Hindawi
Complexity
Volume 2024, Article ID 9854101, 1 page
https://doi.org/10.1155/2024/9854101

Retraction

Retracted: A 3D Reconstruction Method Using Multisensor Fusion in Large-Scale Indoor Scenes

Complexity

Received 23 January 2024; Accepted 23 January 2024; Published 24 January 2024

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] P. Gu, F. Zhou, D. Yu, F. Wan, W. Wang, and B. Yu, "A 3D Reconstruction Method Using Multisensor Fusion in Large-Scale Indoor Scenes," *Complexity*, vol. 2020, Article ID 6973790, 14 pages, 2020.

Hindawi
Complexity
Volume 2020, Article ID 6973790, 14 pages
https://doi.org/10.1155/2020/6973790

Research Article

A 3D Reconstruction Method Using Multisensor Fusion in Large-Scale Indoor Scenes

Panlong Gu,[1] **Fengyu Zhou**,[1] **Dianguo Yu**,[2] **Fang Wan**,[1] **Wei Wang**,[3] **and Bangguo Yu**[1]

[1]*School of Control Science and Engineering, Shandong University, Jinan250061, Shandong, China*
[2]*Institute of Technology, Qu Fu Normal University, Jining 276825, Shandong, China*
[3]*Faculty of Computer Science and Technology, Qi Lu University of Technology, Jining 276825, Shandong, China*

Correspondence should be addressed to Fengyu Zhou; zhoufengyu@sdu.edu.cn

Received 23 April 2020; Revised 11 August 2020; Accepted 9 September 2020; Published 24 September 2020

Academic Editor: Zhihan Lv

RGBD camera-based VSLAM (Visual Simultaneous Localization and Mapping) algorithm is usually applied to assist robots with real-time mapping. However, due to the limited measuring principle, accuracy, and distance of the equipped camera, this algorithm has typical disadvantages in the large and dynamic scenes with complex lightings, such as poor mapping accuracy, easy loss of robot position, and much cost on computing resources. Regarding these issues, this paper proposes a new method of 3D interior construction, which combines laser radar and an RGBD camera. Meanwhile, it is developed based on the Cartographer laser SLAM algorithm. The proposed method mainly takes two steps. The first step is to do the 3D reconstruction using the Cartographer algorithm and RGBD camera. It firstly applies the Cartographer algorithm to calculate the pose of the RGBD camera and to generate a submap. Then, a real-time 3D point cloud generated by using the RGBD camera is inserted into the submap, and the real-time interior construction is finished. The second step is to improve Cartographer loop-closure quality by the visual loop-closure for the sake of correcting the generated map. Compared with traditional methods in large-scale indoor scenes, the proposed algorithm in this paper shows higher precision, faster speed, and stronger robustness in such contexts, especially with complex light and dynamic objects, respectively.

1. Introduction

In order to build intelligent applications for mobile robots in a large scene, autonomous and efficient navigation plays an increasingly important role. VSLAM, a technique with high-precision positioning and obstacle avoidance information, has been popular in the field of robot navigation. At present, VSLAM is usually applied on monocular camera, binocular camera, and depth camera RGBD. Among these three, the RGBD-based VSLAM algorithm has obvious advantages in terms of real-time accuracy. This is mainly because of the fact that the depth camera can input the depth image with accurate scale information to the computer directly. It not only overcomes the problem that the binocular camera needs to consume extra computer resources to obtain the depth image, but also solves the problem that the monocular camera cannot do dense construction during mapping, or has no scale information and poor drawing quality in mapping. Despite of advantages, some popular RGBD-based VSLAM algorithms, such as RGBD SLAM V2 algorithm [1], RTAB algorithm [2, 3], and DVO algorithm [4], still have typical limitations in large-scale indoor scenes, which mainly reflect in that dynamic objects will introduce additional errors to VSLAM or even course to operation failure. However, this would be impossible in real-time large scene reconstruction. For example, some inevitable changes, lights, people moving, and object moving can cause the loss of position and posture of mobile robots [5] and, in turn, lead to inaccurate mapping. In order to deal with some dynamic objects that may exist in large scenes, the current algorithm usually eliminates dynamic objects by adding object detection or image segmentation algorithm based on

deep learning in the system, such as Dynamic SLAM [6], Cluster VO [7], and Cluster SLAM [8] algorithm [9]. However, this will inevitably lead to a large consumption of computing resources, and some microdynamic objects are usually distributed near the camera. After removing these feature points, the accuracy of the camera odometer will also be affected. In addition, since the effective measuring distance of the RGBD depth camera (i.e., Kinect V2) is only 4.5 meters [5], it may result in a large cumulative error in the large scenes. Therefore, it is challenging to obtain a high-precision environmental map. Nevertheless, this weakness can be compensated by using a 2D laser range finder. For example, the commonly used SICK TIM 561 laser range finder has 270 degree field of view angle with 12 meter effective measuring distance. Cartographer [10], a laser SLAM algorithm based on graph optimization proposed by Google in 2016, is specifically used on such kind of a 2D laser range finder. It applies SPA [10, 11] (Sparse Pose Adjustment), as well as the branch and bound method [10, 12], to adjust the constructed map and reduces the accumulated error effectively. Hence, it is more suitable for large-scale scene mapping. However, this method still has some shortcomings. A typical weakness lies in its limitation of data that is obtained only from a certain level of the space. This limitation may lead to the loss of large amount of environmental information during the performance of mobile robots and can hamper their behaviors and control in turn.

According to the analysis mentioned above, based on the existing Cartographer algorithm, this paper proposes a new method for 3D mapping in large scene contexts, which integrates a laser range finder and RGBD depth camera. Its working principles mainly include three steps. Firstly, it inserts the 3D point cloud generated in real-time into the submap generated by the Cartographer algorithm for 3D reconstruction. Then, visual loopback is introduced to check the accuracy of laser loopback. It needs to run a visual loopback test on the back end of the computer (not in real-time), which will check the loopback results established by the Cartographer algorithm. If the two loopback results are significantly different, it will replace the laser loopback with the results obtained by the visual loopback. Finally, after the algorithm detects the loop-closure, it adjusts the pose of the point cloud in real-time, modifies the map, and inserts the 3D point cloud bound to the submap into the appropriate space position. Comparative experiment results show that the proposed algorithm is more accurate, real-time, and robust than the simple vision SLAM and laser SLAM algorithms.

The main contributions of this paper can be summarized into three aspects:

(i) The approach proposed in this paper about reconstruction in large dynamic scene implements the fusion of RGBD camera and laser range finder based on the Cartographer algorithm. Cartographer provides more robust odometer data for algorithm reconstruction, and an RGBD camera provides rich environment data for the algorithm to eliminate the point clouds and laser points of dynamic objects, so as to optimize the odometer calculation of Cartographer.

(ii) A novel approach to calculate the ground true value in large scene, which does not need extra instruments but simply laying the Apriltag on the ground instead. With the time and cost required in the experiment being greatly reduced.

(iii) Experimental results show that the proposed approach has achieved the robust state-of-the-art performance on the real large public scene.

The remainder of this paper is organized as follows: Section 2 presents the related work in the past few years about 3D reconstruction in dynamic scene and the Cartographer algorithm presented in this paper. Section 3 describes the sensor calibration method in our proposed approach. Section 4 introduces the new approach in detail. The experimental results in the real scene are given in Section 5. Conclusions are drawn in Section 6.

