A Fast Registration Method for Building Point Clouds Obtained by Terrestrial Laser Scanner via 2-D Feature Points

Wuyong Tao, Yansheng Xiao, Ruisheng Wang, *Senior Member, IEEE*, Tieding Lu, and Shaoping Xu

***Abstract*—Point cloud registration plays a central role in various applications, such as 3-D scene reconstruction, preservation of cultural heritage and deformation monitoring. The point cloud data are usually huge. Processing such huge data is very time-consuming, so a fast and accurate registration method is crucial. However, the existing registration methods still have high computation complexity or low accuracy. To address this issue, we develop a registration method for terrestrial point clouds. The method projects the point clouds onto the horizontal plane. Therefore, our method processes point cloud data in 2-D space, leading to high computation efficiency. Then, the 2-D feature lines are extracted from the projected point clouds. We calculate the intersection points of the 2-D feature lines, which are treated as the 2-D feature points. Due to the high accuracy of the 2-D feature lines, the 2-D feature points also have high accuracy. Thus, our method can get accurate registration results. Afterward, the feature triangles are constructed by using the 2-D feature points, and the geometric constraints are utilized to find the corresponding feature triangles for calculating the 2-D transformation. This strategy boosts the process of searching for the corresponding 2-D feature points. Subsequently, the *Z*-axis displacement is computed by the cylindrical neighborhoods. By combining the *Z*-axis displacement and 2-D transformation, the 3-D rigid transformation is obtained. Experimental evaluation conducted on two publicly available datasets well demonstrates that the proposed registration method can achieve good computational efficiency and high accuracy.**

***Index Terms*—Point cloud registration, congruent feature triangle, 2-D feature point, 2-D transformation.**

Manuscript received 7 February 2024; revised 9 April 2024; accepted 21 April 2024. Date of publication 24 April 2024; date of current version 6 May 2024. This work was supported in part by the Open Fund of Key Laboratory of Mine Environmental Monitoring and Improving around Poyang Lake of Ministry of Natural Resources under Grant MEMI-2023-06, and in part by the National Natural Science Foundation of China under Grant 42271452, Grant 42374040, Grant 42061077, Grant 42104023, and Grant 62162043. *(Corresponding author: Wuyong Tao.)*

Wuyong Tao is with the School of Mathematics and Computer Sciences, Nanchang University, Nanchang 330031, China, and also with the Key Laboratory of Mine Environmental Monitoring and Improving around Poyang Lake of Ministry of Natural Resources, East China University of Technology, Nanchang 330013, China (e-mail: taowuyong@ncu.edu.cn).

Yansheng Xiao and Shaoping Xu are with the School of Mathematics and Computer Sciences, Nanchang University, Nanchang 330031, China (e-mail: 6109120084@email.ncu.edu.cn; xushaoping@ncu.edu.cn).

Ruisheng Wang is with the School of Architecture & Urban Planning, Shenzhen University, Shenzhen 518060, China, and also with the Department of Geomatics Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: ruiswang@ucalgary.ca).

Tieding Lu is with the Laboratory of Mine Environmental Monitoring and Improving around Poyang Lake of Ministry of Natural Resources, East China University of Technology, Nanchang 330013, China (e-mail: tdlu@ecut.edu.cn).

The code will be available at https://github.com/taowuyong?tab=repositories after publication.

Digital Object Identifier 10.1109/JSTARS.2024.3392927

I. Introduction

LIGHT detection and ranging (lidar) is a type of remote sensing technology [1]. Terrestrial laser scanning is the ground-based lidar, which is capable of accurately capturing surface information of real scenes, i.e., point cloud. It has been extensively employed in various applications, including 3-D scene reconstruction [2], [3], natural resource assessment [4], [5], change detection [6], [7], cultural heritage preservation [8], [9], and so on. However, during the process of point cloud data acquisition, a point cloud from a single scan is unable to represent complete scene due to occlusion and limited field of view. Hence, to construct a complete 3-D scene, point cloud registration technique is required to transform data from different stations into a unified coordinate frame. This comprises two stages: coarse registration and fine registration. Among the fine registration algorithms, the most renowned one is the iterative closest point (ICP) algorithm [10]. Yet, the ICP algorithm, influenced by initial pose, tends to converge to local optima rather than global one. Although the variants of the ICP algorithm [11], [12] attempt to mitigate this issue by exploring larger transformation spaces or refining correspondence relationships, these algorithms cannot guarantee consistent superior performance. In order to deal with the shortcoming of the ICP-like algorithms, most researchers first obtain a favorable initial pose via coarse registration. Hence, this article primarily investigates the coarse registration.

Presently, point cloud registration methods can be generally grouped into three categories: point-based, plane/line-based, and specific object-based methods. The point-based methods primarily include the local shape descriptor (LSD)-based methods and the four-point congruent set (4PCS)-like methods [13], [14]. For the LSD-based methods, the keypoints are first extracted from the point clouds being matched. Subsequently, the LSDs of these keypoints are computed. The similarity of the LSDs is compared to establish the point-to-point correspondences. Numerous LSDs have been proposed to date, including rotational projection statistics (RoPS) [15], local voxelized structure (LoVS) [16], triplet local coordinate images (TriLCI) [17], and so forth. Recently, some learning-based descriptors are also proposed, such as 3DMatch [18], FCGC [19], and SpinNet [20].

However, the establishment of the correspondences is plagued by the noise, variation in point density and partial overlap. Even with excellent LSD, there are still many erroneous correspondences that are established, leading to significant registration error. The 4PCS algorithm utilizes the rules of intersection rate to identify corresponding four-point base. As this algorithm relies on brute-force search to find corresponding points, it can be extremely time-consuming. For the reason, numerous scholars have dedicated to enhance the computational efficiency of the algorithm. Some improved algorithms have been proposed, such as super 4PCS [14], keypoint-based 4PCS (K-4PCS) [21], generalized 4PCS (G4PCS) [22], and semantic K-4PCS [23]. These algorithms employ different strategies to improve the computational efficiency of registration, but they still suffer from low computational efficiency.

The plane/line-based methods [24], [25], [26], [27] typically offer superior registration accuracy compared to the point-based methods because the line or plane features are higher level attributes than the point features. The line or plane features are fitted by using many points, so they have higher accuracy than the point features. This kind of methods is particularly well-suited for the registration of building point clouds, which often contain a wealth of feature lines and planes. These methods extract the feature lines or planes from the point clouds and then employ some constraints to identify corresponding lines or planes, i.e., establish line-to-line or plane-to-plane correspondences, to compute the transformation. Most of these methods extract 3-D feature lines or planes and therefore process point cloud data in 3-D space. As a result, they need to handle a large amount of 3-D points, making the registration algorithms time-consuming. Tao et al. [28] proposed using 2-D feature lines to perform point cloud registration. The method has high computation efficiency, but it is hard to design a strategy to quickly find the corresponding 2-D feature lines. The specific object-based methods focus on specific scene, which employ specific objects (e.g., lampposts [29] and crest lines [9]) presented in the scene to facilitate point cloud registration. Hence, these methods lack universality and can only be applied for the specific scenes.

Based on the aforementioned analysis, the point-based methods generally exhibit low accuracy. The plane/line-based methods often have high computation complexity, and specific object-based methods have limited application scenes. Therefore, a fast and accurate registration method is developed in this article. The proposed method is an improvement to our previous work [28], in which a registration method based on 2-D feature lines is proposed. The method takes the building point clouds captured by terrestrial laser scanner into consideration. The terrestrial laser scanner is usually leveled when working, so the Z-axis of the scanned point clouds is parallel to the direction of gravity, i.e., the point clouds are upright. This means there is no rotation around the X and Y axes. Under this case, the four degree of freedom rather than six degree of freedom rigid transformation needs to be estimated. This can largely simplify the registration problem. The upright point clouds are also very common. Besides our method [28], many methods [29], [30], [31] have also been proposed to register the upright point clouds by using the advantageous trait. Although our previous method has obtained good computation efficiency and high registration accuracy, we want to further improve the computation efficiency and preserve the high registration accuracy. For the purpose, the intersection points of the 2-D feature lines are calculated to perform the point cloud registration in this article. These intersection points are treated as the 2-D feature points. Then, we design a strategy to boost the searching process of the corresponding 2-D feature points from two point clouds. First, the 2-D feature points are used to construct feature triangles. Then, based on three geometric constraints, the feature triangles from the two point clouds are quickly matched. The corresponding feature triangles are found. Thus, three pairs of 2-D feature points are obtained, with which the 2-D transformation can be calculated. Finally, the Z-axis displacement is calculated by the same method in Tao et al.'s [28] work. The 3-D transformation can be directly achieved by combining the 2-D transformation and Z-axis displacement. The main contributions of this study include the following.

1) We propose using 2-D feature points to register the point clouds. Thus, our method mainly processes point cloud data in 2-D space, which lowers computation complexity. The 2-D feature lines are accurate, so the 2-D feature points also have high accuracy.
2) A strategy for searching for corresponding 2-D feature points is proposed. The feature points are first used to construct feature triangles. Then, the strategy uses the geometric constraints to accelerate the searching process of the corresponding feature triangles, so it is computationally cheap. Thus, the correspondences between the 2-D feature points of the source point cloud and the 2-D feature points of the target point cloud are identified.
3) Based on the 2-D feature points, a registration method is presented. The 2-D feature points are fast to be extracted and have high accuracy, and a fast strategy is proposed to find the corresponding 2-D feature points. Therefore, our method has good computation efficiency and high registration accuracy.

