CrossRef](#)]
30. Ambrus, R.; Claici, S.; Wendt, A. Automatic Room Segmentation From Unstructured 3-D Data of Indoor Environments. *IEEE Robot. Autom. Lett.* **2017**, *2*, 749–756. [[CrossRef](#)]
31. Sanchez, V.; Zakhori, A. Planar 3d Modeling of Building Interiors From Point Cloud Data. In Proceedings of the 2012 IEEE International Conference on Image Processing (ICIP 2012), Orlando, FL, USA, 30 September–3 October 2012; pp. 1777–1780.
32. Wang, C.; Cho, Y.K.; Kim, C. Automatic BIM component extraction from point clouds of existing buildings for sustainability applications. *Autom. Constr.* **2015**, *56*, 1–13. [[CrossRef](#)]
33. Mura, C.; Mattausch, O.; Jaspe Villanueva, A.; Gobbetti, E.; Pajarola, R. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Comput. Graph.* **2014**, *44*, 20–32. [[CrossRef](#)]
34. Oesau, S.; Lafarge, F.; Alliez, P. Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS J. Photogramm. Remote Sens.* **2014**, *90*, 68–82. [[CrossRef](#)]
35. Palomba, C.; Astone, P.; Frasca, S. Adaptive Hough transform for the search of periodic sources. *Class. Quant. Grav.* **2005**, *22*, S1255–S1264. [[CrossRef](#)]
36. Wu, H.; Yue, H.; Xu, Z.; Yang, H.; Liu, C.; Chen, L. Automatic structural mapping and semantic optimization from indoor point clouds. *Autom. Constr.* **2021**, *124*, 103460. [[CrossRef](#)]
37. Wang, C.; Cho, Y. Automated 3D building envelope recognition from point clouds for energy analysis. In Proceedings of the Construction Research Congress 2012: Construction Challenges in a Flat World, West Lafayette, IN, USA, 21–23 May 2012; pp. 1155–1164.
38. Xiong, X.; Adan, A.; Akinci, B.; Huber, D. Automatic creation of semantically rich 3D building models from laser scanner data. *Autom. Constr.* **2013**, *31*, 325–337. [[CrossRef](#)]
39. Tang, S.; Zhang, Y.; Li, Y.; Yuan, Z.; Wang, Y.; Zhang, X.; Li, X.; Zhang, Y.; Guo, R.; Wang, W. Fast and Automatic Reconstruction of Semantically Rich 3D Indoor Maps from Low-quality RGB-D Sequences. *Sensors* **2019**, *19*, 533. [[CrossRef](#)]
40. Tang, S.; Zhu, Q.; Chen, W.; Darwish, W.; Wu, B.; Hu, H.; Chen, M. Enhanced RGB-D Mapping Method for Detailed 3D Indoor and Outdoor Modeling. *Sensors* **2016**, *16*, 1589. [[CrossRef](#)] [[PubMed](#)]
41. Stojanovic, V.; Trapp, M.; Richter, R.; Döllner, J. A service-oriented approach for classifying 3D points clouds by example of office furniture classification. In Proceedings of the 23rd International ACM Conference on 3D Web Technology, Poznań, Poland, 20–22 June 2018; pp. 1–9.
42. Jung, J.; Hong, S.; Yoon, S.; Kim, J.; Heo, J. Automated 3D Wireframe Modeling of Indoor Structures from Point Clouds Using Constrained Least-Squares Adjustment for As-Built BIM. *J. Comput. Civ. Eng.* **2016**, *30*, 04015074. [[CrossRef](#)]
43. Jung, J.; Stachniss, C.; Kim, C. Automatic Room Segmentation of 3D Laser Data Using Morphological Processing. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 206. [[CrossRef](#)]

- 
44. Saeed, K.; Tabędzki, M.; Rybnik, M.; Adamski, M. K3M: A universal algorithm for image skeletonization and a review of thinning techniques. *Int. J. Appl. Math. Comput. Sci.* **2010**, *20*, 317–335. [[CrossRef](#)]
  45. Zhang, T.Y.; Suen, C.Y. A fast parallel algorithm for thinning digital patterns. *Commun. ACM* **1984**, *27*, 236–239. [[CrossRef](#)]
  46. Harris, C.; Stephens, M. A combined corner and edge detector. In Proceedings of the Alvey Vision Conference, Manchester, UK, 31 August–2 September 1988; pp. 10–5244.
  47. Smith, S.M.; Brady, J.M. SUSAN—A new approach to low level image processing. *Int. J. Comput. Vis.* **1997**, *23*, 45–78. [[CrossRef](#)]
  48. Chen, S.; Meng, H.; Zhang, C.; Liu, C. A KD curvature based corner detector. *Neurocomputing* **2016**, *173*, 434–441. [[CrossRef](#)]
  49. Renzhong, L.; Man, Y.; Yuan, R.; Huanhuan, Z.; Junfeng, J.; Pengfei, L. Point cloud denoising and simplification algorithm based on method library. *Laser Optoelectron. Prog.* **2018**, *55*, 1–4. [[CrossRef](#)]