2. Related Work

This section focuses on the review of previous works in terms of the 3D reconstruction method of dynamic scale and the introduction of Cartographer algorithm, upon which our new method is developed.

2.1. 3D Reconstruction in Large Dynamic Scenes. To avoid generating errors while reconstructing, existing works addressing this problem are of three types.

The first category is based on dynamic object detection. In [9, 13], algorithms reduce the introduced error by removing the feature points in the image where the dynamic object is located and, then, uses the points in the remaining static objects to calculate the camera odometer data of the two frames.

The second category is based on the random consistency of established maps. In [14], the random consistency algorithm was proposed to evaluate the established map, which, in turn, removes the dynamic objects mistakenly inserted into the map.

The third category is based online and surface features. References [15, 16] proposed methods to calculate the pose change of the camera between two frames by line and surface features instead of point features.

Although these algorithms can solve the interference of moving objects in the scene, they will increase extra burden on computing and bring about disadvantageous effects on the accuracy and the real-time performance. Especially when the proportion of external interference data exceeds a certain threshold [17], no algorithm to remove noise or interference could help.

2.2. Cartographer Algorithm. The Cartographer algorithm [10], a cross-platform and multisensor fusion -based laser

SLAM algorithm proposed by Google, consists of local SLAM and global SLAM algorithms.

In the local SLAM algorithm, Cartographer calculates odometer information by fitting scan points and IMU data from newly inserted scan points. Calculating the position of the newly inserted scan point in the submap is the least square problem [10] which aims to maximize the probability of matching the scan point with the submap.

The submap generated by local SLAM is a part of the global map, which is presented in the form of probability grid. Each grid has a fixed odds value, which indicates the blocked probability of the grid. After the new scan points are inserted into the submap, a set of odds are recalculated, and the probability values p_{hits} or p_{miss} of the grid points represented by odds set is updated using probability formula (1), Reference [10], and the score of each point of the probability grid is calculated by formula (2). After sufficient scans have been inserted, the current submap will be updated and output.

$$\text{odd}(p) = \frac{p}{1-p}, \tag{1}$$

$$M_{\text{new}}(x) = \text{clamp}\left(\text{odd}^{-1}\left(\text{odds}\left(M_{\text{old}}(x)\right) \cdot \text{odds}\left(p_{\text{hits/misses}}\right)\right)\right). \tag{2}$$

In the global SLAM algorithm, Cartographer recorded each submap generated by the local SLAM algorithm and the scans data and their locations for loop-closure detection. During the movement of the robot, the branch and bound method [12] was used to establish loop-closure constraints, using SPA (Sparse Pose Adjustment) algorithm [11] to optimize the locations of all submaps.

With the assistance of global SLAM and local SLAM, the Cartographer algorithm has much better robustness, real-time performance, and accuracy in the real scene than the compared methods.

3. Sensor Calibration

In this paper, the large scene reconstruction algorithm integrates an RGBD depth camera and laser range finder. Due to the need to superimpose the point cloud data of the RGBD camera according to the laser range finder, odometer sensor calibration is the first and the necessary step to do. The proposed algorithm has two key components: (i) point cloud generation and (ii) registration of point cloud to the laser range finder coordinate system.

3.1. Point Cloud Generation. The RGBD camera to be used in this paper is Kinect v_2, and its depth camera and RGB camera are located in different positions of Kinect v_2. In order to obtain accurate environmental point cloud, it is necessary to calibrate the depth camera first.

Firstly, we can get the RGB camera's internal parameter matrix C_c, depth camera's internal parameter matrix C_d, and the external parameter matrix M of these two cameras by Zhang's calibration method [18], respectively.

Then, we assume that the points in the space are at the same distance from the center of the two cameras and substitute matrix C_c, C_d, and M in the derived equation as follows:

$$\begin{bmatrix} Dx_c \\ Dy_c \\ 1 \\ \frac{1}{d} \end{bmatrix} = C_c \cdot M \cdot C_d^{-1} \begin{bmatrix} Dx_d \\ Dy_d \\ 1 \\ \frac{1}{d} \end{bmatrix}, \tag{3}$$

where d is the pixel value at the depth coordinate (Dx_d, Dy_d) and denotes the depth of this point. By equation (3), point (Dx_d, Dy_d)on the depth image can be transformed to the RGB image coordinate system and recorded as (Dx_c, Dy_c).

Finally, this method is used to transform each point on the depth image to the RGB image coordinate. By superposing to display the depth image with the color image, as can be seen in Figure 1, the color image and the depth image are well coincided.

After calibration, the depth d of each point $P_{\text{img}} = (x_{\text{img}}, y_{\text{img}})$ of the RGB image can be obtained by using the corresponding pixel value in the calibrated depth image transformed, then the coordinate P_{img} and depth d are plugged into equations (4)–(6) [19, 20], and the position of this point $P_{\text{cam}} = [P_x, P_y, P_z]$is figured out in space.

$$P_z = d, \tag{4}$$

$$P_x = \frac{P_z}{f_x}\left(x_{\text{img}} - c_x\right), \tag{5}$$

$$P_y = \frac{P_z}{f_y}\left(y_{\text{img}} - c_y\right). \tag{6}$$

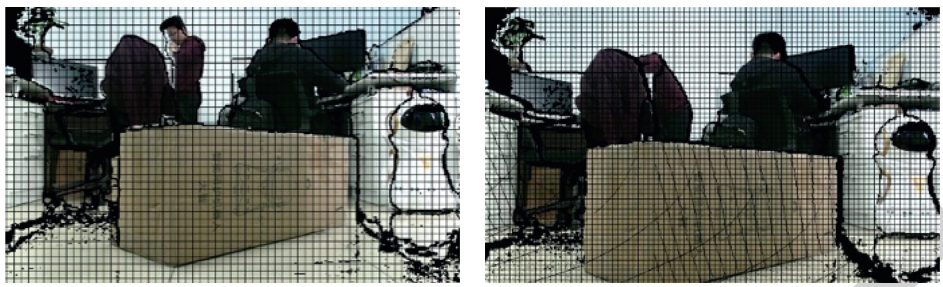
In the abovementioned equations, (f_x, f_y, c_x, c_y) expresses the parameter in the camera's internal parameter matrix focal length of the depth camera. The parameter represents the optical center coordinate and the product of the camera zoom factor and focal length, respectively.

3.2. Registration of Point Cloud to the Laser Range Finder Coordinate System. After obtaining the point cloud, in order to accurately insert the point cloud data into the grid map generated by Cartographer, the camera point cloud and laser range finder scanning points need to be calibrated. The position of the calibration plate relative to the robot is shown in Figure 2.

The point cloud in world coordinateP_{word} and its corresponding points in the RGBD camera coordinateP_{cam} system can be expressed by the following equation:

$$P_{\text{cam}} = RP_{\text{word}} + t. \tag{7}$$

The 3×3 rotation matrix R and 3-vector translation t represent the orientation and the position of the camera,



(a) (b)

Figure 1: Calibration results between RGB and depth of Kinect. (a) Camera data cannot be aligned before calibration. (b) After calibration, the depth image can be well-aligned with the color image.