II. Related Work

A. Point-Based Registration Methods

The point-based methods predominantly utilize the points extracted from the point clouds to implement the registration. This kind of methods is universal and thus can be applied for different scenes, but they generally have relatively low registration accuracy. First, the LSD-based methods are reviewed. Guo et al. [15] calculated the RoPS descriptor for each keypoint. The nearest neighbor similarity ratio (NNSR) was applied to match feature pairs between two point clouds. Subsequently, the rigid transformation was determined by using the one-point random sample consensus (one-point RANSAC) algorithm. Quan et al. [16] encoded the local shape information into the bit string based on point distribution. The LoVS descriptor was proposed, which was a binary descriptor. The transformation was then estimated using the global constrained one point-based sample consensus (GC1SAC) algorithm. Tao et al. [17] utilized the TriLCI descriptor, which encoded local surface information

from five views. The correspondences were also established by using the NNSR method, and the transformation was computed via a median-based robust estimation. Yang et al. [32] employed uniform sampling to extract keypoints and compute the rotational contour signatures descriptor. A voting-based correspondence selection method was developed to calculate the transformation. Du et al. [33] proposed a descriptor named as multi-view depth and contour signatures, which integrated the depth information and 2-D contour cues to facilitate comprehensive multiview and multiattribute description. This made the descriptor include more local surface information. Tao et al. [34] introduced the local occupied voxel comparison (LOVC) descriptor, which was a natural binary descriptor. It encoded the relationship between the voxels. Then, the two-point random sample consensus with constraints (two-point RANSACWC) algorithm was developed to calculate the transformation. With the development of the convolutional neural network (CNN), the learning-based descriptors get significant progress. 3DMatch [18] applies a siamese convolutional network (CN) to learn the descriptor for local patches. FCGF [19] takes the sparse tensor as the input of the CN. DCP [35] employs the dynamic graph CN to learn the local descriptor. However, they are variant to rotation, which is not beneficial for point cloud registration. In order to get rotation-invariant descriptor, SpinNet [20] aligns the local patch with the local reference axis and formulates cylindrical volume. The local feature is extracted by multilayer perceptron and 3-D cylindrical convolutional layers. Gojcic et al. [36] took a rotation-invariant handcrafted feature (i.e., smoothed density value) as the input of CNN. Regarding the 4PCS-like methods, to address the limitations of the 4PCS algorithm, Mellado et al. [14] employed a rasterization method to make quadratic computation complexity become linear computation complexity. The angle constraint was developed to reduce the number of the candidate point pairs in the target point cloud. Mohamad et al. [22] introduced the G4PCS method, which allowed two point pairs falling on two different planes rather than being strictly coplanar. Theiler et al. [21] proposed the K-4PCS method, which leveraged the keypoints as the candidates for identifying corresponding four-point base sets. The computation burden was effectively reduced. Ge et al. [23] incorporated semantic information during the registration process to represent each keypoint, thereby boosting the searching process of the corresponding four-point bases.

B. Plane/Line-Based Registration Methods

The plane/line-based methods extract feature lines or planes from the point clouds for registration. As the feature lines and planes are higher level features than point features, the methods in this category generally have better registration accuracy than the point-based registration methods. Wei et al. [37] took the building point clouds into consideration. The plane features were extracted as the primitives for registration. A plane shape descriptor based on the hole angles and distance of triangle centroids was developed to find corresponding plane features. The optimal transformation was obtained using a topological graph voting method. Hasheminasab et al. [27] proposed a method to integrate image and LiDAR point cloud. In order to obtain the complete point clouds, the plane features were applied to carry out point cloud registration. Tao et al. [28] proposed using 2-D feature lines to perform the point cloud registration. The point clouds were projected on horizontal plane and then the 2-D feature lines were extracted. Thus, the registration method mainly processed the point cloud data in 2-D space, so it had high computation speed. Tan et al. [35] began their work by extracting urban building facades from oblique photogrammetric point cloud and LiDAR point cloud. To describe the building facades, the object vicinity distribution feature [38] was introduced, which facilitated the registration of heterogeneous point clouds. Xu et al. [25] started their research by extracting plane segments from the voxelized point clouds. Then, the ratios of angles were developed to searching for the four-plane base set. The voxel-based four-plane congruent set algorithm was designed. The idea of the algorithm is similar to the 4PCS algorithm. Cheng et al. [39] proposed a registration method for the cross-source point clouds (terrestrial and airborne) based on the 2-D building outlines.

C. Specific Object-Based Registration Methods

The specific object-based methods implement point cloud registration by extracting particular objects and are only applicable in specific scenarios. Chan et al. [29] first proposed a method to extract the octagonal lamp poles, which were then treated as the registration primitives. Yang and Zang [9] started by extracting crest lines from the point clouds. Then, a deformation energy model was constructed to characterize the shape features of the crest lines. The corresponding crest lines were identified by comparing the shape features. The optimal transformation was subsequently determined based on a voting principle. Xu et al. [40] extracted the feature curves to perform the registration of seafloor point clouds. The isoline points were extracted by the triangulation network interpolation. The feature curves were obtained by the spline function interpolation. The RANSAC algorithm was applied to calculate the transformation. Ghorbani et al. [41] extracted the tree positions to perform the registration of the forest point clouds. The geometric constraints were used to facilitate the RANSAC algorithm for computing the transformation.

III. Point Cloud Registration Based on 2-D Feature Points

In this section, our point cloud registration method is given in detail. Specially, our work focuses on the registration of the building point clouds. The delineation of the method is depicted in Fig. 1. The procedures are as follows.

Step 1: Extraction of 2-D feature points. The source and target point clouds are projected onto the horizontal plane. Then, the 2-D feature lines are extracted from the projected point clouds. The intersection points of the 2-D feature lines are calculated.

Step 2: Calculation of 2-D transformation. The extracted 2-D feature points are utilized to form a set of feature triangles. Then, the feature triangles from two point clouds are matched via the

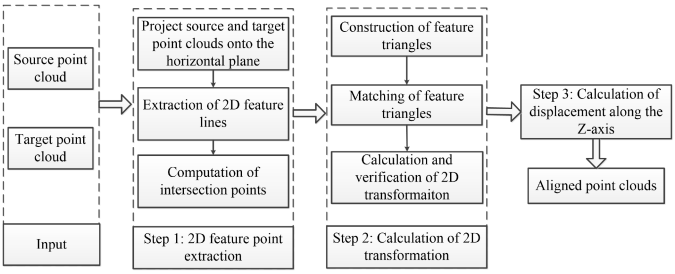


Fig. 1. Flowchart of the proposed registration method

geometric constraints. The 2-D transformation is calculated by each pair of matched feature triangles.

Step 3: Calculation of the *Z*-axis displacement. The cylindrical neighborhoods of the overlapping points between two 2-D point clouds are used to calculate the *Z*-axis displacement.

A. 2-D Feature Point Extraction

Accurate 2-D feature point extraction is paramount for successful registration in our method. The extraction process of the 2-D feature points is decomposed into two phases: extraction of 2-D feature lines and computation of intersection points. The obtained intersection points represent the 2-D feature points in our registration method. Compared to the traditional keypoints [42], [43], [44], the 2-D feature points extracted by our method offer more accurate position information, enabling to compute more accurate transformation.

1) Extraction of 2-D Feature Lines: The method in our previous work [28] is used to extract 2-D feature lines. Here, we just briefly describe the method. Readers can refer to Tao et al.'s [28] work for the parameter setting and analysis. First, the point cloud is projected onto the horizontal plane along the gravitational direction to get projected point cloud. Subsequently, the point density is computed for each projected point. The point density is defined as the number of the neighboring points of a point. The points with high density are preserved. Thus, we get the points on the 2-D lines because the points on the facades will project on the 2-D lines and have high density. To reduce computational burden, the extracted 2-D point cloud undergoes uniform sampling with a sampling interval of pr (pr denotes the resolution of the original point clouds, i.e., the mean value of the distances between the points and their closest points), making the 2-D point cloud have the same point cloud resolution as the original point clouds. Then, the region growing method is employed to extract 2-D feature lines. The singular value decomposition (SVD) [45] is applied to fit a 2-D line and compute the fitting residual of each projected point using its neighboring points. Here, the fitting residual is calculated as the ratio between the small singular value and the large one. The point with the minimal fitting residual is chosen as the seed, and its local neighborhood acts as the initial seed region. The distances between the near points and the fitted 2-D line are utilized as the criterion for expanding the seed region. By successively adding the near points, a 2-D feature line is obtained. Until no points can be added into the 2-D line, the next point with the minimal fitting residual is selected as the seed point. Another 2-D line grows. This procedure terminates until the fitting residual of the currently selected point exceeds a threshold. The method is used to extract 2-D feature lines from both the source and target point clouds.

2) Computation of Intersection Points: After finishing the 2-D feature line extraction, the subsequent step is to calculate the intersection points of the 2-D feature lines. The 2-D feature line set extracted from a point cloud is denoted as $\mathrm{LS} = \{l_1, l_2, \cdots, l_k\}$. For any two lines l_m and l_n, their formulas are denoted as

$$\begin{cases} a_m x + b_m y + c_m = 0 \\ a_n x + b_n y + c_n = 0. \end{cases} \tag{1}$$

If the angle between the two lines is larger than 10° or smaller than 170°, the two lines are considered as nonparallel. Thus, the intersection point of the two lines can be calculated using the standard formula as follows:

$$\begin{cases} x = \frac{-c_m b_n + b_m c_n}{a_m b_n - b_m a_n} \\ y = \frac{-a_m c_n + c_m a_n}{a_m b_n - b_m a_n}. \end{cases} \tag{2}$$

Alternatively, if the two lines l_m and l_n are parallel, the iteration proceeds to the next line pair. The same methodology is applied to compute intersection points for any two lines in the line set LS.

