remote sensing

MDPI

Article

A New RGB-D SLAM Method with Moving Object Detection for Dynamic Indoor Scenes

Runzhi Wang [1,2], Wenhui Wan [1], Yongkang Wang [3] and Kaichang Di [1,*]

1 State Key Laboratory of Remote Sensing Science, Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, No. 20A, Datun Road, Chaoyang District, Beijing 100101, China; wangrz@radi.ac.cn (R.W.); wanwh@radi.ac.cn (W.W.)
2 College of Resources and Environment, University of Chinese Academy of Sciences, Beijing 100049, China
3 School of Environment Science and Spatial Informatics, China University of Mining and Technology, Xuzhou 221116, China; ts16160030a3@cumt.edu.cn
* Correspondence: dikc@radi.ac.cn; Tel.: +86-10-64868229

Received: 21 April 2019; Accepted: 11 May 2019; Published: 14 May 2019

Abstract: Simultaneous localization and mapping (SLAM) methods based on an RGB-D camera have been studied and used in robot navigation and perception. So far, most such SLAM methods have been applied to a static environment. However, these methods are incapable of avoiding the drift errors caused by moving objects such as pedestrians, which limits their practical performance in real-world applications. In this paper, a new RGB-D SLAM with moving object detection for dynamic indoor scenes is proposed. The proposed detection method for moving objects is based on mathematical models and geometric constraints, and it can be incorporated into the SLAM process as a data filtering process. In order to verify the proposed method, we conducted sufficient experiments on the public TUM RGB-D dataset and a sequence image dataset from our Kinect V1 camera; both were acquired in common dynamic indoor scenes. The detailed experimental results of our improved RGB-D SLAM were summarized and demonstrate its effectiveness in dynamic indoor scenes.

Keywords: RGB-D SLAM; dynamic indoor scenes; Kinect; moving object detection

1. Introduction

Simultaneous localization and mapping (SLAM) technology has become a fundamental prerequisite in many robotic applications [1,2]. Because of its ability to conduct navigation and perception simultaneously in an unknown environment, SLAM has attracted the attention of many scholars, and has gradually become a research hotspot over the past decades [3]. Current SLAM technology mainly includes two types of systems, distinguished by different data acquisition devices. The first type is based on light detection and ranging (LiDAR) sensors [4]. LiDAR-based SLAM is expensive, and mainly used in driverless cars. The second type, visual SLAM [5] with imaging sensor, has the advantages of low price and small volume; it can provide more abundant texture information for motion estimation. Owing to the ability of autonomous localization and quick environmental perception, visual SLAM plays an important role in global positioning system (GPS)-denied environments, for instance, indoor scenes [6,7].

Depending on the number of sensors, visual SLAM can be classified into monocular-camera- [8–11], stereo-camera- [12–14], and multiple-camera- [15,16] based versions. Monocular camera based visual SLAM cannot obtain scale information of the environment, so it is usually combined with inertial measurement units (IMUs) to obtain three dimensional (3D) information in the scene [17]. As for stereo-camera- and multiple-camera-based SLAM systems, they can obtain 3D coordinates of feature points by photogrammetry. However, extracting feature points and stereo matching takes a lot of time.

Subsequently, many scholars have tried to improve the efficiency of SLAM systems by using new algorithms for feature extraction and matching, such as Speeded-Up Robust Features (SURF) [18] and oriented FAST and rotated BRIEF (ORB) [19].

As a new type of depth sensor, an RGB-D camera, such as Microsoft Kinect, has the advantages of directly obtaining 3D coordinates like a LiDAR sensor, and providing texture information as an imaging sensor. Due to its low power consumption and small volume, RGB-D cameras are widely used as data acquisition devices on current robots. Meanwhile, effective RGB-D-based SLAM algorithms have been proposed [20,21]. Some of these algorithms are based on sparse features, such as ORB-SLAM2 [22], which is a representative RGB-D SLAM system. These SLAM systems use the correspondence between current and previous frames to estimate cameras' ego-motion iteratively. These correspondences can be established by matching feature points, lines, and even planes. Dense-point-cloud-matching-based algorithms have also been proposed, such as the method proposed in literature [23]. These methods are based on the assumption that the grayscale values of corresponding areas are invariant, and adopt an energy minimization function for iterative calculation of the camera's six degree of freedom (DoFs) motion.