Figure 2: Calibration is performed using a calibration plate affixed to a prominent raised surface.

respectively. Similarly, it can be deduced that each scanning point P_{laser} measured by the laser range finder in the laser coordinate system has a corresponding point P_{cam} in the camera coordinate system, and the transformation relationship between them can be expressed by the following equation:

$$P_{\text{laser}} = \Phi P_{\text{cam}} + \Delta. \tag{8}$$

In equation (8), Φis a 3×3 orthogonal matrix, representing the rotation relationship between the reference system of laser range finder and the world reference system.Δ is a translation 3-vector representing the relative displacement between the two reference systems.

In order to calibrate the laser range finder and RGBD camera, there are two steps to do. Firstly, assuming the calibration board is located in the plane of $Z = 0$ in the world coordinate system, the surface of the calibration plate identified in the camera image is transformed into a 3-vector matrix N [20]. The norm of the matrix is adopted to show the Euclidean distance from the camera phase center to the calibration board. According to the transformation relationship between the camera coordinate system and real coordinate system as expressed in equation (7), the vector matrix can be deduced by the following equation:

$$N = -R_3\left(R_3^T \times t\right). \tag{9}$$

Secondly, assuming that the plane detected by the 2D laser range finder is located at $Y = 0$ in the world coordinate, the scanning points recorded by the laser rangefinder on the chessboard are represented by the following equation:

$$\widehat{P}_{\text{laser}} = [X, Z, 1]^T. \tag{10}$$

Substituted $\widehat{P}_s$ into equation (8), the corresponding points of the scanning point in the camera reference can be obtained. When $\widehat{P}_{\text{laser}} = [X, Z, 1]^T$ is located in the calibration plane represented by the parameter N,

$$\|N\|^2 = N \cdot P. \tag{11}$$

Furthermore, equation (11) can be deduced as equation (12):

$$\|N\|^2 = N \cdot H\widehat{P}_s, \tag{12}$$

$$H = \Phi^{-1}\begin{bmatrix} 1 & 0 & -\Delta_1 \\ 0 & 0 & -\Delta_2 \\ 0 & 1 & -\Delta 3 \end{bmatrix}, \tag{13}$$

$$\Delta = \left[\Delta_1, \Delta_2, \Delta_3\right]^T. \tag{14}$$

After solving linear equation (12), the external parameter matrix from the RGBD camera to the laser range finder can be calculated. Then, we substituteΦandΔinto equation (8) and transfer each laser scanning point. The calibration between the s scanning point and the point cloud can be calibrated. The calibration effect is shown in Figure 3.

4. System Overview

Figure 4 is the overall architecture of the proposed algorithm in this paper, which is divided into two major parts: the Cartographer algorithm and 3D reconstruction.

The Cartographer algorithm consists of local SLAM and global SLAM algorithms and provides odometer information and submap information for 3D reconstruction. The local SLAM is used to fit the acceleration and angle data of the newly inserted laser scanning point and IMU (inertial measurement unit) compare with the laser scanning points of the previous frame in order to calculate the acceleration, angle data, and odometer information. Also, the pose of the generated submap representing the new area is continuously adjusted, to which the robot reaches. The global SLAM algorithm, or global optimization algorithm, applies the branch and bound method to perform closed-loop matching on the newly generated submap with all previous submaps and point set data.

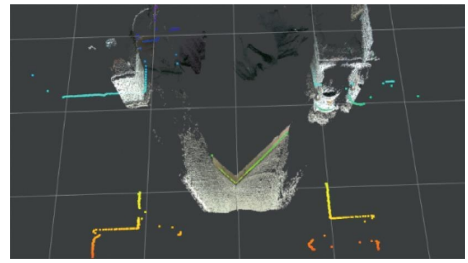


Figure 3: Calibration results between Kinect and the laser range finder.

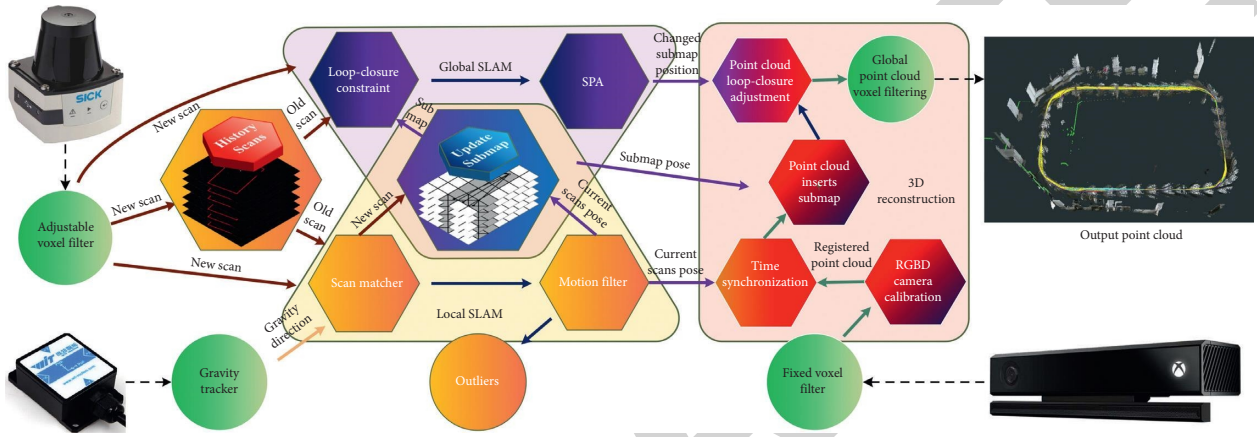


Figure 4: Overview of the 3d scenario construction system.

It uses SPA (Sparse Pose Adjustment) to adjust all previously generated submaps and scan point sets. After the match completes, the construction of 2D map is performed.

In the part of 3D reconstruction, after receiving the data of the depth camera, firstly, the corresponding point cloud information is calculated, and the position and pose are transformed and bound to the corresponding submap space. Along with the adjustment of the submap pose by Cartographer, the point cloud data bound to the submap is adjusted in real-time simultaneously. When superposition submap generates a 2D map, superposition point cloud generates a 3D map and publishes point cloud information to the robot in real-time.

In more detail, the proposed algorithm is introduced from three major functional parts which are described as follows.

4.1. Filling Space Point Cloud into the Submap. The proposed algorithm builds the map by filling the space point cloud into the submap. In order to accurately superimpose the 3D point cloud information that is inserted into each submap, the point cloud information needs to be bound with each submap. Then, a method with less resource consumption needs to be found to update the adjusted point cloud. Along with robot moving forward and continuous insertion of laser scanning points, the Cartographer algorithm will estimate the robot's mobile posture among scanning points and output the robot pose at the frequency of 70 Hz as shown in the following equation:

$$\delta = \left[\delta_x, \delta_y, \delta_z, \delta_{\text{roll}}, \delta_{\text{pitch}}, \delta_{\text{yaw}}\right]^T. \tag{15}$$

When the robot reaches a new area, the algorithm will generate a submap and publish all the previously established submaps' information at the frequency of 5 Hz, and the pose of submap can be expressed by the following equation:

$$\phi = \left[\phi_x, \phi_y, \phi_z, \phi_{\text{roll}}, \phi_{\text{pitch}}, \phi_{\text{yaw}}\right]^T. \tag{16}$$

At the same time, the Kinect depth camera on the robot publishes depth images and RGB images at a frequency of 25 Hz, and the IMU publishes robot pose information at a frequency of 200 Hz.