The process of the 2-D feature point extraction is shown in Fig. 2. Because our method treats the 2-D feature lines as infinitely long lines rather than line segments, the extracted 2-D feature points contain the real intersection points (the intersection points of the 2-D line segments) and unreal intersection points (the intersection points of the extension of the 2-D line segments), such as the points in the black circle in Fig. 2(c). This can help to increase the number of the 2-D feature points, which is beneficial for the feature point matching. In the overlapping area, if one unreal intersection point exists in source point cloud, there is also one corresponding unreal intersection point in target point cloud. That is the unreal intersection points exist both the source and target point clouds, and they are congenetic and can form matches.

B. Calculation of 2-D Transformation

1) Construction of Feature Triangles: The extracted 2-D feature point sets from the source and target point clouds are denoted as $\mathrm{FP}^s = \{\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, \cdots, \boldsymbol{p}_{n_1}^s\}$ and $\mathrm{FP}^t = \{\boldsymbol{p}_1^t, \boldsymbol{p}_2^t, \cdots, \boldsymbol{p}_{n_2}^t\}$, respectively. The maximum number of feature triangles constructed through exhaustive combinations of the feature points is respectively $C_{n_1}^3$ and $C_{n_2}^3$. As a result, the computation complexity is very high. In theory, each 2-D feature point just needs to form a feature triangle with its two nearest 2-D feature points. The obtained feature triangles are sufficient to calculate the 2-D transformation. However, in another point cloud, a 2-D feature point may have nearer 2-D feature points because of the intersection of other 2-D lines. Thus, the corresponding feature triangles cannot be identified. Therefore, we construct small feature triangles. The side lengths of the feature triangles are required to be smaller than a specific value ε_1. This can

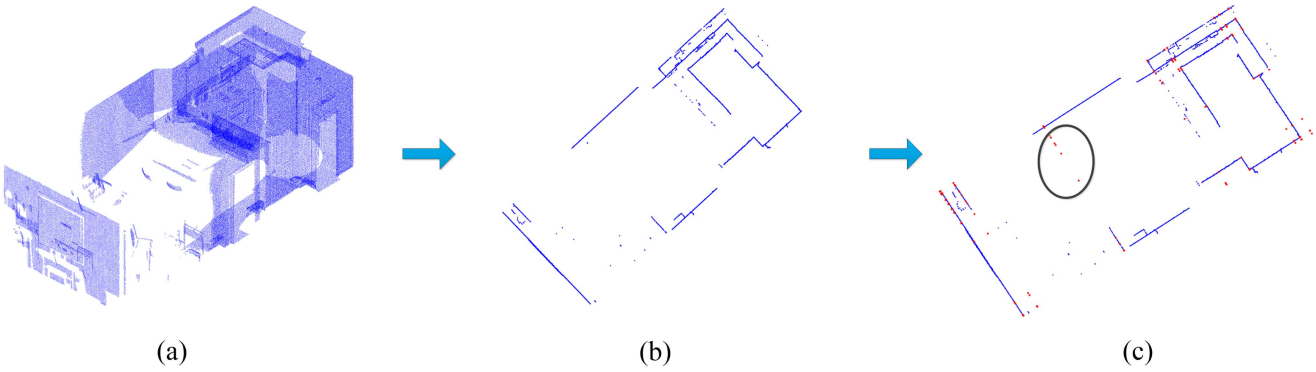


Fig. 2. Process of extracting 2-D feature points. (a) 3-D point cloud. (b) 2-D point cloud. (c) Extracted 2-D feature points (red points).

largely reduce the computation burden. Then, the equilateral and isosceles triangles are discarded because the vertexes of such triangles cannot be sorted (the vertexes will be sorted according to the side lengths later). It is worth noting that we do not need to discard the collinear triangles because only two 2-D feature points are enough to calculate the 2-D transformation. Hence, even though three 2-D feature points are collinear, they can also be applied to calculate the 2-D transformation. In order to eliminate equilateral and isosceles triangles, we require that the difference of any two side lengths of a triangle is larger than 3pr. In fact, the difference of any two angles is more suitable to eliminate the equilateral and isosceles triangles, but this needs to additionally compute the three angles of a triangle. In order not to increase the computation burden, we apply the difference of any two side lengths. The implementation detail of the construction of the feature triangles is presented in Algorithm 1

Algorithm 1 is also used to get the feature triangle set $\boldsymbol{T}^t$ and point set $\boldsymbol{Q}^t$ of the target point cloud. $\boldsymbol{Q}^s$ and $\boldsymbol{Q}^t$ will be explained later. In this algorithm, ε_1 is set as 1 for indoor scene point clouds and 3 for outdoor scene point clouds. This is because the outdoor scenes have larger scale. The two values are determined by many experiments. In most cases, the successful registration can be obtained by using the two values. The vertexes of a feature triangle have been sorted at this step. As a consequence, once a feature triangle in $\boldsymbol{T}^s$ is matched with a feature triangle in $\boldsymbol{T}^t$, three pairs of 2-D feature points are determined. When sorting, $\boldsymbol{p}_i^s, \boldsymbol{p}_j^s$, and $\boldsymbol{p}_k^s$, respectively, have the same order as L_1, L_2, and L_3.

2) Feature Triangle Matching: Let $\boldsymbol{T}^s = \{\boldsymbol{t}_1^s, \boldsymbol{t}_2^s, \cdots, \boldsymbol{t}_{e_1}^s\}$ be the set of feature triangles from the source point cloud and $\boldsymbol{T}^t = \{\boldsymbol{t}_1^t, \boldsymbol{t}_2^t, \cdots, \boldsymbol{t}_{e_2}^t\}$ be the set of feature triangles from the target point cloud. If the three side lengths of a feature triangle from the source point cloud are approximately equal to those of a feature triangle from the target point cloud, the two feature triangles can be considered congruent feature triangles, i.e., a match.

In order to quickly find the matches between $\boldsymbol{T}^s$ and $\boldsymbol{T}^t$, each feature triangle is viewed as a point in 3-D space. The *X*, *Y*, and *Z* coordinates of the point are respectively equal to the three side lengths of the feature triangle, as shown in Fig. 3. Accordingly, all the feature triangles are mapped into the points

Algorithm 1: Construction of Feature Triangles.

Input: the 2D feature point sets $\boldsymbol{FP}^s = \{\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, \cdots, \boldsymbol{p}_{n_1}^s\}$.
Output: the feature triangle set $\boldsymbol{T}^s$ and point set $\boldsymbol{Q}^s$.
L_1, L_2, L_3 denote side lengths and $d()$ denotes distance.
For $i = 1$ to $n_1 - 2$ **do**
  For $j = i + 1$ to $n_1 - 1$ **do**
    $L_1 = d(\boldsymbol{p}_i^s, \boldsymbol{p}_j^s)$
    If $L_1 > \varepsilon_1$
      continue.
    End If
    For $k = j + 1$ to n_1 **do**
      $L_2 = d(\boldsymbol{p}_j^s, \boldsymbol{p}_k^s)$
      $L_3 = d(\boldsymbol{p}_k^s, \boldsymbol{p}_i^s)$
      If $L2 > \varepsilon_1$ or $L3 > \varepsilon_1$
        continue.
      end If
      $dL_1 = L_1 - L_2, dL_2 = L_2 - L_3, dL_3 = L_3 - L_2$
      If $\min(dL_1, dL_2, dL_3) < 3pr$
        continue.
      End If
      Sort L_1, L_2, L_3 in ascending order and accordingly sort $\boldsymbol{p}_i^s, \boldsymbol{p}_j^s, \boldsymbol{p}_k^s$.
      Add sorted $\boldsymbol{p}_i^s, \boldsymbol{p}_j^s, \boldsymbol{p}_k^s$ in $\boldsymbol{T}^s$.
      Add sorted L_1, L_2, L_3 in $\boldsymbol{Q}^s$.
    End for
  End for
End for

in 3-D space, as shown in Fig. 3(b). That is all the feature triangles are represented by the points.

In theory, the corresponding side lengths of two congruent feature triangles are the same, thus leading to three geometric constraints. These geometric constraints can be used to quickly find the corresponding feature triangles. We incorporate the kd-tree structure to accelerate the search of feature triangles and enhance matching efficiency. The points representing the feature triangles in $\boldsymbol{T}^s$ are denoted as $\boldsymbol{Q}^s$ and those representing the feature triangles in $\boldsymbol{T}^t$ are denoted as $\boldsymbol{Q}^t$. $\boldsymbol{Q}^s$ and $\boldsymbol{Q}^t$ have been

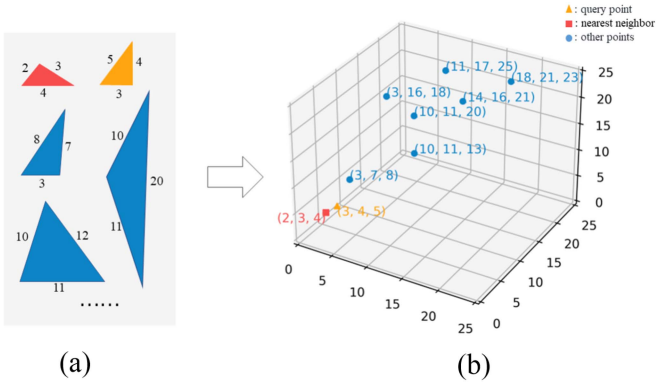


Fig. 3. Illustration of mapping feature triangles into the points in 3-D space. (a) Feature triangles. (b) Points representing the feature triangles. The yellow feature triangle is from the source point cloud and others are from the target point cloud. The red feature triangle is the neatest feature triangle of the yellow feature triangle.

achieved in Algorithm 1. Thus, we can use the kd-tree structure to organize the points in $\boldsymbol{Q}^t$. For each point in $\boldsymbol{Q}^s$, the nearest neighbor search is executed to find its closest point in $\boldsymbol{Q}^t$. If the distance between a point and its closest point is less than a threshold, the related two feature triangles are deemed as a match

$$\left\|\boldsymbol{q}_i^s - \boldsymbol{q}_{i1}^t\right\| < d_{\min} \tag{3}$$

where $d_{\min}$ is set as 0.2pr by many trials, $\|\cdot\|$ is the Euclidean distance, $\boldsymbol{q}_i^s$ is a point in $\boldsymbol{Q}^s$ and $\boldsymbol{q}_{i1}^t$ is the closest point to $\boldsymbol{q}_i^s$ in $\boldsymbol{Q}^t$. For each match, three pairs of 2-D feature points are determined to calculate the candidate 2-D transformation.

