Fast and Automatic Registration of Terrestrial Point Clouds Using 2D Line Features

Wuyong Tao 1,2, Xianghong Hua 1,*, Zhiping Chen 3 and Pengju Tian 1

1 School of Geodesy and Geomatics, Wuhan University, Wuhan 430079, China; wuyong_tao@whu.edu.cn (W.T.); pjtian@whu.edu.cn (P.T.)
2 Department of Geomatics Engineering, University of Calgary, Calgary T2N 1N4, Canada
3 Faculty of Geomatics, East China University of Technology, Nanchang 330013, China; zhpcchen@ecut.edu.cn
* Correspondence: xhhua@sgg.whu.edu.cn

Received: 25 February 2020; Accepted: 16 April 2020; Published: 18 April 2020

Abstract: Point cloud registration, as the first step for the use of point cloud data, has attracted increasing attention. In order to obtain the entire point cloud of a scene, the registration of point clouds from multiple views is necessary. In this paper, we propose an automatic method for the coarse registration of point clouds. The 2D lines are first extracted from the two point clouds being matched. Then, the line correspondences are established and the 2D transformation is calculated. Finally, a method is developed to calculate the displacement along the z-axis. With the 2D transformation and displacement, the 3D transformation can be easily achieved. Thus, the two point clouds are aligned together. The experimental results well demonstrate that our method can obtain high-precision registration results and is computationally very efficient. In the experimental results obtained by our method, the biggest rotation error is 0.5219°, and the biggest horizontal and vertical errors are 0.2319 m and 0.0119 m, respectively. The largest total computation time is only 713.4647 s.

Keywords: point cloud recognition; 2D line feature; 2D transformation; feature extraction; line correspondence

1. Introduction

The terrestrial laser scanning technology has been extensively used in many practical applications such as the reconstruction of scenes [1,2], deformation monitoring [3–6], cultural heritage management [7,8], and the measurement of exploitative volumes over open-pit mines [9]. Due to the limited field of view of a laser scanner, the point cloud of one view is not enough to describe the complete scene. Therefore, a process to register the point clouds of all the views into a common coordinate system is necessary. This process includes two steps: coarse and fine registration. For the scenes that contain man-made objects (e.g., building), there are many line and plane features, which are useful for the point cloud registration. In this paper, our goal is to perform the coarse registration based on 2D line features. The fine registration is usually carried out by the iterative closest point (ICP) algorithm [10] or its variants [11,12]. A large number of methods have been developed for the coarse registration of point clouds. These methods calculate the transformation parameters by extracting different geometric features, including point, line, plane and specific object.

The point-based methods are the most common. Numerous methods first extract the keypoints from two point clouds to be matched, and calculate local shape descriptors for these keypoints. The correspondences between keypoints from different point clouds are established by comparing the descriptor similarity. In [13], the fast point feature histogram (FPFH) descriptor was proposed, and the sample consensus method was developed to select the likely correct correspondences. In [14], the authors designed a new descriptor called quintuple local coordinate image (QLCI), which was
then used to perform the coarse registration of point clouds. Besides, a large number of descriptors have been reported in the literature, including binary shape context (BSC) [1], rotational contour signature (RCS) [15], local voxelized structure (LoVS) [16], and so on. There are also some methods that are different from the before-mentioned methods. For example, the four-point congruent sets (4PCS) [17] is also a point-based method, and some improvements have been made for it, e.g., Super 4PCS [18], keypoint-based 4PCS (K-4PCS) [19] and semantic keypoint-based 4PCS (SK-4PCS) [20]. These methods utilize the rule of intersection ratios to establish the correspondences without calculating the descriptors. In [21], the semantic feature point was defined as the intersection point of the vertical feature line with the ground. The point clouds were then registered by establishing the correspondences between the semantic feature points. Although these methods can obtain enough good registration results, they possess a high computation complexity. This is because the keypoint extraction is time-consuming. Furthermore, the number of the keypoints is generally large, making keypoint matching computationally expensive.

As for the line/plane-based methods, in [22], the authors first segmented the planar features from the point clouds. The linear features were extracted by intersecting the planar features. The angles and distances between all pairs of lines were calculated in order to select the candidate line correspondences. Subsequently, the association-matrix-based sample consensus approach was developed to find the correct line correspondences by the same authors in [23]. In [24], a cached-octree region growing algorithm was first proposed for segmenting the planes. Then, a method for planar segment area calculation was given. If two planar segments from two scans have similar areas, they are treated as a candidate plane correspondence. However, when a planar segment is completely acquired in a scan but partly acquired in another scan, the corresponding planar segments may have different areas. Therefore, applying the planar segment area will reduce the number of the correct plane correspondences. In [25], in order to establish the plane correspondences, the minimally uncertain maximal consensus (MUMC) algorithm was formulated, which leverages the covariance matrices of plane parameters. Xiao et al. [26] employed a point-based region growing method to extract the planar patches, and then Hessian form plane equation and area were calculated for each planar patch. The two attributions were used to search the corresponding planes. In [27], the point clouds were first voxelized, and the planar patches were extracted by merging planar voxels. The geometric constraints (ratios of distance and ratios of angles) of four planar patches were used to find the potential corresponding four planar patches in another scan. Finally, the voxel-based four-plane congruent sets (V4PCS) method was proposed. Pathak et al. [28] proposed a registration method based on large planar segments for simultaneous localization and mapping (SLAM). Fan et al. [29] fist proposed a plane extraction method based on the definition of virtual scanlines, and then a point cloud alignment workflow based on plane features were presented. Stamos and Leordeanu [30] utilized both line and plane features for the point cloud registration of urban scenes. The length of the lines and the area of the planes were used to reject unreasonable feature correspondences. The method will also reduce the number of correct line correspondences and plane correspondences, because the corresponding lines (planes) may have different lengths (areas) due to occlusion. Yao et al. [31] also used both line and plane features to perform the point cloud registration. A probability-based random sample consensus (RANSAC) algorithm was formulated to find the candidate correspondences.