In ROS, computer uses time stamp to record time information when data are published. The time stamps of the robot pose, submap, RGBD camera image, and IMU are represented by T_δ, T_Φ, T_{RGBD}, and T_{IMU}, respectively.

In order to ensure the normal operation of the algorithm, different sensors need to be synchronized to ensure the sensor data input to the system is of synchronization. Time synchronization includes hardware synchronization and software synchronization. Hardware synchronization is achieved using the BIOS time to screen multiple sensors' data, but cannot be achieved among different devices. Software synchronization caters for this need. This type of synchronization uses software to select data closest to the base time from the sensor data that are input to the same host and completes the time binding of sensor data. Main steps are taken as follows:

The first step is to synchronize the pose of the RGBD camera and the pose of robot, which is to find the pose data closest to T_{RGBD} in the prestored 2000 pose data of the robot, as the process shown in Algorithm 1.

Similarly, such synchronization can be achieved for the RGBD camera and the submap, by which the point cloud data closest to T_{Φ} can be selected.

In the process of binding RGBD image data, robot pose information, and IMU data, the maximum theoretical time difference is 7 ms while the average time difference in actual tests is less than 1 ms. If the robot runs at the speed of 2 m/s, the distance error generated is of the millimeter level. In the process of synchronizing submap and RGBD point cloud, the time difference in theory is 20 ms if the release frequency of the RGBD image is at 25 Hz. In real running, the measured time difference is about 100 ms due to the time consumption on point cloud generation. When the robot is running at 2 m/s, the center of the submap and the point cloud at a distance deviation of 20 cm will be generated if the point cloud is directly inserted into the submap. The abovementioned error data generated in time synchronization are shown in Table 1.

In order to eliminate the abovementioned spatial dislocation, it is necessary to transfer the point cloud to the reference frame of its corresponding submap before the point cloud is filled into the submap space. To achieve this, it is, firstly, needed to calculate the spatial position difference between the robot pose and the submap pose while generating the point cloud and, then, make use of equation (22) to transform the change of the rotation angle between two reference systems into rotation matrix R[21], which can be calculated by equations (17)–(21).

$$\tau = \left[\delta_x - \phi_x, \delta_y - \phi_y, \delta_z - \phi_z\right], \tag{17}$$

$$\gamma = \left[\delta_{\mathrm{roll}} - \phi_{\mathrm{roll}}, \delta_{\mathrm{pitch}} - \phi_{\mathrm{pitch}}, \delta_{\mathrm{yaw}} - \phi_{\mathrm{yaw}}\right], \tag{18}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma_{\mathrm{roll}} & -\sin\gamma_{\mathrm{roll}} \\ 0 & \sin\gamma_{\mathrm{roll}} & \cos\gamma_{\mathrm{roll}} \end{bmatrix}, \tag{19}$$

$$R_y = \begin{bmatrix} \cos\gamma_{\mathrm{pitch}} & 0 & \sin\gamma_{\mathrm{pitch}} \\ 0 & 1 & 0 \\ -\sin\gamma_{\mathrm{pitch}} & 0 & \cos\gamma_{\mathrm{pitch}} \end{bmatrix}, \tag{20}$$

$$R_z = \begin{bmatrix} \cos\gamma_{\mathrm{yaw}} & -\sin\gamma_{\mathrm{yaw}} & 0 \\ \sin\gamma_{\mathrm{yaw}} & \cos\gamma_{\mathrm{yaw}} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{21}$$

$$R = R_z R_y R_x, \tag{22}$$

$$\mathrm{Trans} = \begin{bmatrix} R & \tau \\ 0^T & 1 \end{bmatrix}. \tag{23}$$

After obtaining the rotation matrix and translation matrix, R and τ will be taken into equation (23). According to equation (24), all points in the original point cloud, denoted by P, can be inserted into the submap, denoted by P_{submap}.

$$P_{\mathrm{submap}} = P \cdot \mathrm{Trans}. \tag{24}$$

Procedure Time synchronized
$T_{\mathrm{base}} \longleftarrow T_{\mathrm{RGBD}}$
data[2000] ⟵ Sensor_data(aim_sensor)
mini_time ⟵ max_timeinterval()
for i in 2000, **do**
  $T_{\mathrm{aim}} \longleftarrow$ data $[i]$
  if abs($-T_{\mathrm{aim}}$) < threshold, **then**
    minitime ⟵ abs(−)
    mini_I ⟵ i
  end if
end for
aimsensordata ⟵ data[i]
return aimsensordata

Algorithm 1: Time synchronized.

Table 1: Errors occurred in time synchronization.

Objects that synchronize with RGBD	IMU	Submap
Theoretical time error (ms)	7	20
Theoretical distance error (cm)	1.4	4
Actual time error (ms)	<1	>100
Actual distance error (cm)	<0.2	>20

4.2. Point Cloud Superposition. The pose of the submap needs to be adjusted continuously in the process of loop optimization. To facilitate the adjustment, on top of the transformation mentioned in part A, it still needs to transfer the pose of the point cloud to its corresponding spatial position. This requires a translation of the point cloud using equation (25) based on the submap's location information.

$$\tau_{\mathrm{submap}} = \begin{bmatrix} \phi_x & \phi_y & \phi_z \end{bmatrix}^T. \tag{25}$$

After that, the translated point cloud will be superimposed on the total point cloud. The Cartographer algorithm constantly generates submaps when running.

With the superposition of submaps, the bound point clouds are also superposed. In the process of generating the 2D map, the joint of 3D point cloud can be completed. Figure 5 gives an example. Along with the robot moving forward, the Cartographer algorithm generates submap14 and submap15. When the two submaps are superposed, the corresponding point clouds of these two submaps are also superposed.

4.3. Dynamic Object Removal. In the process of 3D reconstruction, there are usually a large number of dynamic objects in the scene, and it will not only affect the accuracy of odometer but also introduce dynamic noise points into map.

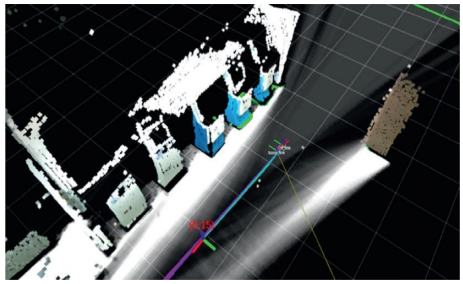


Figure 5: Binding of point cloud data to the submap.

The proposed algorithm based on the odometer calculated by the Cartographer algorithm has higher accuracy than VSLAM, in the dynamic scene.

However, the more accurate odometer can ensure the accuracy of the map, but it cannot avoid the residual dynamic point cloud being inserted into the map in the process of reconstruction. Therefore, in order to solve this problem, this section proposes a method to eliminate the residual dynamic point cloud with the least computing resources in the reconstruction.

In this algorithm, we calculate the relative pose of two submaps and their corresponding point clouds relative to submap (16) and (17). Then, the view intersection area of two cloud points is calculated based on this pose.