3) Calculation and Verification of 2-D Transformation: Each match can be used to calculate a 2-D transformation. The SVD [45] is applied to compute the rotation matrix $\boldsymbol{r}_a$ and translation vector $\boldsymbol{t}_a$. The three 2-D feature points of a feature triangle from the source point cloud are denoted as $\{\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, \boldsymbol{p}_3^s\}$ and those of its corresponding feature triangle from the target point cloud are denoted as $\{\boldsymbol{p}_1^t, \boldsymbol{p}_2^t, \boldsymbol{p}_3^t\}$. The following matrices are calculated:

$$\tilde{\boldsymbol{P}}_{2\times 3}^s = \left[\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, \boldsymbol{p}_3^s\right] \tag{4}$$

$$\tilde{\boldsymbol{P}}_{2\times 3}^t = \left[\boldsymbol{p}_1^t, \boldsymbol{p}_2^t, \boldsymbol{p}_3^t\right]. \tag{5}$$

Subsequently, the SVD is performed

$$\boldsymbol{U}\boldsymbol{\Xi}\boldsymbol{V}^T = \text{svd}\left(\left[\tilde{\boldsymbol{P}}_{2\times 3}^s - \bar{\boldsymbol{P}}_{2\times 1}^s \begin{bmatrix}1 & 1 & 1\end{bmatrix}\right] \times \left[\tilde{\boldsymbol{P}}_{2\times 3}^t - \bar{\boldsymbol{P}}_{2\times 1}^t \begin{bmatrix}1 & 1 & 1\end{bmatrix}\right]^T\right) \tag{6}$$

where $\bar{\boldsymbol{P}}_{2\times 1}^t$ is the centroid of $\tilde{\boldsymbol{P}}_{2\times 3}^t$, and $\bar{\boldsymbol{P}}_{2\times 1}^s$ is the centroid of $\tilde{\boldsymbol{P}}_{2\times 3}^s$. The 2-D rotation matrix is calculated as

$$\boldsymbol{r}_a = \boldsymbol{V}\boldsymbol{U}^T \tag{7}$$

and the 2-D translation vector is calculated as

$$\boldsymbol{t}_a = \bar{\boldsymbol{P}}_{2\times 1}^t - \boldsymbol{r}_a \bar{\boldsymbol{P}}_{2\times 1}^s. \tag{8}$$

After calculating a 2-D transformation, it is imperative to assess the reliability of the 2-D transformation. The 2-D source point cloud (i.e., the points on the 2-D lines) is transformed using the calculated 2-D transformation. The degree of overlap between the transformed 2-D source point cloud and 2-D target point cloud is computed as follows:

$$\text{overlap} = \frac{\text{Number of overlapping points}}{\min(a, b)} \tag{9}$$

where a and b represent the point numbers of the 2-D source and 2-D target point clouds, respectively. For each point in the transformed 2-D source point cloud, its nearest point is searched from the 2-D target point cloud. If the distance between the two points is smaller than a threshold (2pr in this article), the point in the transformed 2-D source point cloud is considered as an overlapping point.

The optimal 2-D transformation $\boldsymbol{T}' = \left[\bar{\boldsymbol{r}} \quad \bar{\boldsymbol{t}}\right]$ is determined by identifying the one that maximizes the degree of overlap between the transformed 2-D source point cloud and 2-D target point cloud. The calculation of the 2-D transformation is presented in Algorithm 2.

Algorithm 2: 2D Transformation Calculation.

Input: the feature triangle sets $\boldsymbol{T}^s$ and $\boldsymbol{T}^t$, and point sets $\boldsymbol{Q}^s$ and $\boldsymbol{Q}^t$.
Output: the optimal 2D transformation.
Search the nearest point $\boldsymbol{q}_{i1}^t$ from $\boldsymbol{Q}^t$ for each point $\boldsymbol{q}_i^s$ in $\boldsymbol{Q}^s$ by using kd-tree.
Set the initial overlap: $\text{overlap}_0 = 0$.
For $i = 1$ to e_1 **do**
  If $\boldsymbol{q}_i^s$ and $\boldsymbol{q}_{i1}^t$ satisfy the geometric constraints defined by (3)
    The two feature triangles $\boldsymbol{t}_i^s$ and $\boldsymbol{t}_{i1}^t$ form a match.
    Calculate the 2D transformation by (7) and (8).
    Calculate the degree of overlap overlap_i
    If $\text{overlap}_i > \text{overlap}_0$
      The 2D transformation is the optimal one.
      $\text{overlap}_0 = \text{overlap}_i$
    End if
  End if
End for

C. Calculation of the Z-axis displacement

Now, the 2-D rotation and 2-D translation have been obtained. As long as the Z-axis displacement is calculated, the 3-D transformation can be easily got. Tao et al. [28] proposed a good method to do this. For completeness, the method is briefly described here. Readers can refer to Tao et al.'s [28] work for parameter setting and analysis.

Initially, the 2-D source point cloud is transformed by using the 2-D transformation. Subsequently, the overlapping region between the transformed 2-D source point cloud and 2-D target point cloud is identified, from which a certain number of points are randomly selected from the transformed 2-D source point cloud. We can also get the corresponding points from the 2-D source point cloud. For each pair of points, the cylindrical neighborhoods of the two points are, respectively, extracted from

the source point cloud and target point cloud. The lowest points in the two cylindrical neighborhoods are used to calculate the Z-coordinate difference. All the pairs of points can be used to calculate a set of Z-coordinate differences. Ultimately, a clustering algorithm is applied to group these Z-coordinate differences. The Z-axis displacement is the mean value of the Z-coordinate differences in the largest cluster. Thus, the 3-D transformation is calculated as

$$\boldsymbol{T}'' = \begin{bmatrix} \boldsymbol{R}_{3\times3} & \boldsymbol{t}_{3\times1} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{r}} & 0_{2\times1} & \bar{\boldsymbol{t}} \\ 0_{1\times2} & 1 & t_z \end{bmatrix} \tag{10}$$

where t_z is the Z-axis displacement, $\boldsymbol{R}_{3\times3}$ is the 3-D rotation matrix, and $\boldsymbol{t}_{3\times1}$ is the 3-D translation vector.

IV. Experiments and Analysis

The experiments are implemented to assess the proposed registration method. The line-based method [28] and two LSD-based methods [16], [34] are used for comparison with our method. The line-based method is our previous method. It has good computation efficiency and high registration accuracy. The current method is an improvement to the line-based method, so the line-based method is used to compare with our method. The first LSD-based method uses the LoVS descriptor to establish correspondences, and then uses GC1SAC algorithm to calculate the 3-D transformation. The iteration number of the GC1SAC algorithm is set as the number of the established correspondences. The method is denoted as "LoVS+GC1SAC." The second LSD-based method is a recently proposed registration method, which has good computation efficiency and high registration accuracy in their experiments. The method uses the LOVC descriptor to establish correspondences and then uses two-point RANSACWC algorithm to calculate the 3-D transformation. The iteration number of the two-point RANSACWC algorithm is set to 20 000 so as to get successful registration for all pairs of point clouds. The method is denoted as "LOVC+two-point RANSACWC."

All the experiments are conducted by MATLAB. The experiments run on a PC equipped with an AMD Ryzen 9 5900HX 3.30 GHz processor and 32 GB of RAM.

A. Experimental Data

An indoor Lidar dataset [46] and two outdoor Lidar dataset [3], [47] are employed to carry out the experiments. The indoor dataset[1] is acquired using the FARO Focus 3D X330 HDR scanner. This dataset comprises the point clouds of the five scenes: lobby, boardroom, apartment, bedroom, and loft. The publishers perform multiple scans and obtain multiple point clouds in each scene. For the purpose of our study, three point clouds from the apartment scene and three point clouds from the boardroom scene are chosen. Each scene has two pairs of point clouds. The first outdoor dataset[2] is collected by the Leica C10 laser scanner, providing the point clouds of the city and castle scenes. Each scene also includes multiple point clouds. Four point clouds are chosen from the castle scene and three point cloud are chosen from the city scene. The castle scene has three pairs of point clouds and the city scene has two pairs of point clouds. The second outdoor dataset is from the WHU-TLS dataset.[3] Three point clouds are chosen from the park scene, forming two pairs of point clouds. These point clouds are obtained by the Riegl VZ-400 scanner. Finally, 11 pairs of point clouds are used to perform the experiments. The point clouds can be found in Figs. 5 and 7. The experimental data contain different scenes, different data quality and different degree of overlap. These make our evaluation more comprehensive. In the indoor dataset, the point clouds are symmetric and have featureless and similar local surfaces, which makes registration harder. In the outdoor dataset, the noise and point density variation are more severe. The outdoor scenes also contain many cluttered objects such as tree. The park scene has the buildings with curved contours. The information of the point cloud pairs is list in Table I.