There are a small number of registration methods that employ specific objects. In [7], the crest lines were extracted to implement the point cloud registration of statues. The crest line correspondences were identified by the shape similarity of these lines, which was measured by a deformation energy model. In [32], the authors exploited the objects with regular shape (e.g., plane, sphere, cylinder and tori) for the point cloud registration of industrial scenes. In [33], the octagonal lamp poles were extracted to perform the point cloud registration in a road environment. Due to the use of the specific objects, these methods have a limited scope of application.

In general, the point-based methods are more universal, because this kind of methods are suitable for a variety of scenes. The line/plane-based methods are more robust to noise and point density.
variation. For the scenes with man-made objects, there are a plenty of line and plane features in scenes, so extracting line and plane features to perform the registration of point clouds is a good choice. However, most of the existing line/plane-based methods usually extract the 3D lines and/or 3D planes. This means that these methods process the point cloud data in 3D space. As a result, the extraction of 3D lines and/or 3D planes is rather time-consuming. In this paper, we first propose a method to extract the 2D line features. Then, the 2D transformation is obtained by using the 2D line features. A method is also developed to find the correct line correspondences and calculate the 2D transformation. Owing to the dimensionality reduction, the proposed registration method processes the point clouds in 2D space. This can largely reduce the processing time [34]. In our method, only two pairs of line correspondences are enough to register the point clouds. In addition, we treat the 2D lines as infinite lines rather than line segments and do not use the lengths of the lines to facilitate the search of the candidate line correspondences. Hence, even when the corresponding lines are partly scanned, they can also be used to calculate the 2D transformation. The terrestrial laser scanner is usually leveled, so the scanned point clouds are vertical. Thus, there is no rotation around the x and y axes. Therefore, with the 2D transformation, we can easily achieve the 3D transformation. The main contributions of our work are summarized as follows:

1. A 2D line feature extraction method is proposed. Because we process the point clouds in 2D space, the method is computationally efficient.
2. A method is formulated to search the line correspondences and calculate the 2D transformation.
3. A method is developed to calculate the displacement along z-axis.
4. Finally, a registration method based on the 2D line features is presented. Owing to the use of the 2D line features, the registration method has relatively good time efficiency.

The remainder of this paper is organized as follows. In Section 2, the 2D line extraction method is detailly given. The coarse registration method based on the 2D line features are presented in detail in Section 3. Section 4 shows the experimental results and analysis. The conclusions are discussed in Section 5.

2. D Line Extraction

In our registration method, the 2D line extraction is the first task. In order to get the 2D line features, the points on the 2D lines need to be extracted first. The extraction process is shown in Figure 1. The original point cloud is first projected on the horizontal plane. Because the points on a facade will be projected on a line, and therefore have high point density in the 2D space. The density of projected points (DoPP) method [35] has been proposed to extract the points on 2D lines. In this method, the 2D point cloud is divided into regular mesh grids. Then the number of the points in a grid is treated as the density value of the grid. In order to only extract the points on the 2D lines as possible, the grid size should be set as a very small value. This means that the number of the grids is very large. Thus, the method is very time-consuming. Here, we calculate the number of the neighboring points around a point, and regard it as the density value of the point. The neighborhood radius is set as a small value (0.3pr in this paper, where pr denotes the point cloud resolution). The points with point density being higher than 5 are extracted from the 2D point cloud. The extraction result is shown in Figure 1c. This method is significantly faster to compute than the DoPP method. The extracted points are denoted as point cloud $P_{Line}$. As we can see, the point cloud $P_{Line}$ is highly dense. We simplify the point cloud to further reduce the number of the points. Thus, the computation cost is further reduced. In order not to result in the loss of some regions, the point cloud is simplified according to the space distance. The sampling interval is set as pr, making it have the same point cloud resolution as the original point cloud. The 2D line features are extracted from the simplified 2D point cloud $P_{Line-s}$. 
Then, the 2D line features are extracted by a region growing method. For 3D plane extraction, the seed region is often grown according to the angle between the normal vectors of the current seed point and its neighboring point [36]. However, we find that the angle between the normal vectors is not suitable for the 2D line extraction, because the points (e.g., the points in the red circle in Figure 1c) that are close to a line will be segmented to the line. This will degrade the accuracy of line fitting. In [37], the distance from the neighboring point to the line is used to grow the seed region for 3D plane extraction. The distance indicator is suitable for 2D line extraction, and is therefore used in our method. Certainly, the distance here denotes the distance from the neighboring point to the line. We extend the point-based region growing method in [37] to extract the 2D line features and make some improvements. First, we calculate the fitting residual of the local neighborhood of each point. This is accomplished by performing singular value decomposition on the local neighborhood. The fitting residual is defined as the ratio between the small singular value and the big one. The point \( p \) is selected, and the local neighborhood \( N(p) \) of this point is treated as the initial seed region \( SG \). The points in the neighborhood are also removed from the point cloud. The seed region \( SG \) is iteratively extended by adding its neighbors. If the distance from a neighbor \( p_k \) to the line fitted by \( SG \cup p_k \) is smaller than a threshold \( \delta \) (0.5\( pr \) in this paper), the neighbor is added into the seed region and removed from the point cloud. The line is also fitted by singular value decomposition. In this iteration process, we use the newly added points \( AP \) rather than the whole seed region to search the neighbors \( N(AP) \) in order to increase the computational efficiency. The initial added point set \( AP \) is set as the local neighborhood \( N(p_{\text{min}}) \). The process is stopped until no point can be added into the seed region. The obtained line \( I_k \) at the last iteration is preserved in \( L \). Then, the next point with minimum fitting residual is selected. In [37,38], the algorithms are ended when the point cloud \( P \) is empty. By contrast, we end the algorithm when the minimum fitting residual of the currently selected point is bigger than a threshold \( \epsilon \) (0.01 in this paper). This is because the points with big fitting residual are those cluttered points, which are not on the lines. The 2D line extraction is shown in Algorithm 1 (see Appendix A) in detail. Note that in this algorithm, the input point cloud \( P \) is the simplified 2D point cloud.