However, most SLAM methods are based on the assumption that the environment is static. In other words, the geometric distribution of objects in the scene is assumed to be stationary in the process. These methods are unable to distinguish the features in static and dynamic objects, which leads to the deterioration of SLAM systems because of erroneous data association and faults in motion estimation. Therefore, the question of how to build a robust SLAM system in a dynamic environment has attracted much attention from scholars in recent years.

The first remarkable SLAM system to combine detection and tracking of moving objects method was proposed by Wang et al. [24]. They used a laser rangefinder to obtain the data and verify their algorithm in urban outdoor environments. After that, one of the basic ideas of SLAM in a dynamic environment is to identify the moving objects and then eliminate their associated data. By means of dense scene flow representation, the visual SLAM system proposed by Alcantarilla et al. [25] can detect moving objects within a distance of about five meters. However, it has distance limitation in moving object detection, and misjudges static points as dynamic ones in some non-artificial and poor texture scenes. The dense RGB-D SLAM proposed by Wang et al. [26] combines moving object segmentation with dense SLAM. It merges regions of which the predominant parts share the same fundamental matrices across the whole SLAM process into a group. Based on the assumption that the largest group corresponds to the static environment, it then removes the largest group and gets the areas of moving objects. Another dense 3D SLAM, proposed by Bakkay et al. [27], assumes a constant motion during a fixed time interval. Those pixels, of which the absolute velocities are higher than the preset threshold, are taken as seed segmentation of moving objects. Starting from these seeds, they use a region growing algorithm to identify dynamic areas. The motion removal approach proposed by Sun et al. [28] roughly determines the moving areas by subtracting the corresponding pixels of the continuous frames. They then apply the maximum-a-posterior (MAP) to precisely moving object determination. This approach is integrated into the front end of dense visual odometry (DVO), and tested in TUM RGB-D datasets [29]. However, only one moving cluster can be segmented, and the performance is not as good as original DVO in a lowly dynamic environment. Recently, with the rapid development of deep learning algorithms, there are some SLAM systems combined with deep learning methods to judge moving objects. Yang et al. [30] proposed an RGB-D SLAM integrated with the Faster R-CNN algorithm [31]. It detects the rectangular areas of humans through Faster R-CNN and removes the feature points in these rectangular areas and the corresponding data, so as to obtain robust results. Zhong et al. proposed a Detect-SLAM [32], which is based on ORB-SLAM and a deep detector to determine and remove dynamic objects. The DRE-SLAM proposed by Yang et al. [33] combines the YOLOv3 algorithm [34] with the multi-view constraint to identify dynamic pixels on moving objects.

Another idea of SLAM in a dynamic environment is to incorporate new, robust data processing mathematical methods into the process of the SLAM system, and then reduce the impact of moving objects. Moratuwage et al. used the random finite set (RFS) model to express features and measurement results in the current sensor field of view (FOV) [35]. They then used a Bayes recursion to estimate the static feature map and obtain the dynamic feature locations. The CoSLAM proposed by Zou et al. [36] studies the problem of visual SLAM in dynamic environments with multiple cameras. This SLAM system uses the re-projection error to distinguish static and dynamic feature points. It works robustly in both highly dynamic environments and static environments, however, the data is processed offline and it has a large drift error in dynamic experiments. The DVO algorithm, proposed by Kerl et al. [37], estimates the camera's ego-motion by considering the probabilistic model in the error function, which reduces the influence of large residuals and thus obtains robust results. Lee et al. [38] used a pose graph to represent robot poses, and false constraints of the pose graph that were associated with moving objects were pruned according to an error metric based on the grouped nodes. The correct results were then obtained by re-optimizing the new pose graph. In addition to detecting moving objects, there are methods that attempt to find static backgrounds or features to estimate the camera's ego-motion. The background-model-based dense-visual-odometry (BaMVO) algorithm proposed by Kim et al. [39] estimates the background model from depth scenes and the ego-motion of the camera through eliminating the influence of moving objects. The RGB-D SLAM proposed by Li et al. [40] uses foreground edge points to estimate camera's ego-motion, and static weights, which indicate the probability that an edge point is static, are combined with intensity assisted iterative closest point (IAICP) algorithm in order to reduce the effect of dynamic moving objects on the motion estimation. Since this method requires depth edges, it is more suitable for geometry rich environments where depth edges abound than the texture-poor environments.