When the algorithm inserts a new point cloud into the total point cloud using formula (26), it first opens a window around each point of the new point cloud, runs, "AND" operates with the points in the total point cloud so that the points that exist in both point clouds are preserved.

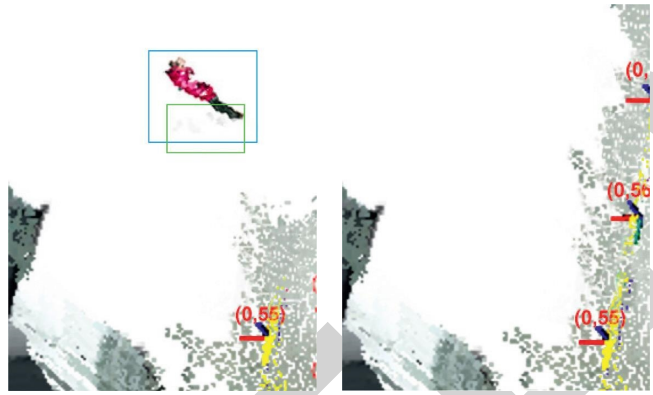
In this way, the residual point cloud in the map can be eliminated, as shown in Figure 6.

4.4. Adjustment of Point Cloud Sloop-Closure. Along with the continuous insertion of new scanning points, the global SLAM algorithm will open a window around the scanning points for detection with the help of the branch and bound method.

However, the laser range finder can only obtains the data of a certain plane in space, which is limited for the Cartographer algorithm to calculate loop-closure precisely or establish a loop-closure in the position where laser features are scarce.

We designed to use the ORB feature points to extract point cloud feature in every submap. When a new submap is generated by local SLAM, our algorithm will detect the loop-closure between the current submap's point cloud and the previous submap' point cloud. When the visual loop and the laser loop are established at the same time, we use the loop information calculated by the visual loop to replace the laser loop information. When visual loop-closure was established without laser loop-closure established, visual loop-closure will force the Cartographer algorithm to establish loop-closure between submaps.

During the course of loop-closure detection, no matter whether the algorithm detects a loop-closure or not, SPA will be performed to adjust the pose of the submap, in which the



(a) (b)

Figure 6: Dynamic object removal. (a) Before removal, the foot of the woman in red is detected, leaving a gray occupying grid and point cloud in the generated submap. (b) After removal, the woman in red walked through the machine and not appeared on the point cloud again. The Cartographer algorithm used formula (2) to update the occupied grid of basket position in (a) to the non-occupied state, and the proposed algorithm in section C removes the point cloud of the woman in red.

loop-closure occurs, from ϕ_{old} to ϕ_{new}. Through calculating the position difference of the old and the new submap, we can obtain the following equations:

$$\tau_{\text{loop}} = \left[\phi_{\text{new}x} - \phi_{\text{old}x}, \phi_{\text{new}y} - \phi_{\text{old}y}, \phi_{\text{new}z} - \phi_{\text{old}z}\right], \tag{26}$$

$$\gamma_{\text{loop}} = \left[f_{\text{newroll}} - f_{\text{oldroll}}, \delta_{\text{newpitch}} - f_{\text{oldpitch}}, \delta_{\text{newyaw}} - f_{\text{oldyaw}}\right]. \tag{27}$$

Taking equations (26) and (27) into equations (22) and (23), the Euclidean transformation matrix between the new and the old submaps can be calculated, which is recorded as $\text{Trans}_{\text{loop}}$. Then, the loop-closure adjustment of the point cloud can be made by multiplying the $\text{Trans}_{\text{loop}}$ and the point cloud bound by the submap that the pose has changed based on formula (24) and, then, translating and superposing the new generated point cloud according to formulas (25) and (26). Whenever a new submap is inserted, SPA will optimize and adjust the pose ϕ of all generated submaps. Due to this reason, it is easy to jam the program and affect the display effect by resuperimposition of the point cloud. Moreover, after the superimposed point cloud is filtered by the voxel, the point cloud bound by the submap that the pose has been changed cannot be directly subtracted from the superimposed point cloud.

To solve the abovementioned problem, our algorithm divides the point cloud into several regions based on a certain threshold, and only the point clouds in the regions, where the submaps adjust their pose is located, are superimposed. The above algorithms are shown as (Algorithm 2)

5. Experiment

5.1. Experiment Platform and Design. In this paper, a Meconium-wheel mobile robot equipped with a Kinect V2

```
Procedure Regional point cloud stack
Submap ⟵ SubmapMessage
for i in size of (Submap), do
  if position(Submap[i]) is changed and Submap[i] is not new Submap then
    RegionalPointcloudUnfinish [IndexInRegional(i)]
      ⟵Equation (25) (RegionalPointcloudUnfinish [IndexInRegional(i)], NewPosition(Submap[i]))
  end if
  else if Submap[i] is new Submap then
    NewPointcloud ⟵ TimeSynchronized(Submap)
    TransformedPointCloud ⟵ Equation (23) (NewPointcloud)
    ShiftedPointcloud ⟵ Equation (25) (TransformedPointCloud, position(Submap[i]))
    RegionalPointcloudUnfinish.pushback(ShiftedPointcloud)
    ResidualPointcloud ⟵ ShiftedPointcloud + ResidualPointcloud
    if sizeof(RegionalPointcloudUnfinish) ≥ threshold then
      for j in range(threshold)
        RegionAllPointcloud ⟵ RegionalPointcloudUnfinish[j] + RegionAllPointcloud
      end for
      RegionPointcloud.pushback(RegionAllPointcloud)
      RegionalPointcloudUnfinish.clear()
      ResidualPointcloud.clear()
      return RegionAllPointcloud
    else if
      return RegionAllPointcloud + ResidualPointcloud
    end if
  end if
end if
```

Algorithm 2: Regional point cloud stack.

depth camera, SICK TIM561 laser range finder, and IMU is chosen as the experiment platform, as shown in Figure 7, to perform various tests and comparative experiments in large indoor scenes.

In the experiment, we choose a relatively accurate range between 0.2 m and 5.5 m of Kinect depth data for calculation and down sample the point cloud using a 0.03 m grid-size voxel filter. After the calculation, the point cloud is sent to RVIZ for display. Also, we define the starting point of the mapping work as the origin of the world coordinate system, that is, the position of No. 0 submap.

Figure 7: Experiment platform: the Meconium-wheel mobile robot.

In addition, in order to verify the robustness and accuracy of the algorithm based on its real-time performance, we select RTAB algorithm for comparison because of its better accuracy and robustness than other popular algorithms, such as DVO SLAM algorithm and RGB-D SLAM V2 algorithm.

5.2. Comparison of Algorithm Robustness. In order to verify the robustness of the proposed algorithm, we select the outer of the corridor hall (an area of 27 m × 16 m) on the 2nd floor of the main building at our campus as the experimental scene, as shown in Figure 8(a). In the course of mapping, the robot moves around the selected area at a speed of 2 m/s for one and a half laps.

As shown in Figure 8(b), due to a glass wall at the entrance of the hall, there is serious ambient light interference. Under this circumstance, the RGBD camera will lose most of its depth information, and such lost can exert a great impact on RTAB algorithm's performance due to its sensitivity on visual information.