3. Point Cloud Registration with 2D Line Features

After the 2D line extraction, we need to identify the line correspondences. Then the 2D transformation is calculated according to the line correspondences. Afterward, we need to calculate the displacement along the z-axis. Finally, the 3D transformation is obtained. The outline of our registration method is shown in Figure 2.
3.1. 2D Transformation Calculation

Assuming that we have two point clouds: source point cloud \( P_s \) and target point cloud \( P_t \), the source point cloud will be aligned into the target point cloud. The two line sets extracted from the two point clouds are denoted as \( L^s = \{ l^s_1, l^s_2, \ldots, l^s_{m_s} \} \) and \( L^t = \{ l^t_1, l^t_2, \ldots, l^t_{m_t} \} \). Because the calculated normals of the lines are ambiguous, we add the opposite lines into \( L^t \). That is the line set becomes \( L^t = \{ l^t_1, l^t_2, \ldots, l^t_{m_t}, -l^t_1, -l^t_2, \ldots, -l^t_{m_t} \} \). Then we need to find the corresponding lines between \( L^s \) and \( L^t \). Random sample consensus (RANSAC) [39] is the most frequently used method. However, the number of iterations is very difficult to determine. Considering that the number of the extracted 2D lines is generally small, we choose to traverse all the candidate line correspondences. Certainly, if the number of the extracted 2D lines is big, a RANSAC-like algorithm can also be applied. The cosine value of the normals of two lines is used to search the candidate line correspondences. In order to avoid repeatedly calculating the cosine values during the traverse process, we calculate the cosine values of two arbitrary lines for the two line sets beforehand. Arbitrary two lines in \( L^s \) can form a pair \( c^s = \{ l^s_i, l^s_j \} \). If the absolute value of the cosine value of \( c^s \)

\[
\text{abs}(\cos < l^s_i, l^s_j >) < \cos(10^6)
\]

the pair \( c^s \) and its cosine value are preserved. The aim is to eliminate the pairs that are formed by parallel two lines, because the parallel two lines cannot be used to compute the 2D transformation. The same operation is also used for \( L^t \). The preserved pairs and their cosine values for the line set \( L^s \) are denoted as \( C^s = \{ c^s_1, c^s_2, \ldots, c^s_{m_s} \} \) and \( \text{COS}^s = \{ \cos^s_1, \cos^s_2, \ldots, \cos^s_{m_s} \} \). Similarly, the preserved pairs and their cosine values for the line set \( L^t \) are denoted as \( C^t = \{ c^t_1, c^t_2, \ldots, c^t_{m_t} \} \) and \( \text{COS}^t = \{ \cos^t_1, \cos^t_2, \ldots, \cos^t_{m_t} \} \). We traverse all the pairs in \( C^s \) and \( C^t \) to identify the correct line correspondences. For each pair in \( C^s \), its corresponding pair is searched in \( C^t \). If

\[
\text{abs}(\cos^s_i - \cos^t_j) < \lambda
\] (2)

the pair \( c^t_j \) is considered as the possible match of \( c^s_i \). The threshold \( \lambda \) is experimentally set as 0.2pr. The two pairs \( c^s_i = \{ l^s_i, l^s_j \} \) and \( c^t_j = \{ l^t_p, l^t_q \} \) are used to calculate the 2D transformation. There are two matching conditions between \( c^s_i \) and \( c^t_j \). The line correspondences can be \( \{ l^s_i, l^t_p \} \) and \( \{ l^s_j, l^t_q \} \), but they can also be \( \{ l^s_i, l^t_q \} \) and \( \{ l^s_j, l^t_p \} \). Therefore, we calculate two 2D transformations.