In this paper, a new RGB-D SLAM method with moving object detection for dynamic indoor scenes is presented. The rest of the paper is organized as follows. Section 2 illustrates the proposed RGB-D SLAM in three parts of the flowchart. In Section 3, abounding experiments are reported to analyze the performance of our RGB-D SLAM in common dynamic indoor scenes. Conclusions and future work are discussed in the last section.

2. Methodology

Figure 1 illustrates the proposed RGB-D SLAM in a flowchart, which consists of the following parts: (1) input data and pre-processing; (2) moving object detection and elimination; (3) camera pose estimation. Detailed processes of the algorithms and models are given in the following sub-sections.

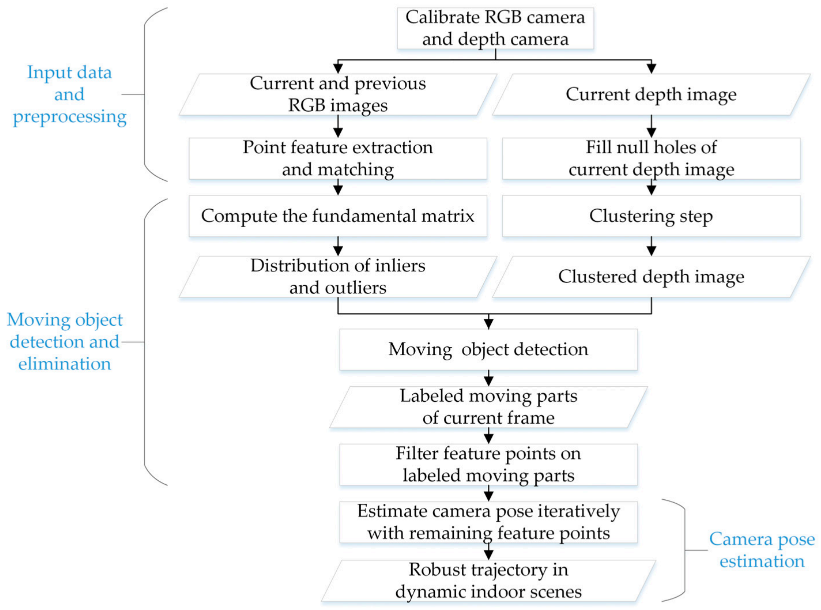
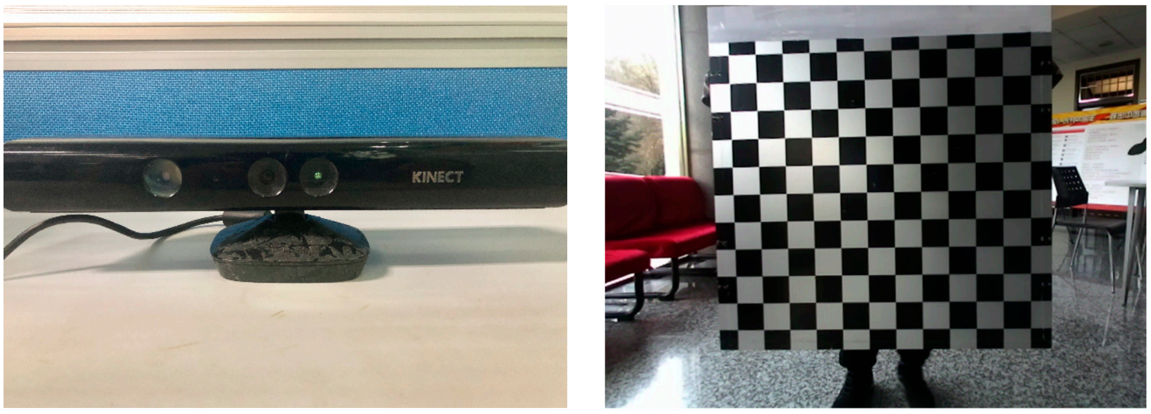