TABLE I
Information of Point Cloud Pairs

Scene	Point cloud pair	pr	degree of overlap
Apartment	A1	0.0224	0.7757
	A2	0.0224	0.8600
Boardroom	B1	0.0256	0.6549
	B2	0.0256	0.5554
Castle	Ca1	0.1652	0.7262
	Ca2	0.1652	0.3987
	Ca3	0.1652	0.5024
City	Ci1	0.1230	0.4432
	Ci2	0.1230	0.2756
Park	P1	0.1705	0.394
	P2	0.1705	0.572

B. Evaluation Metrics

In order to assess the accuracy of the registration methods, three metrics are employed. The rotation error is calculated using the following equation:

$$e_R = \arccos\left(\frac{\text{trace}(\boldsymbol{R}_{\text{true}}(\boldsymbol{R})^{-1})-1}{2}\right)\frac{180}{\pi} \tag{11}$$

where $\boldsymbol{R}$ represents the computed rotation matrix, $\boldsymbol{R}_{\text{true}}$ denotes the true rotation matrix, $\arccos()$ is the inverse cosine function, and trace denotes the trace of a matrix. Because our method computes the Z-axis displacement separately, the translation error is bifurcated into horizontal and vertical errors

$$e_{T_h} = \|\bar{\boldsymbol{t}}_{\text{true}} - \bar{\boldsymbol{t}}\| \tag{12}$$

$$e_{T_v} = \|t_{z-\text{true}} - t_z\| \tag{13}$$

where $\bar{\boldsymbol{t}}$ signifies the computed 2-D translation vector, t_z is the computed Z-axis displacement, $\bar{\boldsymbol{t}}_{\text{true}}$ is the true 2-D translation vector, and $t_{z-\text{true}}$ represents the true Z-axis displacement. The

[1][Online]. Available: http://redwood-data.org/indoor_lidar_rgbd/download.html

[2][Online]. Available: http://www.cvg.ethz.ch/research/saldir-rgbd-registration/

[3][Online]. Available: http://3s.whu.edu.cn/ybs/en/benchmark.htm

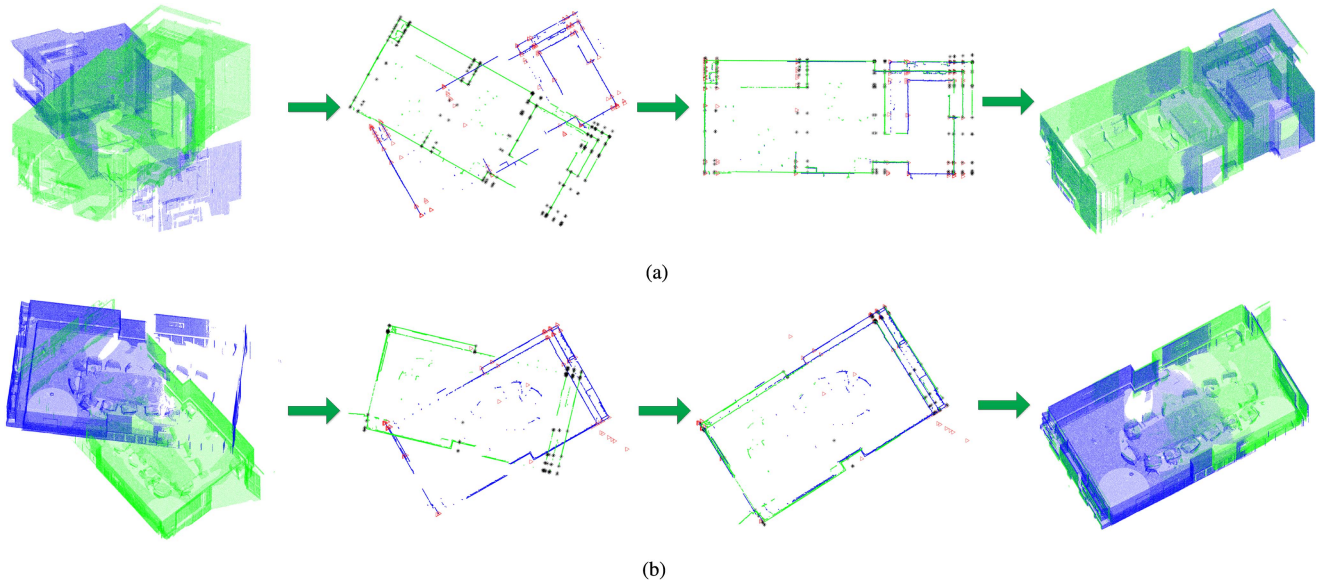


Fig. 4. Registration process of the two pairs of point clouds in the apartment and boardroom scenes. In each subfigure, the original point clouds, extracted 2-D point clouds, aligned 2-D point clouds, and aligned 3-D point clouds are exhibited from left to right. (a) Apartment. (b) Boardroom.

true values $\boldsymbol{R}_{\text{true}}$, $\bar{\boldsymbol{t}}_{\text{true}}$, and $t_{z-\text{true}}$ of the apartment, boardroom, castle, and city scenes are derived through manual coarse registration and ICP fine registration, while the true values of the park scene are provided by the publishers.

C. Reconstruction of the Indoor Scene

In order to better understand our registration method, the registration process of one pair of point cloud for each scene is exhibited in Fig. 4. The 2-D feature points are also displayed in the figure. It can be discerned that the 2-D feature points are well registered. Thus, the 2-D point clouds are registered together. In the last plot of each subfigure, the 3-D point clouds have been automatically registered together. The registration results are good enough to offer initial position for the fine registration.

The registration accuracy of the four methods is displayed in Table II. The computation time is also listed in the table. It can be seen that the two LSD-based methods obtain relatively big rotation error, horizontal error, and vertical error. The line-based method and our method obtain small rotation error, horizontal error, and vertical error. This indicates that the line-based method and our method have higher registration accuracy. It is worth noting that it is meaningless to compare the vertical errors of the line-based method and our method, because the two methods apply the same manner to calculate the Z-axis displacement. The reason why the calculated vertical errors are different is that the manner has randomness. The aim for exhibiting the vertical errors is to illustrate the manner is effective. It can calculate an accurate Z-axis displacement.

The two LSD-based methods extract keypoints (i.e., point feature) as the registration primitive. The point features are easy to be affected by point noise and variations in point density, so they get low registration accuracy. On the "B2" point cloud pair, the two methods get unsuccessful registration due to the featureless and similar local surfaces. The featureless and similar local surfaces make the LSDs indistinctive, which is not helpful to establish correct correspondences. The line-based method extracts 2-D feature lines to perform the registration. A 2-D line is fitted by many points, so it has higher accuracy. Therefore, the line-based method obtains high registration accuracy. On the "B1" point cloud pair, the method gets unsuccessful registration. This is because the two point clouds are symmetric. The 2-D feature lines are easy to be affected by the problem. Our method uses the intersection points of the 2-D lines as the 2-D feature points, so the 2-D feature points also have high accuracy. Hence, our method can achieve high registration accuracy. The 2-D feature points are used to construct the feature triangles, which are not affected by the symmetry problem. Thus, our method gets successful registration on the "B1" point cloud pair.

TABLE II
REGISTRATION ACCURACY AND COMPUTATION EFFICIENCY OF THE FOUR METHODS ON THE INDOOR DATASET

Point cloud pair		LOVC+two-point RANSACWC	LoVS+ GC1SAC	Line-based method	Ours
A1	e_R	**0**	0.6052	0.6691	0.0685
	e_{T_h}	**0.0011**	0.2491	0.0226	0.0101
	e_{T_v}	0.0037	0.0402	**0.0008**	0.0012
	Time (s)	85.8722	226.8242	25.6398	**19.0217**
A2	e_R	0.9815	0.2533	0.1008	**0**
	e_{T_h}	0.3962	0.0877	**0.0153**	0.0165
	e_{T_v}	0.0537	0.0301	**0.0001**	0.0014
	Time (s)	90.8687	227.2748	18.6129	**15.0189**
B1	e_R	1.0075	1.385	90.0001	**0.6391**
	e_{T_h}	1.1361	1.5624	5.4540	**0.1687**
	e_{T_v}	0.0055	0.0849	0.1012	**0.0009**
	Time (s)	88.5349	94.9669	14.2025	**11.4528**
B2	e_R	1.1800	179.7188	**0**	0.0969
	e_{T_h}	3.1796	5.6337	**0.0789**	0.1051
	e_{T_v}	0.0227	0.0040	0.0042	**0.0041**
	Time (s)	74.7922	93.9362	14.6388	**12.4394**

The best results are denoted by bold font and the unsuccessful registration is labeled by underline.