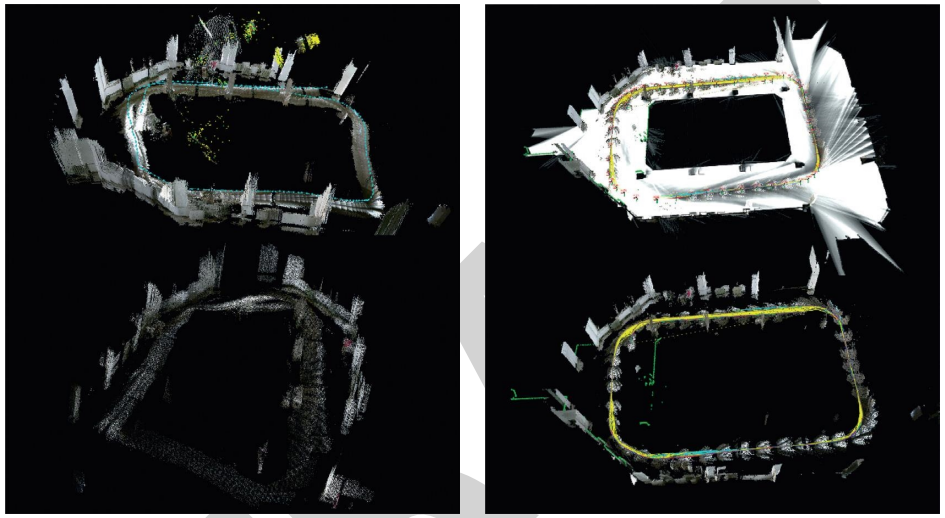
In many mapping experiments, RTAB algorithm is unable to complete the 3D mapping. The failures are all caused by the loss of location at the entrance. Despite that only a few mappings are successful, ghosting dislocation and distortion appear in many places, as shown in Figure 9(a).

Since RGBD's RTAB algorithm can lead to large error, our algorithm introduces a Cartographer algorithm, which reduces this error and improves the robustness, as shown in Figure 9(b).

5.3. Comparison of Mapping Accuracy. The lobby on the 1st floor of the selected building, which has sunlight interference

(a) (b) (c)

Figure 8: Second floor cloister of the main building. (a) Corridor hall on the 2nd floor. In the scene (b) with sunlight interference, depth camera (c) cannot obtain depth information.



(a) (b)

Figure 9: Comparison of mapping results on the second floor. (a) RTAB algorithm result. (b) Our result.

at night, sufficient indoor light source, and rich environmental characteristics, is selected as the experimental scene.

Its outer dimension is also 27 m × 16 m. The richer layout of the lobby, including several self-service recharge machines on both sides of the hall, many tables, and chairs, is the ideal place for comparative experiments, as shown in Figure 10. In addition, the scene has ideal conditions because there is no strong outdoor infrared interference.

The robot moves around the selected area of the first floor at the speed of 2 m/s for one and a half laps, and the constructed 3D maps by our algorithm and RTAB algorithm are shown in Figures 11(a) and 11(b), respectively. It can be seen that both algorithms can complete the mapping in an ideal environment, without serious loss or damage to important environmental information.

Figures 11(c) and 11(d) are enlarged views of the 3D map constructed by our algorithm and RTAB algorithm, respectively. It can be seen that the map constructed by RTAB in Figure 11(d) has an obvious dislocation, while the mapping effect by our algorithm shown in Figure 11(c) is more accurate without significant dislocation.

A basic criterion of 3D mapping is no obvious error in point cloud superposition. For this reason, we measured and compared the errors of the constructed 3D maps by both algorithms. Firstly, we use the laser range finder and meter ruler to measure the six specific locations in the field to obtain the actual size of the map, as shown in Figure 12.

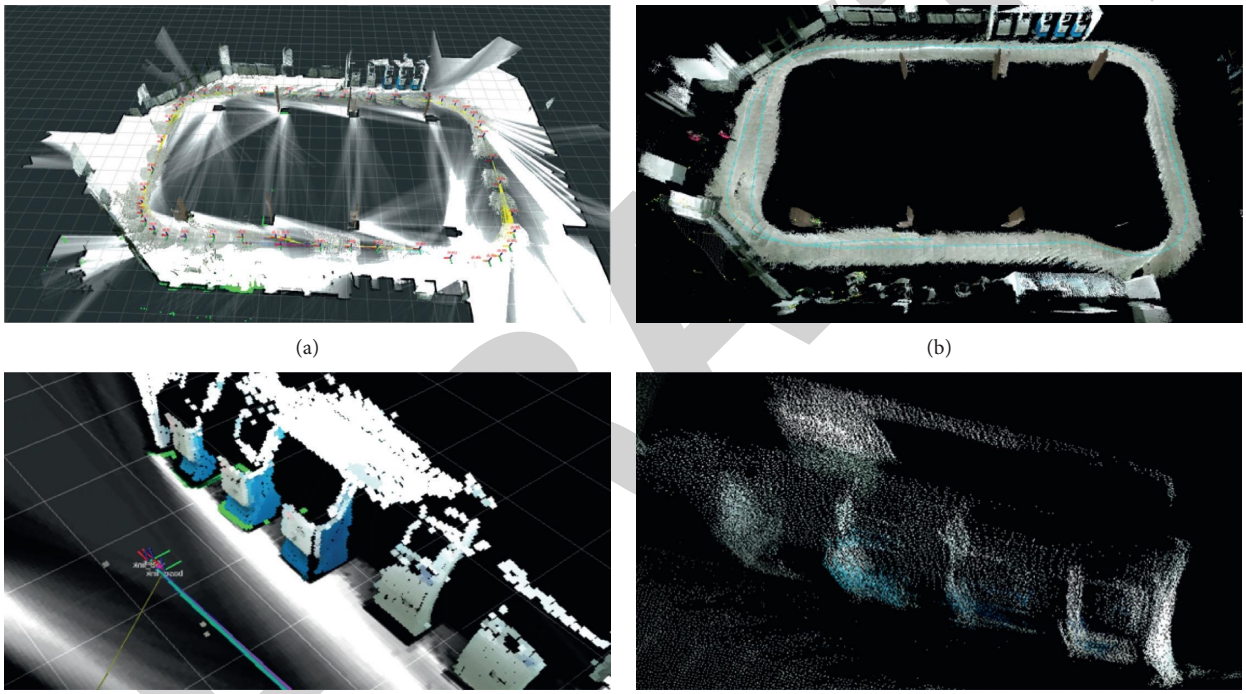
In that, six numbers marked on the given map are used to represent the features of the selected six positions, which are (1) the length of self-service machine of campus card, (2) the horizontal distance between two bearing pillars of the main building hall, (3) the distance between the two walls at the entrance of the main building hall elevator, (4) the vertical distance between the main building hall's bearing columns, (5) the longest vertical distance of the main building hall, and (6) the longest horizontal distance of the main building hall.

Then, RVIZ is used to measure the length of the map constructed by two algorithms, and the absolute error and relative error of them are calculated, respectively, compared with the real value of the map. The comparison results are shown in Table 2.

As seen from Table 2, in terms of the relative errors, all relative errors of the maps constructed by our algorithm are less than 1%, which is much lower than that of the RTAB algorithm. In addition, the distribution of relative errors in

(a) (b)

Figure 10: Environment of the lobby on the first floor. (a) The lobby on the 1st floor. (b) Self-service machine.