Figure 2. The outline of our registration method.
The transformation that generates the big overlap between \( L^t \) and \( L^i \) is preserved. The overlap is defined as

\[
\text{overlap} = \frac{\text{the number of correct line correspondences}}{\min(n_1, n_2)}
\]  

(3)

Note that when we generate the pair set \( C^s \), the \( \{ \ell^t_i, \ell^t_j \} \) is a pair, but \( \{ \ell^i_p, \ell^i_q \} \) does not exist in the pair set because this will increase the number of the pairs and finally increase the computation time of the traverse process. For the pair set \( C^d \), \( \{ \ell^d_p, \ell^d_q \} \) is a pair, but \( \{ \ell^t_p, \ell^t_q \} \) does not exist in the pair set as well. Taking \( \{ \ell^t_i, \ell^t_j \} \) and \( \{ \ell^i_p, \ell^i_q \} \) as an example, we can formulate

\[
r \begin{bmatrix} n^d_i & n^d_j \end{bmatrix} = \begin{bmatrix} n^t_i & n^t_j \end{bmatrix} 
\]

(4)

\[
\begin{bmatrix} n^t_p & n^t_q \end{bmatrix}^T t = \begin{bmatrix} c^p_i - c^p_j \\ c^q_j - c^q_i \end{bmatrix}
\]

(5)

where \( \begin{bmatrix} n^d_i & n^d_j \end{bmatrix} \) and \( \begin{bmatrix} n^d_p & n^d_q \end{bmatrix} \) are the line parameters of \( \ell^d_i, \ell^d_j, \ell^d_p \) and \( \ell^d_q \), respectively. Performing singular value decomposition

\[
\begin{bmatrix} U & S & V \end{bmatrix} = \text{svd}(\begin{bmatrix} n^d_i & n^d_j \end{bmatrix} \begin{bmatrix} n^d_p & n^d_q \end{bmatrix}^T)
\]

(6)

we can represent the 2D rotation matrix as

\[
r = VU^T
\]

(7)

The 2D translation vector is calculated as

\[
t = (\begin{bmatrix} n^t_p & n^t_q \end{bmatrix}^T)^{-1} \begin{bmatrix} c^p_i - c^p_j \\ c^q_j - c^q_i \end{bmatrix}
\]

(8)

After all the pairs in \( C^s \) and \( C^d \) are traversed and the 2D transformations are calculated, the 2D transformation with the biggest overlap between \( L^s \) and \( L^i \) is output. The calculation of the 2D transformation is detailly illustrated in Algorithm 2 (see Appendix B).

### 3.2. The Calculation of the Displacement along the Z-axis

With the 2D transformation, the 3D transformation can be obtained as

\[
TT = \begin{bmatrix} R & T \end{bmatrix}
\]

(9)

where \( R = \begin{bmatrix} r & 0_{2 \times 1} \\ \theta_{1 \times 2} & 1 \end{bmatrix} \) is the 3D rotation matrix and \( T = \begin{bmatrix} t \\ t_2 \end{bmatrix} \) is the 3D translation vector. Hence, our next task is to calculate the displacement \( t_2 \) along the z-axis.

Assuming that the two simplified 2D point clouds from \( P^s \) and \( P^i \) are \( P^s_{\text{Line-s}} \) and \( P^i_{\text{Line-s}} \), \( P^s_{\text{Line-s}} \) is first transformed by using the 2D transformation

\[
\overrightarrow{P^s_{\text{Line-s}}} = r P^s_{\text{Line-s}} + t
\]

(10)

Then, the points in the overlap area are extracted from \( \overrightarrow{P^s_{\text{Line-s}}} \). For a point \( \overrightarrow{P^s_{\text{Line-s}}} \) in the overlap area, we find its corresponding point \( P^s_{\text{Line-s}} \) in \( P^s_{\text{Line-s}} \). Then, a vertical cylindrical neighborhood around the point \( P^s_{\text{Line-s}} \) is used to extract the neighboring points \( N(P^s_{\text{Line-s}}) \) from the source point cloud \( P^s \), as shown in Figure 3a. Similarly, a vertical cylindrical neighborhood around \( \overrightarrow{P^s_{\text{Line-s}}} \) is also used to
extract the neighboring points \( N(P_{\text{Line-s}}^t) \) from the target point cloud \( p^t \), as shown in Figure 3b. Thus, we can use the minimum z-values in the two cylindrical neighborhoods to calculate the displacement

\[
l_z = z_{\min}^t - z_{\min}^s
\]  

where \( z_{\min}^t \) is the minimum z-value of \( N(P_{\text{Line-s}}^t) \), and \( z_{\min}^s \) is the minimum z-value of \( N(p_{\text{Line-s}}^s) \). As long as the lower parts of the two cylindrical neighborhoods are not missed, the displacement can be correctly computed. In most cases, this can be guaranteed when the point \( P_{\text{Line-s}}^t \) is one in the overlap area. In order to get the reliable displacement, we randomly select 50 points from the overlap area, and calculate 50 displacements. Then, a clustering method is performed on these displacements. The displacements with similar size will be clustered together. The false displacements are less likely to cluster together, meaning that a displacement represents a cluster. The cluster with the maximum number of the displacements is used to get the final displacement, which is the mean value of the displacements in the maximal cluster. Eventually, the 3D transformation is obtained by Equation (9). Because the lower parts of two cylindrical neighborhoods are applied to calculate the displacement, our method is not suitable for the aerial point clouds, for which the lower parts are always missed.