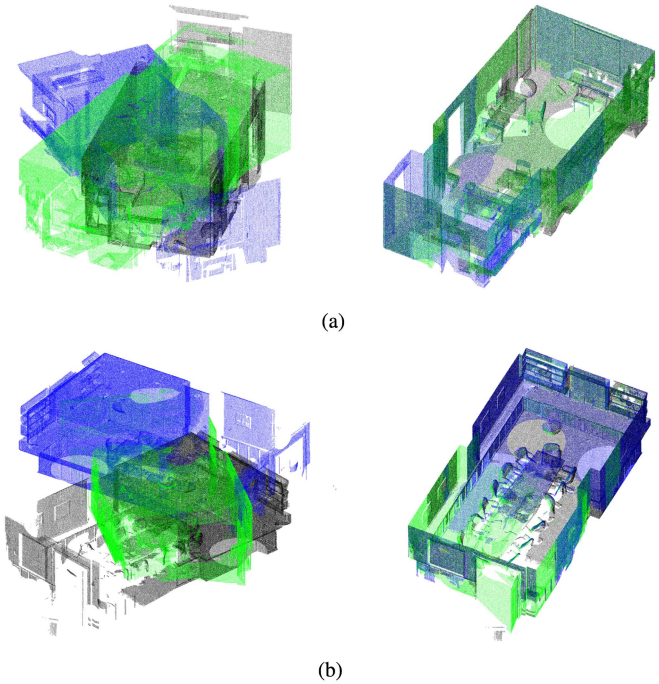


Fig. 5. Reconstructed complete scenes of the apartment and boardroom. In each subfigure, the original point clouds are on the left and the complete scene point cloud is on the right. The points on the ceiling are removed for a better view. (a) Apartment. (b) Boardroom.

TABLE III
REGISTRATION ACCURACY AND COMPUTATION EFFICIENCY OF THE FOUR METHODS ON THE OUTDOOR DATASET

Point cloud pair		LOVC+two-point RANSACWC	LoVS+ GC1SAC	Line-based method	Ours
Ca1	e_R	2.3468	0.4387	0.0685	**0**
	e_{T_h}	1.3340	0.3574	**0.1116**	1.0117
	e_{T_v}	1.7266	0.0920	0.0201	**0.0060**
	Time (s)	1951.2876	1895.1842	224.7621	**70.8994**
Ca2	e_R	5.8114	1.5298	0.2093	**0.1186**
	e_{T_h}	0.7851	0.2348	0.5476	**0.1347**
	e_{T_v}	0.6650	0.5517	0.0622	**0.0174**
	Time (s)	1564.4844	1549.6880	98.2310	**45.1420**
Ca3	e_R	0.1607	0.5143	0.2438	**0.1046**
	e_{T_h}	**0.0219**	0.4185	0.3392	0.0479
	e_{T_v}	**0.0049**	0.0547	0.0294	0.0117
	Time (s)	1950.1723	1918.3672	52.1741	**26.8669**
Ci1	e_R	2.5932	1.2517	**0**	**0**
	e_{T_h}	1.9284	0.3407	**0.2318**	0.2671
	e_{T_v}	3.1590	0.0947	**0.0111**	0.0390
	Time (s)	360.4770	422.1552	52.1200	**28.5067**
Ci2	e_R	**0.3711**	0.3926	89.9999	89.9999
	e_{T_h}	0.1678	**0.0711**	35.7967	21.4919
	e_{T_v}	0.0532	0.0325	5.2014	1.2298
	Time (s)	351.5018	439.7266	106.7022	**99.1623**
P1	e_R	**0.1544**	0.6598	0.1609	0.2493
	e_{T_h}	**0.0382**	1.3223	0.2980	0.1404
	e_{T_v}	0.0618	0.2723	**0.0261**	0.0361
	Time (s)	3398.3542	3466.7312	448.8186	**78.3563**
P2	e_R	0.2987	0.2014	**0.0703**	0.1935
	e_{T_h}	**0.1452**	0.5182	1.6765	1.5146
	e_{T_v}	0.0728	0.0832	**0.0170**	0.0193
	Time (s)	3588.8361	3559.689	567.6456	**80.4083**

The best results are denoted by bold font and the unsuccessful registration is labeled by underline

In aspect of computation efficiency, the two LSD-based methods are more time-consuming. The main time of the two methods is, respectively, spent on the GC1SAC and Two-point RANSACWC algorithms because the two algorithms need many iterations. The line-based method processes the point cloud data in 2-D space, so it has high computation efficiency. Our method has the best computation efficiency. This is owing to the following reasons. First, our method also processes the point cloud data in 2-D space. Second, a strategy is proposed to quickly get the matches. It first constructs the feature triangles by using the 2-D feature points. Only the small feature triangles are preserved, which largely reduces the number of the feature triangles. Then, the geometric constraints and kd-tree are used to quickly find the corresponding feature triangles.

According to the calculated transformation of each point cloud pair, all the point clouds of one scene are transformed into the same coordinate system. The complete scenes obtained by our method are shown in Fig. 5. Different point clouds are colored by different colors. From the complete scenes, we can see that there is not obvious gap between the point clouds. Our registration method can provide a high-quality reconstructed scene.

D. Reconstruction of the Outdoor Scene

The registration process of one pair of point clouds for each scene is illustrated in Fig. 6. We can see that the three pairs of the point clouds have been registered well. The 2-D feature points are transformed into the same coordinate system, and the 2-D point clouds are thus registered. For the outdoor point clouds, our method is also effective. In the outdoor scenes, the distribution of the 2-D feature points is irregular and cluttered. There are many unreal 2-D feature points that are far away from the 2-D point clouds. These far 2-D feature points are useful. As long as the two 2-D lines used for calculating the 2-D feature point exist in both the source and target point clouds, the unreal 2-D feature point also exists in both the source and target point clouds. In addition, the number of the 2-D feature points is also large, which can increase computation burden. Fortunately, we only use the small feature triangles to calculate the transformation.

The registration accuracy and computation efficiency of the four methods are presented in Table III. The two LSD-based methods still get relatively large rotation error, horizontal error, and vertical error. This is because the keypoints used by the two methods are easy to be affected by the noise and variation in point density. Due to external factors, such as weather and scanning distance, the perception capability of the LiDAR is constrained by the outdoor circumstances, which results in the increased point noise. The variation in point density is also obvious. Hence, the two LSD-based methods obtain poor registration accuracy. The line-based method and our method obtain smaller rotation error, horizontal error and vertical error because they use 2-D feature lines and 2-D feature points to perform the registration. Both of the 2-D feature lines and 2-D feature points have high accuracy, which results in accurate registration. On the "Ci2" point cloud pair, the two methods get unsuccessful registration. This is because the point cloud pair has low degree of overlap. There is no corresponding 2-D lines between the two point clouds.

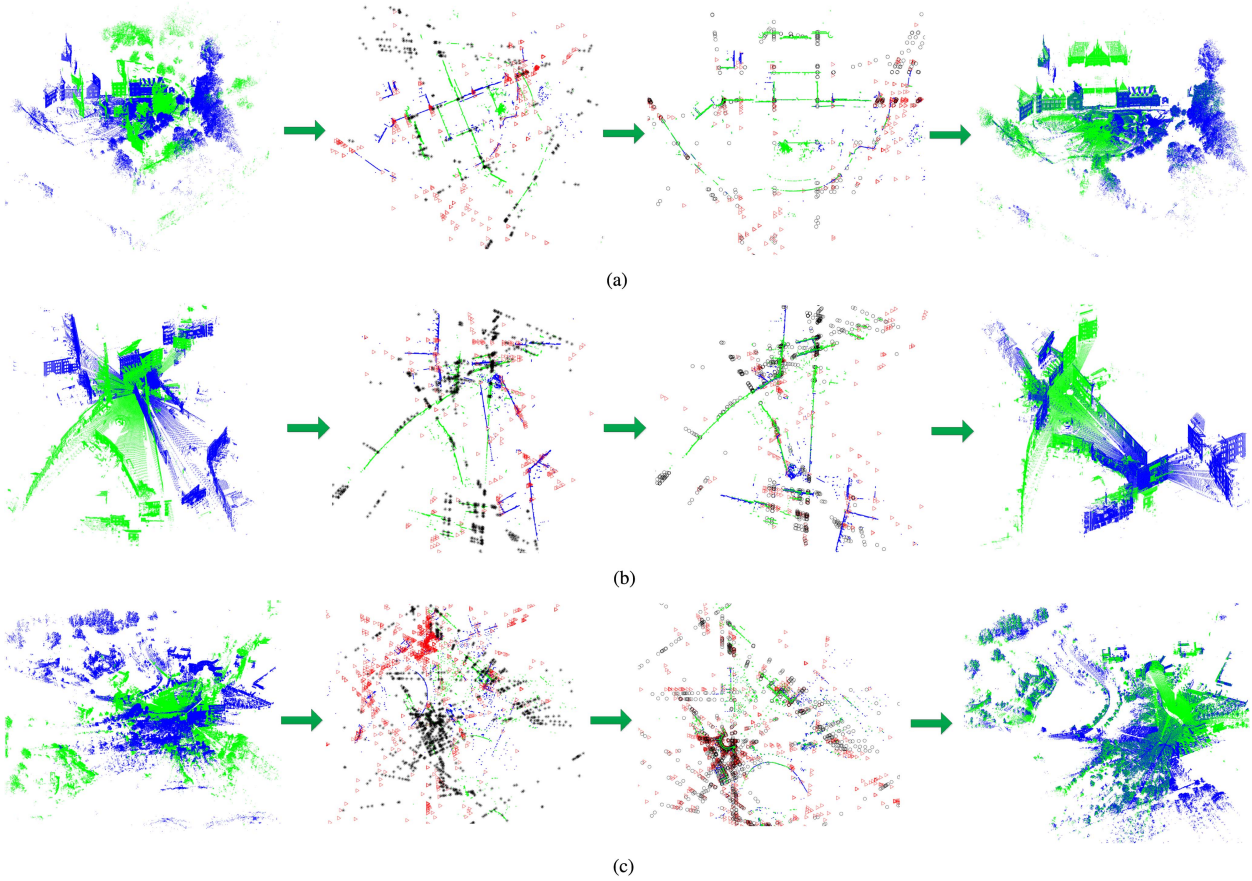


Fig. 6. Registration process of the three pairs of point clouds in the castle, city, and park scenes. In each subfigure, the original point clouds, extracted 2-D point clouds, aligned 2-D point clouds, and aligned 3-D point clouds are exhibited from left to right. (a) Castle. (b) City. (c) Park.