(a) (b)

(c) (d)

Figure 11: Comparison of the mapping effect on the first floor. (a), (b) The mapping effect of our proposed algorithm and RTAB, respectively. (c), (d) Enlarged details of the algorithm results.

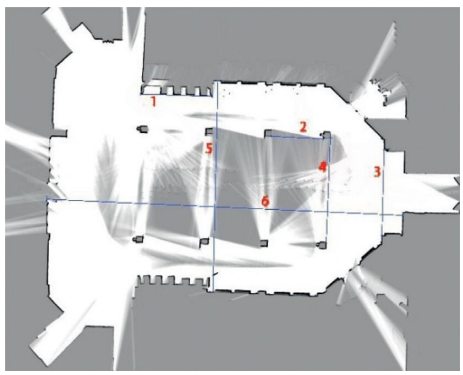


Figure 12: Mapping precision comparison.

Table 2: Measurement results of 3D mapping reconstruction.

No.	Algorithm	GT (m)	Measurement (m)	Absolute error (m)	Relative error (%)
1	OURS	5.51	5.53	0.02	0.363
	RTAB		5.41	0.10	1.815
2	OURS	4.03	4.01	0.02	0.496
	RTAB		3.94	0.09	2.234
3	OURS	6.52	6.50	0.02	0.307
	RTAB		6.36	0.16	2.454
4	OURS	7.90	7.89	0.01	0.127
	RTAB		7.03	0.87	11.013
5	OURS	16.05	15.98	0.07	0.436
	RTAB		15.56	0.49	3.053
6	OURS	27.04	27.08	0.04	0.148
	RTAB		26.04	1.00	3.698

our algorithm is more even and is not affected by the sparse sensor feature points. However, the relative errors of RTAB algorithm are smaller in vertical scenes such as the position No. 3, No. 4, and No. 5 due to rich features, but larger in the case of No. 6 scene due to fewer feature points.

In terms of the absolute error, the results of our method are of relative balance, while the accuracy of RTAB algorithm increases along with the increase of map size and, in particular, reaches to 1 m in scene 6.

Based on the abovementioned comparisons, our algorithm is of significant advantages in a large scene.

5.4. Comparison Experiment on Track and Loop Accuracy

5.4.1. Calculating the Ground True Value of Robot Motion. Experiments on the comparison of the track and the loop-closure precision require comparing the error between the ground truth value of robot motion and the errors of odometers in our method and the RTAB algorithm.

In the experiment, the ground truth value of robot motion is calculated by the position of Apriltag code on the floor tiles (60 cm × 60 cm per floor tile), as shown in Figure 13.

Firstly, the location of Apriltag code in the field is prestored in the industrial personal computer equipped by the robot using the following equation:

$$L_n = (x, y), \quad n = 1, 2, 3, \ldots. \tag{28}$$

Then, by identifying the Apriltag code laid on the ground, the position of the robot relative to Apriltag code is obtained, captured by the following equation:

$$L_A = (x, y). \tag{29}$$

Next, the position of the robot in the test site can be obtained by the following equationfd30:

$$L_R = L_A + L_n. \tag{30}$$

According to the experiment of Wang and Olson [22], the robot's recognition accuracy of the Apriltag code can be up to 100% within 6 meters. Therefore, the ground true value of the robot's track can be calculated by the Apriltag algorithm.

Figure 13: Apriltag on the ground.

5.4.2. Contrast Experiments on Track and Loop-Closure Accuracy. In order to compare the odometer accuracy and the loop-closure ability of two algorithms (our method and RTAB algorithm), we designed the following experiment. The remote-control robot first runs one lap along a rectangular track in the hall on the 1st floor, and continues running a certain distance after the robot passes the starting point in order to allow the algorithm to run loop-closure sufficiently. Finally, the robot returns to the starting point again.

Figures 14(a) and 14(b) show the comparison of the odometer. Here, the red asterisk and green asterisk in (a) represent the end point and the start point of the robot running, respectively, and (b) gives an enlarged figure showing that the proposed algorithm has much higher odometer accuracy in large-scale indoor scenes than that of RTAB algorithm.

To be more specific, as can be seen from Figure 14(b), the distance difference value between the start point and the end point of our method is 15.1 cm, while the data of RTAB is 65.307 cm.

Figures 14(c) and 14(d) show the comparison between the two algorithms after full loop-closure, and the result is that the loop-closure accuracy of our method is better than that of RTAB algorithm. As can be seen in (d) that shows more detailed information about the results, after loop-closure, most accumulated errors can be eliminated. In this case, the distance error in our algorithm is only 0.6 cm, while the result of RTAB algorithm is 1 cm.