![Figure 3](image_url)

_Figure 3._ The illustration of the extraction of the points in the cylindrical neighborhood. The blue points are the extracted points by using the cylindrical neighborhood. (a) The simplified 2D point cloud \( P_{\text{Line-s}}^t \) and the source point cloud \( p^t \), (b) the simplified 2D point cloud \( P_{\text{Line-s}}^s \) and the target point cloud \( p^s \).

### 4. Experiments and Results

In this section, the experiments will be performed on an indoor dataset and an outdoor dataset. In order to save time, we first perform the point cloud simplification. A point-based method and a plane-based method are implemented for comparison. In the point-based method, the intrinsic shape signature (ISS) detector [40] is used to extract the keypoints, and the QLCI descriptors [14] are calculated on the keypoints. The nearest neighbor similarity ratio (NNSR) [41] is applied to establish the point-to-point correspondences. The 1-point RANSAC algorithm [41], which is an improved version of RANSAC, is applied to find the correct correspondences and calculate the 3D transformation. For the plane-based method, a similar pipeline to our method is applied for a fair comparison. The 3D planes are extracted by the region growing method in [38], then the cosine values of the normals of three planes are calculated to find the candidate plane correspondences and the triplets with parallel planes are eliminated. All the triplets are traversed to identify the correct plane correspondences and
compute the 3D transformation. All the experiments are implemented in Matlab by a PC with an Intel Core i7-7500U 2.7 GHz CPU and 20 GB of RAM.

4.1. Evaluation Criterion of Registration Accuracy

Three metrics are used to evaluate the registration accuracy. The rotation error is measured by

\[ e_R = a \cos \left( \frac{\text{trace}(R_{\text{true}} (R)^{-1}) - 1}{2} \right) \frac{180}{\pi} \]  

(12)

where \( R \) is the calculated rotation matrix, and \( R_{\text{true}} \) is the true rotation matrix. Considering that the displacement along z-axis is separately computed in our method, we divide the translation error into horizontal error and vertical error, which are defined as

\[ e_{T_h} = ||t_{\text{true}} - t|| \]  

(13)

\[ e_{T_v} = ||t_{z-\text{true}} - t_z|| \]  

(14)

where \( t \) is the calculated 2D translation vector, \( t_{\text{true}} \) is the true 2D translation vector, \( t_z \) is the calculated displacement along z-axis, and \( t_{z-\text{true}} \) is the true displacement along z-axis. Here, the true values \( R_{\text{true}}, t_{\text{true}} \), and \( t_{z-\text{true}} \) are obtained by the fine registration, which is performed by the ICP algorithm [10].

4.2. Registration Results on the Indoor Dataset

The indoor dataset [42] is acquired by a FARO Focus 3D X330 HDR scanner. The publishers scanned five scenes: Apartment, Bedroom, Boardroom, Lobby and Loft. For each scene, multiple point clouds were obtained from different views. We choose the two point clouds of Apartment and two point clouds of Boardroom to perform the experiments. The registration results obtained by our method are shown in Figures 4 and 5. In order to have a better visual experience, we remove the points on the ceiling after two point clouds are registered and exhibit the point clouds without the points on the ceiling.

As we can see from Figures 4 and 5, the two point clouds of both Apartment and Boardroom are automatically aligned. There is no significant inconsistency between the two registered point clouds. The simplified 2D point clouds are quite well aligned together. From the indoor furniture (e.g., sofa and chair in Figure 4d, and the council board and meeting chair in Figure 5), we can further see that the registration results are rather good.

The registration accuracy of the three methods is shown in Table 1. In order to save space, Apartment is abbreviated as Ap, and Boardroom is abbreviated as Bo. We can see that all the three methods can provide enough good initial pose for the fine registration. Our method obtains the best registration accuracy. The point-based method is the poorest method. This is because the point-based method is sensitive to noise and point density variation. The 3D plane and 2D line belong to the higher-level geometric feature, so the plane-based and 2D line-based methods can achieve significantly more accurate registration results. The plane-based method shows inferior registration accuracy compared to our method. This is because the region growing method [38] is not robust to outliers, meaning that the fitting accuracy of plane is low. In contrast, the distance indicator is applied in our 2D line extraction method. The outliers can be removed automatically. For the vertical error, our method also gets the smallest error. This indicates that the proposed method for calculating the displacement along the z-axis is feasible.
where $\mathbf{R}$ is the calculated rotation matrix, and $\mathbf{R}_{\text{true}}$ is the true rotation matrix. Considering that the displacement along z-axis is separately computed in our method, we divide the translation error into horizontal error and vertical error, which are defined as

$$
\mathbf{t}_{\text{calc}} = \|\mathbf{R}_{\text{true}} - \mathbf{R}_{\text{true}}\| \quad (13)
$$

$$
\mathbf{t}_{\text{calc}} = \|\mathbf{R}_{\text{true}} - \mathbf{R}_{\text{true}}\| \quad (14)
$$

where $\mathbf{t}_{\text{calc}}$ is the calculated 2D translation vector, $\mathbf{t}_{\text{true}}$ is the true 2D translation vector, $\mathbf{t}_{\text{calc}}$ is the calculated displacement along z-axis, and $\mathbf{t}_{\text{true}}$ is the true displacement along z-axis. Here, the true values $\mathbf{R}_{\text{true}}, \mathbf{t}_{\text{true}}$ and $\mathbf{t}_{\text{true}}$ are obtained by the fine registration, which is performed by the ICP algorithm [10].