On the outdoor scenes, our method still achieves the best computation efficiency. The difference between our method and the line-based method is more significant. The line-based method costs much time on finding the corresponding 2-D feature lines, while our method uses a fast strategy to find the corresponding feature triangles. The two LSD-based methods are rather time-consuming on the castle and park scenes. This is because there are many trees in the scenes. As a result, many points on the trees are extracted as the keypoints. The descriptors of so many keypoints cost much time to compute.

All the point clouds of a scene are transformed into the same coordinate system by using the calculated transformations. The complete scene point cloud is obtained. Because our method gets an unsuccessful registration on the city scene, only the complete scenes of the castle and park are shown in Fig. 7. We can see that all the point clouds are aligned together. The aligned point cloud is able to render the real scene.

E. Parameter Analysis

In our method, there are two main parameters, which are the threshold ε_1 of the side lengths and threshold $d_{\min}$ of the geometric constraints. In order to analyze the influence of the two parameters, we choose one pair of point clouds from the indoor dataset and outdoor dataset, respectively. First, the value

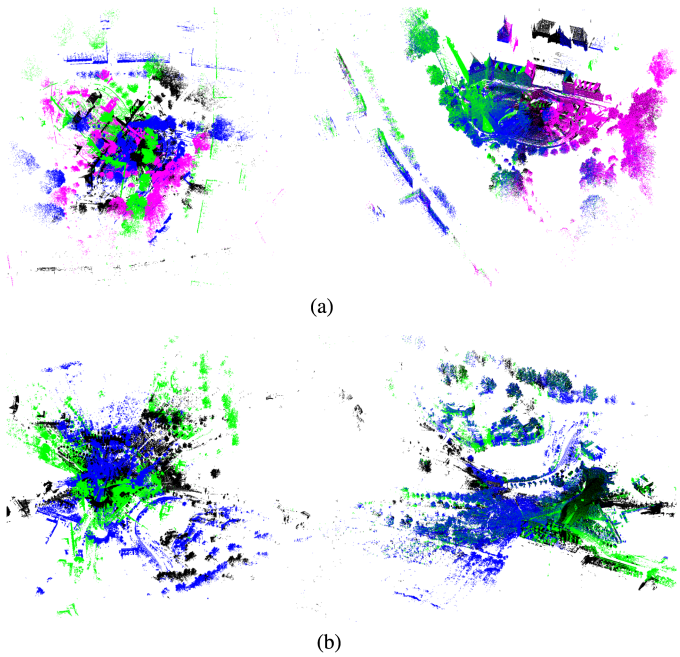


Fig. 7. Reconstructed complete scenes of the castle and park. In each subfigure, the original point clouds are on the left and the complete scene point cloud is on the right. (a) Castle. (b) Park.

TABLE IV
REGISTRATION ACCURACY AND COMPUTATION EFFICIENCY UNDER DIFFERENT VALUES OF ε_1

	Value of ε_1	0.5	1	2	3
A1	e_R	0.0685	0.0685	0	0
	e_{T_h}	0.0101	0.0101	0.0186	0.0186
	e_{T_v}	0.0004	0.0012	0.0021	0.0002
	Time (s)	14.4155	19.0217	29.3739	133.2978
	Value of ε_1	2	3	4	5
Ca1	e_R	0	0	0	0
	e_{T_h}	1.0117	1.0117	1.0117	1.0117
	e_{T_v}	0.0921	0.0060	0.0037	0.0056
	Time (s)	63.9707	70.8994	90.1527	117.4916

TABLE V
REGISTRATION ACCURACY AND COMPUTATION EFFICIENCY UNDER DIFFERENT VALUES OF d_{MIN}

	Value of d_{min}	0.05pr	0.1pr	0.2pr	0.3pr
A1	e_R	0.1897	0.0685	0.0685	0.0685
	e_{T_h}	0.0215	0.0101	0.0101	0.0101
	e_{T_v}	0.0007	0.0012	0.0012	0.0008
	Time (s)	15.3247	16.5552	19.0217	20.6800
Ca1	e_R	89.9998	0	0	0
	e_{T_h}	38.5136	1.0117	1.0117	1.0117
	e_{T_v}	1.2385	0.006	0.006	0.0294
	Time (s)	60.7092	63.3584	70.8994	77.8094

of d_{min} is fixed as 0.2pr, the value of ε_1 is increased from 0.5 to 3 for the indoor scene and from 2 to 5 for the outdoor scene. The obtained registration accuracy and computation time are listed in Table IV. As we can see, when the value of ε_1 increases, the rotation errors and translation errors almost keep unchanged. The change of the vertical errors is due to the randomness of the manner to calculate the *Z*-axis displacement. Therefore, the registration accuracy is almost not influenced by the value of ε_1. The computation time is increasing as the value of ε_1 increases. This is because more and more feature triangles are preserved. The computation efficiency is influenced by the value of ε_1. However, we have tested different values of ε_1 to perform the registration on all point cloud pairs before though the results are not displayed. When the value of ε_1 is too small, the registration will fail. The value of ε_1 should be set to get successful registration and reduce the computation time.

Then, the value of ε_1 is fixed as 1 for indoor scene and 3 for outdoor scene. The value of d_{min} is increased from 0.05 to 0.3pr. The registration accuracy and computation time are listed in Table V. When the value of d_{min} is 0.05pr, on the "Ca1" point cloud pair, the registration is unsuccessful. As the value increases from 0.1 to 0.5pr, the rotation and translation errors are unchanged. Therefore, the value of d_{min} has less influence on registration accuracy. The computation time is increasing as the value increases, but the influence is not very big.

V. CONCLUSION

A registration method based on the 2-D feature points was proposed in this article. The 2-D feature lines were first extracted from the source and target point clouds. Then, the 2-D feature points were calculated by the intersections of the 2-D lines. These 2-D feature points were used to construct feature triangles, which were then matched by the geometric constraints. The matched feature triangles were applied to calculate the 2-D transformation. Finally, the *Z*-axis displacement was calculated by the cylinder neighborhoods of the point pairs in the overlapping area between 2-D source and target point clouds. The 3-D transformation was achieved by combining the *Z*-axis displacement and 2-D transformation.

The experiments had been carried out to demonstrate the performance of our registration method. Our method got good performance in aspect of registration accuracy. The 2-D feature lines were fitted by many points, so they had good accuracy. The 2-D feature points were calculated by the intersections of the 2-D feature lines, so the 2-D feature points also had good accuracy. This ensured that the proposed method had high registration accuracy. In aspect of computation efficiency, our method got excellent performance. First, our method processed the point cloud data in 2-D space. The 2-D feature points were extracted as the registration primitives. Then, the 2-D feature points were used to construct the feature triangles. Only the small feature triangles were used to calculate the 2-D transformation. The geometric constraints were used to quickly match the feature triangles. A fast strategy was proposed to find the corresponding feature triangles. Thus, our method had high computation efficiency. In comparison with our previous method, the proposed method had better computation efficiency and comparative registration accuracy.

The limitation of our method is that it is only suitable for the building point clouds scanned by the terrestrial laser scanner. In future, we want to design a voxel-based registration method that is suitable for the building point clouds with arbitrary poses. Also, the method should have good computation efficiency and high registration accuracy.

VI. DISCUSSION

This article focuses on the registration of the building point clouds, which can be used for the reconstruction of the real scenes. In the field of remote sensing, point cloud registration has attracted much attention. Therefore, this article is in the scope of JSTARS.

ACKNOWLEDGMENT

The authors would like to thank the publishers of the datasets used in this article.

REFERENCES

[1] S. Xia, D. Chen, R. Wang, J. Li, and X. Zhang, "Geometric primitives in LiDAR point clouds: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 685–707, 2020.

[2] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5556–5565.

[3] Z. Dong et al., "Registration of large-scale terrestrial laser scanner point clouds: A review and benchmark," *ISPRS J. Photogrammetry Remote Sens.*, vol. 163, pp. 327–342, May 2020.

[4] L. Guo, Y. Wu, L. Deng, P. Hou, J. Zhai, and Y. Chen, "A feature-level point cloud fusion method for timber volume of forest stands estimation," *Remote Sens.*, vol. 15, no. 12, Jun. 2023, Art. no. 2995.

[5] P. Bożek, J. Janus, and B. Mitka, "Analysis of changes in forest structure using point clouds from historical aerial photographs," *Remote Sens.*, vol. 11, no. 19, Oct. 2019, Art. no. 2259.

[6] Ö. Zováthi, B. Nagy, and C. Benedek, "Point cloud registration and change detection in urban environment using an onboard lidar sensor and MLS reference data," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 110, Jun. 2022, Art. no. 102767.

[7] X. Zhang, C. Glennie, and A. Kusari, "Change detection from differential airborne LiDAR using a weighted anisotropic iterative closest point algorithm," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 7, pp. 3338–3346, Jul. 2015.

[8] J. Shao et al., "Automated markerless registration of point clouds from TLS and structured light scanner for heritage documentation," *J. Cultural Heritage*, vol. 35, pp. 16–24, Jan. 2019.

[9] B. Yang and Y. Zang, "Automated registration of dense terrestrial laser-scanning point clouds using curves," *ISPRS J. Photogrammetry Remote Sens.*, vol. 95, pp. 109–121, Sep. 2014.