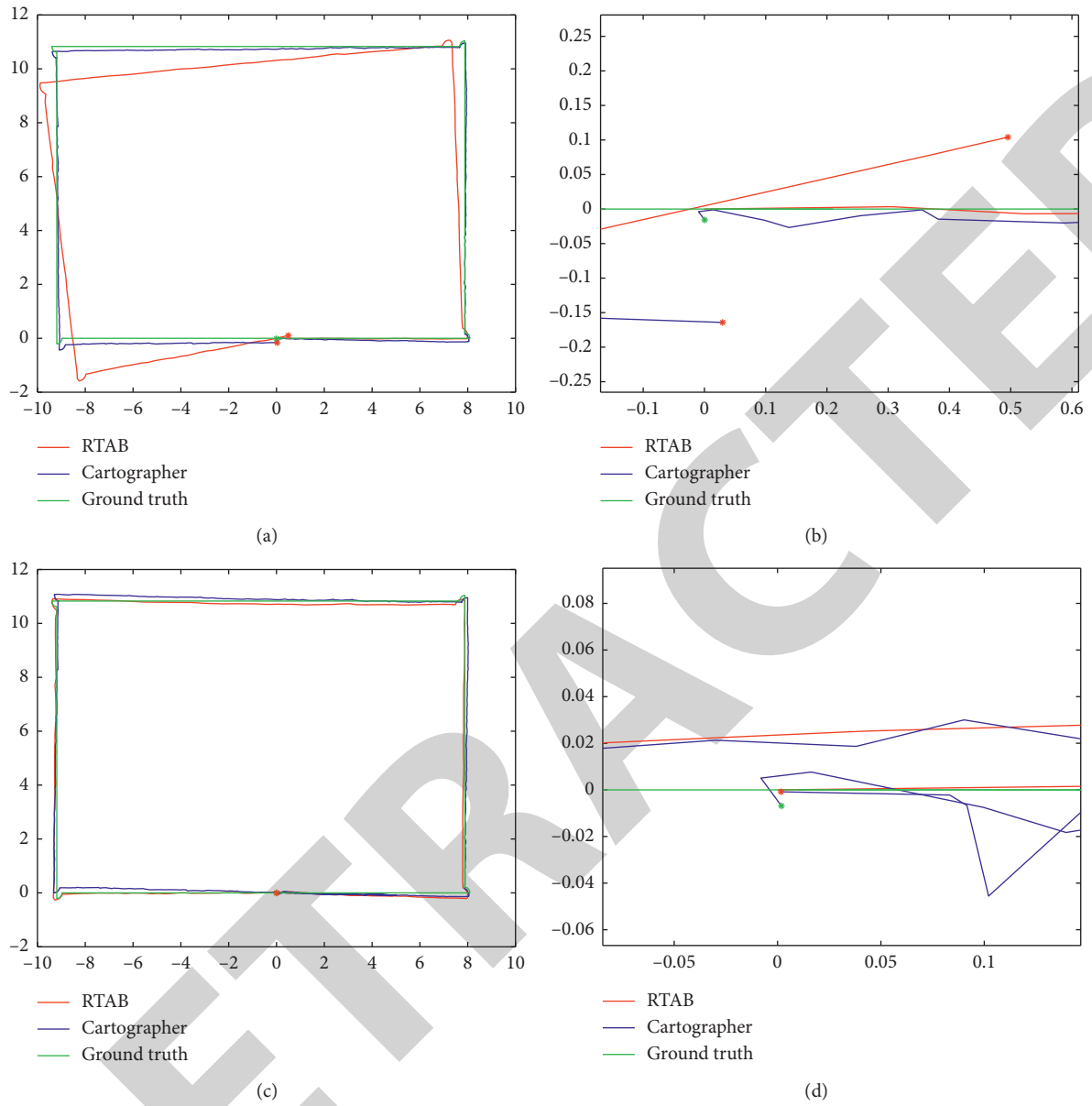


Figure 14: Trajectory accuracy contrast. (a) Accuracy comparison of the odometer. (b) Comparison of odometer accuracy after robot returns to origin. (c) Accuracy comparison of loop-closure. (d) Comparison of loop-closure accuracy after robot returns to origin. Unit: meter.

5.5. Algorithm Contrast Experiment in Real-Time. In order to ensure the integrity of the established map, in principle, our algorithm and RTAB algorithm are designed to process the current frame after all previously stored frames are processed. However, this design may not guarantee the real-time of map construction in reality.

In order to compare the real-time of two algorithms, in this paper, the environment information recorded by the robot in the real scene is packaged as rosbag function in ROS (robot operating system). During the verification phase, it is ensured that the sensor data received by the two algorithms are consistent through data packet playback in ROS.

Since the data types of the two algorithms are different, it is very troublesome and error-prone to record and process each frame of the data produced by each algorithm. Instead, in the comparative experiment, we use the total time spent on mapping to compare the real-time performance of them. The experimental results are shown in Table 3.

It can be seen that the real-time of this algorithm is much higher than that of RTAB algorithm.

The computer configuration and environment used in this algorithm experiment are as follows:

CPU: I5-5200U 2.2GHZ

Ubuntu: 16.04

ROS: Kinetic

Table 3: The comparison of total time spent by the algorithms.

Experiment site	1st floor (s)	2nd floor (s)
Total rosbag duration	241	213
Time for our algorithm	241	213
Time for RTAB algorithm	506	421

6. Conclusions

In the 3D reconstruction of a large scene, although the interference of moving objects and uneven sunlight can be eliminated, it needs to consume a huge amount of extra computing resources and, more importantly, is not conducive to the improvement of the real-time and accuracy of the algorithm. In this paper, a new algorithm is presented for 3D reconstruction in large scenes, which integrates the 3D point cloud generated by the Kinect camera and odometer information output by the Cartographer algorithm. Also, it is of high robustness and high accuracy. In addition, we also propose a new method applying Apriltag code to calculate the ground truth value of robot motion in large scenes. This method effectively solves the problem of the insufficient measurement range of the optical tracker in large scenes. The results of the real tests conducted in two selected actual scenes show that the newly proposed algorithm has higher real-time, robustness, and accuracy than the traditional RGBD camera-based 3D reconstruction algorithms.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB1302400, the National Natural Science Foundation of China under Grant nos. 61773242 and 61803227, and the Major Agricultural Applied Technological Innovation Projects of Shandong Province under Grant SD2019NJ014.

References

[1] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[2] M. Labbé and F. Michaud, "RTAB-map as an open source lidar and visual SLAM library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[3] M. Labbé and F. Michaud, "Online global loop-closure detection for large-scale multi-session graph-based SLAM," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Chicago, IL, USA, November 2014.

[4] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proceedings of the 2013 IEEE/RSJ international conference on Intelligent Robots and Systems (IROS)*, January 2013.

[5] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[6] L. Xiao, J. Wang, X. Qiu et al., "Dynamic-SLAM: semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robotics & Autonomous Systems*, vol. 17, pp. 1–16, 2019.

[7] J. Huang, S. Yang, T. J. Mu, and S.-M. Hu, "ClusterVO: clustering moving instances and estimating visual odometry for self and surroundings," 2020, https://arxiv.org/abs/2003.12980.

[8] J. Huang, S. Yang, Z. Zhao et al., "ClusterSLAM: a SLAM backend for simultaneous rigid body clustering and motion estimation," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, South Korea, February 2019.

[9] L. Zhao, Z. Liu, J. Chen, W. Cai, W. Wang, and L. Zeng, "A compatible framework for RGB-D SLAM in dynamic scenes," *IEEE Access*, vol. 7, pp. 75604–75614, 2019.

[10] W. Hess, D. Kohler, H. Rapp et al., "Real-time loop-closure in 2D LIDAR SLAM," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*, June 2016.

[11] K. Konolige, G. Grisetti, K. Rainer et al., "Efficient sparse pose adjustment for 2D mapping," in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Taipei, Taiwan, October 2010.

[12] J. Clausen, *Branch and Bound Algorithms-Principles and Examples*, pp. 1–30, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark, 1999.

[13] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song, "Semantic SLAM based on object detection and improved octomap," *IEEE Access*, vol. 6, pp. 75545–75559, 2018.

[14] C. Bibby and I. Reid, "Simultaneous localization and mapping in dynamic environments (SLAMIDE) with reversible data association," in *Proceedings of the Robotics: Science & Systems III, June, Georgia Institute of Technology*, Atlanta, GA, USA, 2007.

[15] H. Li, Z. Hu, and X. Chen, "PLP-SLAM: a visual SLAM method based on point-line-plane feature fusion," *Robot*, vol. 39, no. 2, pp. 214–220+229, 2017.

[16] Y. Liu, D. Yang, J. Li, Y. Gu, J. Pi, and X. Zhang, "Stereo visual-inertial SLAM with points and lines," *IEEE Access*, vol. 6, pp. 69381–69392, 2018.

[17] D. Zou and P. Tan, "CoSLAM: collaborative visual SLAM in dynamic environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 354–366, 2012.

[18] Y. Liu and T. Li, "Reaserch of the improvement of Zhang's camera calibration method," *Optical Technique*, vol. 40, no. 6, pp. 565–570, 2014.

[19] X. Jing, J.-L. Gou, X.-M. Ma, K. Huang, D. Liu, and Y.-M. Zhang, "A large viewing angle 3-dimensional V-SLAM algorithm with a kinect-based mobile robot system," *Robot*, vol. 36, no. 5, pp. 560–568, 2014.

[20] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems*, Sendai, Japan, February 2005.

[21] X. Gao and T. Zhang, *Visual SLAM 14: From Theory to Practice*, pp. 41–45, China Industry and Information Technology Publishing Group, Electronic Industry Press, Beijing, China, 2017.

[22] J. Wang and E. Olson, "Apriltag 2: efficient and robust fiducial detection," in *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.