### 4.2. Registration results on the indoor dataset

The indoor dataset [42] is acquired by a FARO Focus 3D X330 HDR scanner. The publishers scanned five scenes: Apartment, Bedroom, Boardroom, Lobby and Loft. For each scene, multiple point clouds were obtained from different views. We choose the two point clouds of Apartment and two point clouds of Boardroom to perform the experiments. The registration results obtained by our method are shown in Figure 4 and Figure 5. In order to have a better visual experience, we remove the points on the ceiling after two point clouds are registered and exhibit the point clouds without the points on the ceiling.

**Figure 4.** The coarse registration result of Apartment. (a) The source and target point clouds. (b) The registration result of the simplified 2D point clouds. (c) The registration result of the 3D point clouds. (d) The registered point clouds after removing the points on the ceiling for a better view.

The computation time of the three methods is listed in the Table 2. In the Apartment scene, the number of points in the source and target point clouds is 124,426 and 185,436, respectively. In the Boardroom scene, the number of points in the two point clouds is 101,276 and 138,880, respectively. We also give the numbers of the features of the two point clouds being matched for better comparison. The feature in the three methods respectively denotes point feature (i.e., keypoint), plane feature and 2D line feature. Because the accuracy of the coarse registration has a great influence on the iteration number of the fine registration, we also give the computation time for the fine registration. For the plane-based method, we also set a threshold of the minimum fitting residual to stop the feature extraction. Actually, the numbers of the extracted plane features are 65 vs. 77 for Apartment and 87 vs. 108 for boardroom. In this experiment, we only use the large planes to calculate the 3D transformation. Nevertheless, we can see from Table 2 that the computation time for the feature extraction in the plane-based method is much more than that in our method. The plane-based method extracts the 3D planes and processes the point clouds in 3D space. As consequence, the computation cost is very high. By contrast, in our method, only the points with high density are extracted and further simplified to reduce the number of the points. Finally, the 2D lines are extracted in 2D space. Thus, our feature extraction is computationally very efficient. The time cost of the coarse registration in the plane-based method is also significantly more than that in our method, though the plane-based method has less features and our method has an additional step to calculate the displacement along the z-axis. This is because the number of the formed triplets in the plane-based method is much more than the number of the
formed pairs in our method. The point-based method is also more time-consuming than our method, because the number of the keypoints is big. Processing thousands of keypoints needs a great deal of computation cost. The 1-point RANSAC algorithm uses the local reference frames of keypoints to calculate the rotation, so only one point is enough to calculate the 3D transformation. This largely reduces the required iteration number of the RANSAC algorithm. In this experiment, the iteration number is set as 200, but the point-based method still spends more time on the coarse registration than our method.

Figure 4. The coarse registration result of Apartment. (a) The source and target point clouds. (b) The registration result of the simplified 2D point clouds. (c) The registration result of the 3D point clouds. (d) The registered point clouds after removing the points on the ceiling for a better view.

As we can see from Figure 4 and Figure 5, the two point clouds of both Apartment and Boardroom are automatically aligned. There is no significant inconsistency between the two registered point clouds. The simplified 2D point clouds are quite well aligned together. From the indoor furniture (e.g., sofa and chair in Figure 4d, and the council board and meeting chair in Figure 5), we can further see that the registration results are rather good.

The registration accuracy of the three methods is shown in Table 1. In order to save space, Apartment is abbreviated as Ap, and Boardroom is abbreviated as Bo. We can see that all the three methods can provide enough good initial pose for the fine registration. Our method obtains the best registration accuracy. The point-based method is the poorest method. This is because the point-based method is sensitive to noise and point density variation. The plane-based and 2D line-based methods can achieve significantly more accurate registration results. The plane-based method shows inferior registration accuracy compared to our method. This is because the region growing method [38] is not robust to outliers, meaning that the fitting accuracy of plane is low. In contrast, the distance indicator is applied in our 2D line extraction method. The outliers can be removed automatically. For the vertical error, our method also gets the smallest error. This indicates that the proposed method for calculating the displacement along the z-axis is feasible.

Figure 5. The coarse registration result of Boardroom. (a) The source and target point clouds. (b) The registration result of the simplified 2D point clouds. (c) The registration result of the 3D point clouds. (d) The registered point clouds after removing the points on the ceiling for a better view.

Table 1. The registration accuracy of the three methods on the indoor dataset. The best results are denoted in bold font.

<table>
<thead>
<tr>
<th>Method</th>
<th>Point-Based</th>
<th>Plane-Based</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ap</td>
<td>Bo</td>
<td>Ap</td>
</tr>
<tr>
<td>Rotation error</td>
<td>0.9756</td>
<td>0.2798</td>
<td>0.1454</td>
</tr>
<tr>
<td>Horizontal error</td>
<td>0.3933</td>
<td>0.3363</td>
<td>0.0360</td>
</tr>
<tr>
<td>Vertical error</td>
<td>0.0367</td>
<td>0.0244</td>
<td>0.0101</td>
</tr>
</tbody>
</table>
Table 2. The number of features and the time spent on each phase for the three methods on the indoor dataset. $T_{E}$ denotes the time for feature extraction, $T_{C}$ denotes the time for coarse registration, $T_{F}$ denotes the time for fine registration, and $T_T$ denotes the total time.