[10] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[11] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vis. Comput.*, vol. 10, no. 3, pp. 145–155, Apr. 1992.

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

[13] D. Aiger, N. J. Mitra, and D. Cohen-Or, "4-points congruent sets for robust pairwise surface registration," *ACM Trans. Graph.*, vol. 27, no. 3, p. 1–10, Aug. 2008.

[14] N. Mellado, D. Aiger, and N. J. Mitra, "Super4PCS: Fast global pointcloud registration via smart indexing," *Comput. Graph. Forum*, vol. 33, no. 5, pp. 205–215, Feb. 2015.

[15] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "An integrated framework for 3-D modeling, object detection, and pose estimation from point-clouds," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 3, pp. 683–693, Mar. 2015.

[16] S. Quan, J. Ma, F. Hu, B. Fang, and T. Ma, "Local voxelized structure for 3D binary feature representation and robust registration of point clouds from low-cost sensors," *Inf. Sci.*, vol. 444, pp. 153–171, May 2018.

[17] W. Tao, X. Hua, X. He, J. Liu, and D. Xu, "Automatic multi-view registration of point clouds via a high-quality descriptor and a novel 3d transformation estimation technique," *Vis. Comput.*, vol. 40, pp. 2615–2630, Jun. 2023.

[18] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 199–208.

[19] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8957–8965.

[20] S. Ao, Q. Hu, B. Yang, A. Markham, and Y. Guo, "SpinNet: Learning a general surface descriptor for 3D point cloud registration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11748–11757.

[21] P. W. Theiler, J. D. Wegner, and K. Schindler, "Keypoint-based 4-points congruent sets–automated marker-less registration of laser scans," *ISPRS J. Photogrammetry Remote Sens.*, vol. 96, pp. 149–163, Oct. 2014.

[22] M. Mohamad, D. Rappaport, and M. Greenspan, "Generalized 4-points congruent sets for 3D registration," in *Proc. IEEE 2nd Int. Conf. 3D Vis.*, 2014, pp. 83–90.

[23] X. Ge, "Automatic markerless registration of point clouds with semantic-keypoint-based 4-points congruent sets," *ISPRS J. Photogrammetry Remote Sens.*, vol. 130, pp. 344–357, Aug. 2017.

[24] H. Wu, M. Scaioni, H. Li, N. Li, M. Lu, and C. Liu, "Feature-constrained registration of building point clouds acquired by terrestrial and airborne laser scanners," *J. Appl. Remote Sens.*, vol. 8, no. 1, Jul. 2014, Art. no. 083587.

[25] Y. Xu, R. Boerner, W. Yao, L. Hoegner, and U. Stilla, "Pairwise coarse registration of point clouds in urban scenes using voxel-based 4-planes congruent sets," *ISPRS J. Photogrammetry Remote Sens.*, vol. 151, pp. 106–123, May 2019.

[26] Y. Fu, Z. Li, F. Xiong, H. He, W. Wang, and Y. Deng, "Pairwise coarse registration of point clouds by traversing voxel-based 2-plane bases," *Int. J. Remote Sens.*, vol. 43, no. 13, pp. 5100–5123, Jul. 2022.

[27] S. M. Hasheminasab, T. Zhou, and A. Habib, "Linear feature-based image/LiDAR integration for a stockpile monitoring and reporting technology," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 16, pp. 2605–2623, 2023.

[28] W. Tao, X. Hua, Z. Chen, and P. Tian, "Fast and automatic registration of terrestrial point clouds using 2D line features," *Remote Sens.*, vol. 12, no. 8, Apr. 2020, Art. no. 1283.

[29] T. On Chan, D. D. Lichti, D. Belton, and H. L. Nguyen, "Automatic point cloud registration using a single octagonal lamp pole," *Photogrammetry Eng. Remote Sens.*, vol. 82, no. 4, pp. 257–269, Apr. 2016.

[30] Z. Cai, T.-J. Chin, A. P. Bustos, and K. Schindler, "Practical optimal registration of terrestrial LiDAR scan pairs," *ISPRS J. Photogrammetry Remote Sens.*, vol. 147, pp. 118–131, Jan. 2019.

[31] B. Yang, Z. Dong, F. Liang, and Y. Liu, "Automatic registration of large-scale urban scene point clouds based on semantic feature points," *ISPRS J. Photogrammetry Remote Sens.*, vol. 113, pp. 43–58, Mar. 2016.

[32] J. Yang, Y. Xiao, and Z. Cao, "Aligning 2.5 D scene fragments with distinctive local geometric features and voting-based correspondences," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 3, pp. 714–729, Mar. 2019.

[33] Z. Du et al., "MDCS with fully encoding the information of local shape description for 3D rigid data matching," *Image Vis. Comput.*, vol. 121, May 2022, Art. no. 104421.

[34] W. Tao, S. Xu, W. Huang, S. Hu, and M. Pang, "A distinctive binary descriptor and two-point RANSACWC for point cloud registration," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 16, pp. 7529–7542, 2023.

[35] Y. Wang and J. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3522–3531.

[36] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3D point cloud matching with smoothed densities," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5540–5549.

[37] P. Wei, L. Yan, H. Xie, and M. Huang, "Automatic coarse registration of point clouds using plane contour shape descriptor and topological graph voting," *Automat. Construction*, vol. 134, Feb. 2022, Art. no. 104055.

[38] Y. Tan, Y. Shi, Y. Li, and B. Xu, "Automatic registration method of multi-source point clouds based on building facades matching in urban scenes," *Photogrammetric Eng. Remote Sens.*, vol. 88, no. 12, pp. 767–782, Dec. 2022.

[39] X. Cheng, X. Cheng, Q. Li, and L. Ma, "Automatic registration of terrestrial and airborne point clouds using building outline features," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 2, pp. 628–638, Feb. 2018.

[40] W. Xu et al., "Feature curve-based registration for airborne LiDAR bathymetry point clouds," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 112, Aug. 2022, Art. no. 102883.

[41] F. Ghorbani, Y.-C. Chen, M. Hollaus, and N. Pfeifer, "A robust and automatic algorithm for TLS–ALS point cloud registration in forest environments based on tree locations," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 17, pp. 4015–4035, 2024.

[42] I. Sipiran and B. Bustos, "Harris 3D: A robust extension of the Harris operator for interest point detection on 3D meshes," *Vis. Comput.*, vol. 27, pp. 963–976, Nov. 2011.

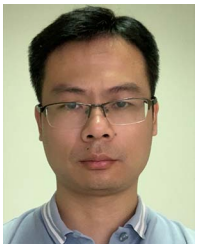
[43] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3D object recognition," in *Proc. IEEE 12th Int. Conf. Comput. Vis. Workshops*, 2009, pp. 689–696.

[44] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3D range scans taking into account object boundaries," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 2601–2608.

[45] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numer. Math.*, vol. 14, pp. 403–420, Apr. 1970.

[46] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 143–152.

[47] B. Zeisl, K. Köser, and M. Pollefeys, "Automatic registration of RGB-D scans via salient directions," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2808–2815.



Wuyong Tao received the master's degree in surveying and mapping from the East China University of Technology, Nanchang, China, in 2015, and the Ph.D. degree in geodesy from Wuhan University, Wuhan, China, in 2020.

He is currently a Lecturer with Nanchang University, Nanchang, China. From 2019 to 2020, he was a Visiting Ph.D. Student with the University of Calgary, Calgary, AB, Canada. He has authored or coauthored more than 40 research papers. He has been a Reviewer for IEEE TRANSACTIONS ON INTELLIGENT VEHICLES, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, *The Visual Computer*, *Remote Sensing*, *Survey Review*, *Journal of applied remote sensing*, etc. His research interests include point cloud registration, 3-D computer vision, and laser scanning.

Dr. Tao has been nominated as a Full Member in Sigma Xi (The Scientific Research Honor Society) from 2024.

Yansheng Xiao is currently working toward the bachelor's degree with the School of Mathematics and Computer Sciences, Nanchang University, Nanchang, China.

His research interests include point cloud registration and segmentation.

Ruisheng Wang (Senior Member, IEEE) received the B.Eng. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, China, the M.Sc.E. degree in geomatics engineering from the University of New Brunswick, Fredericton, NB, Canada, and the Ph.D. degree in electrical and computer engineering from McGill University, Montreal, QC, Canada.

In 2012, he joined the University of Calgary, where he is currently a Professor with the Department of Geomatics Engineering. Prior to that, he was an Industrial Researcher with HERE (formerly NAVTEQ), Chicago, IL, USA, in 2008. His research interests include mobile LiDAR data processing for next-generation map making and navigation.

Tieding Lu received the B.S. degree in geomatics and the M.S. degree in geodesy from the East China University of Technology, Nanchang, China, in 1997 and 2006, respectively. He received the Ph.D. degree in geodesy from Wuhan University, Wuhan, China, in 2010.

He is currently a Professor with the School of Surveying and Geoinformation Engineering, East China University of Technology. From 2011 to 2016, he was a Postdoctoral Researcher with Changan University, Xi'an, China. His research interests include GNSS navigation and position, 3-D laser scanning, and deformation monitoring.

Shaoping Xu received the M.S. degree in computer application from the China University of Geosciences, Wuhan, China, in 2004, and the Ph.D. degree in mechatronics engineering from the University of Nanchang, Nanchang, China, in 2010.

He is currently a Full Professor with the Department of Computer Science and Technology, School of Mathematics and Computer Sciences, Nanchang University, Nanchang, China. He has authored or coauthored more than 50 articles in journals and conference proceedings. His research interests include digital image processing and analysis, computer graphics, and virtual reality.