<table>
<thead>
<tr>
<th>Method</th>
<th>Point-Based</th>
<th>Plane-Based</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ap</td>
<td>Bo</td>
<td>Ap</td>
</tr>
<tr>
<td>Number of features</td>
<td>2892 vs. 4255</td>
<td>2073 vs. 2878</td>
<td>20 vs. 15</td>
</tr>
<tr>
<td>$T_{E}$ (s)</td>
<td>190.6316</td>
<td>56.7230</td>
<td>1239.3671</td>
</tr>
<tr>
<td>$T_{C}$ (s)</td>
<td>527.4623</td>
<td>374.7924</td>
<td>184.3647</td>
</tr>
<tr>
<td>$T_{F}$ (s)</td>
<td>142.7769</td>
<td>70.0760</td>
<td>31.7702</td>
</tr>
<tr>
<td>$T_T$ (s)</td>
<td>860.8708</td>
<td>501.5914</td>
<td>1455.5020</td>
</tr>
</tbody>
</table>

4.3. Registration Results on the Outdoor Dataset

The outdoor dataset [43] is acquired by a Leica C10 laser scanner. The publishers provide the point clouds of the City and Castle scenes. We choose two point clouds from each scene to perform the experiments. The experimental results obtained by our method are shown in Figures 6 and 7. We also amplify the local point clouds in order to see the registration results more clearly.

Figure 6. The coarse registration result of City. (a) The source and target point clouds. (b) The registration result of the simplified 2D point clouds. (c) The registration result of the 3D point clouds. (d) The locally amplified point clouds for a better view.

For the outdoor scenes, our method is still feasible. The simplified 2D point clouds are aligned quite well. We can also see from the registration results of the 3D point clouds (Figures 6c and 7c) and
the locally amplified point clouds (Figures 6d and 7d) that the point clouds of both the two scenes are well registered.

The registration accuracy tested on the outdoor dataset is listed in Table 3. Our method obtains the highest accuracy, followed by the plane-based method and point-based method. Due to the sensitivity to noise and point density variation, the point-based method achieves the poorest accuracy. In the plane-based method, the applied plane extraction method is not able to resist outliers, so the method also exhibits a poorer registration accuracy than our method. In comparison with the other two methods, the vertical error of our method is rather small as well. This means that our method can get the precise displacement along the z-axis.

Table 3. The registration accuracy of the three methods on the outdoor dataset. The best results are denoted in bold font.

<table>
<thead>
<tr>
<th>Method</th>
<th>Point-based</th>
<th>Plane-based</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>Rotation error</td>
<td>2.0206</td>
<td>0.7600</td>
</tr>
<tr>
<td>Castle</td>
<td>Horizontal error</td>
<td>0.3091</td>
<td>0.2877</td>
</tr>
<tr>
<td>City</td>
<td>Vertical error</td>
<td>8.2586</td>
<td>1.8412</td>
</tr>
<tr>
<td>Castle</td>
<td>Vertical error</td>
<td>1.2927</td>
<td>0.5308</td>
</tr>
</tbody>
</table>

Figure 7. The coarse registration result of Castle. (a) The source and target point clouds. (b) The registration result of the simplified 2D point clouds. (c) The registration result of the 3D point clouds. (d) The locally amplified point clouds for a better view.

The time efficiency tested on the outdoor dataset is listed in Table 4. In the City scene, the number of points in the source and target point clouds is 193,219 and 177,535, respectively. In the Castle scene, the number of points in the two point clouds is 455,367 and 426,357, respectively. The exact numbers of the extracted planes in the plane-based method are 160 vs. 172 for City and 158 vs. 157 for Castle. Only the large planes are used to perform the point cloud registration. As we can see, our method achieves the best computation efficiency. The plane-based method is the most time-consuming, because it processes the point clouds in 3D space. The point-based method is also computationally expensive due to the large number of the keypoints. Specially, for the Castle scene, the number of the extracted
keypoints is very large. This is because the ISS detector prefers to extract the points with abundant geometric information (i.e., non-planar points) as the keypoints, and there is a large amount of such points (e.g., the points of the trees) in the scene.

<table>
<thead>
<tr>
<th>Method</th>
<th>Point-Based</th>
<th>Plane-Based</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>City</td>
<td>Castle</td>
<td>City</td>
</tr>
<tr>
<td>Rotation error</td>
<td>2.0206</td>
<td>8.2586</td>
<td>1.8412</td>
</tr>
<tr>
<td>Horizontal error</td>
<td>0.3091</td>
<td>1.2927</td>
<td>0.5308</td>
</tr>
<tr>
<td>Vertical error</td>
<td>0.6536</td>
<td>2.6860</td>
<td>0.0443</td>
</tr>
</tbody>
</table>

Table 3. The registration accuracy of the three methods on the outdoor dataset. The best results are denoted in bold font.

Table 4. The number of features and the time spent on each phase for the three methods on the outdoor dataset. \( T_E \) denotes the time for feature extraction, \( T_C \) denotes the time for coarse registration, \( T_F \) denotes the time for fine registration, and \( T_T \) denotes the total time.

<table>
<thead>
<tr>
<th>Method</th>
<th>Point-based</th>
<th>Plane-based</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>City</td>
<td>Castle</td>
<td>City</td>
</tr>
<tr>
<td>Number of features</td>
<td>2627 vs. 2632</td>
<td>14,778 vs. 11,482</td>
<td>29 vs. 42</td>
</tr>
<tr>
<td>( T_E ) (s)</td>
<td>195.9942</td>
<td>1260.3</td>
<td>1958.3</td>
</tr>
<tr>
<td>( T_C ) (s)</td>
<td>558.7629</td>
<td>4244.7</td>
<td>1696.0</td>
</tr>
<tr>
<td>( T_F ) (s)</td>
<td>105.3999</td>
<td>2601.5</td>
<td>77.4270</td>
</tr>
<tr>
<td>( T_T ) (s)</td>
<td>860.1570</td>
<td>8106.5</td>
<td>3731.7</td>
</tr>
</tbody>
</table>

5. Conclusions

In this paper, we presented a registration method based on the 2D line features. Firstly, a 2D line extraction algorithm was proposed. Then, the simplified 2D point clouds were registered by establishing correspondences between the 2D lines from the two point clouds being matched, and the 2D transformation was calculated. Finally, a method was formulated to compute the displacement along the z-axis, and the 3D transformation was obtained by integrating the 2D transformation and displacement.

The experiments have been performed on an indoor dataset and an outdoor dataset to validate the performance of our method. A point-based method and a plane-based method are compared with our method. The experimental results show that our method can obtain a relatively good registration result. In terms of the registration accuracy, our method is the best on both of the two datasets. The rotation error is less than 0.55°. The horizontal error is less than 0.25m and the vertical error is less than 0.015m. The more significant advantage is that our method has a very good computation efficiency. On the point clouds of the Castle scene, the total computation time is the largest, which is less than 12 minutes.

Author Contributions: W.T. designed the method and wrote this paper. X.H. performed the experiments and analyzed the experiment results. Z.C. and P.T. checked and revised this paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 41674005; 41374011; 41904031, the China Scholarship Council (CSC) Scholarship, grant number 201906270179, the Open Foundation of Key Laboratory for Digital Land and Resources of Jiangxi Province, grant number DLLJ201905, and the PhD early development program of East China University of Technology, grant number DHBK2018006, and the APC was funded by Xianghong Hua.

Acknowledgments: The authors want to thank the ETH Computer Vision and Geometry group for making their dataset available online.

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

Algorithm 1: 2D line extraction

**Input:** Point cloud $P$

**Output:** the parameters of all extracted lines $L$

1: Calculate the fitting residual of each point.
2: $L = \emptyset$
3: for $i = 1$ to the preset maximal number of iterations do
4: find the point with the minimum fitting residual $p_{\text{min}}$.
5: the initial seed region $SG = N(p_{\text{min}})$.
6: the initial added points $AP = N(p_{\text{min}})$
7: remove the $N(p_{\text{min}})$ from the point cloud $P$.
8: for $j = 1$ to the preset maximal number of iterations do
9: find the neighbors $N(AP)$ of $AP$.
10: for $k = 1$ to size($N(AP)$) do
11: $p_k$ is the $k$th point in $N(AP)$.
12: perform line fitting on $SG \cup p_k$ by singular value decomposition.
13: if the distance from $p_k$ to the fitted line $l_k$ is smaller than $\delta$ then
14: $SG = SG \cup p_k$.
15: remove $p_k$ from the point cloud $P$.
16: end if
17: end for $k$
18: if no point can be added into $SG$ then
19: $AP$ is updated as the newly added points in $SG$.
20: end if
21: $L = L \cup l_k$
22: if the minimum fitting residual is bigger than $\epsilon$ then
23: break;
24: end if
25: end for $i$

Appendix B

Algorithm 2: 2D transformation calculation

**Input:** the line sets $L_s$ and $L_t$, the pair sets $C_s$ and $C_t$, and the cosine values $\cos s$ and $\cos t$.

**Output:** the 2D transformation $tt$.

1: Set the initial overlap: $\text{overlap}(00) = 0$.
2: for $i = 1$ to $m_1$ do
3: for $j = 1$ to $m_2$ do
4: if $\text{abs}(\cos^s - \cos^t) < \lambda$ then
5: Use the two pairs $c^s_i$ and $c^t_j$ to compute two 2D transformations.
6: Calculate the two overlaps between $L^s$ and $L^t$ corresponding to the two 2D transformations.
7: Preserve the 2D transformation $\begin{bmatrix} r & t \end{bmatrix}$ with the big overlap $\text{overlap}_{ij}$.
8: if $\text{overlap}_{ij} > \text{overlap}_{(00)}$ then
9: The 2D transformation $tt = \begin{bmatrix} r & t \end{bmatrix}$.
10: $\text{overlap}_{(00)} = \text{overlap}_{ij}$
11: end if
12: end if
13: end for $j$
14: end for $i$
References

12. Li, W.; Song, P. A modified ICP algorithm based on dynamic adjustment factor for registration of point cloud and CAD model. *Pattern Recognit. Lett.* 2015, 65, 88–94. [CrossRef]


© 2020 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 (http://creativecommons.org/licenses/by/4.0/).