Input data and preprocessing
Calibrate RGB camera and depth camera
Current and previous RGB images
Current depth image
Point feature extraction and matching
Fill null holes of current depth image
Moving object detection and elimination
Compute the fundamental matrix
Clustering step
Distribution of inliers and outliers
Clustered depth image
Moving object detection
Labeled moving parts of current frame
Filter feature points on labeled moving parts
Estimate camera pose iteratively with remaining feature points
Robust trajectory in dynamic indoor scenes
Camera pose estimation

Figure 1. Flowchart of the proposed moving object detection based RGB-D SLAM.

2.1. Input Data and Preprocessing

In this part, we mainly introduce the image data used in our RGB-D SLAM and the pre-processing steps. In order to test the proposed method, we recorded RGB-D sequence images at 640×480 resolution with a Microsoft Kinect V1 camera, shown in Figure 2a. The camera was calibrated first, in order to use these sequence images in later high-accuracy processing. Although the RGB-D camera was adjusted in production, it did not satisfy the requirements of the experiment. A flat calibration board, as shown in Figure 2b, was used to calibrate the RGB camera and depth camera. We captured infrared images, RGB images, and depth images of the flat calibration board at different distances and directions. We then used a MATLAB-based calibration toolbox [41] to calculate the calibration parameters of the RGB camera and depth camera. The internal parameters and lens distortion coefficients are shown in Table 1, and the external parameters are shown in Table 2. Using these parameters, we can map the depth image to the RGB image.



(**a**) (**b**)

Figure 2. RGB-D camera and flat calibration board used in data acquisition and camera calibration. (**a**) Microsoft Kinect V1 sensor used in data acquisition; (**b**) flat calibration board used in camera calibration.

Table 1. Internal parameters and distortion coefficients of the RGB camera and depth camera.

Camera	f_x	f_y	c_x	c_y	k_1	k_2	k_3	p_1	p_2
RGB	584.35	584.33	317.97	252.80	−0.10585	0.27096	0.0	0.00504	−0.00166
Depth	519.95	519.55	315.82	238.71	0.04810	−0.19281	0.0	0.00458	−0.00014

[1] The unit of f_x, f_y, c_x, and c_y is the pixel.

Table 2. External parameters of the RGB camera and depth camera.

Rotation Angles (°)	−0.00079	−0.00084	−0.00541
Translation Vector (mm)	−25.59983	0.16700	−0.40571

[1] The unit of rotation angles is the degree and the unit of translation vector is the mm.

In order to calculate the camera pose and detect moving objects, we then needed to input the current RGB image, depth image, and previous RGB image. These two RGB images, the current frame and the previous frame, were used for the extraction and matching of point features. In addition to the RGB images, we also needed to use the current depth image. However, due to the influence of optical conditions and imaging modes, some areas of invalid holes existed in the depth images.

Figure 3a shows an original depth image of TUM RGB-D datasets. Owing to the reflection of the computer screen and the optical conditions of some areas, there are invalid holes in Figure 3a. Since the existence of these holes would weaken the effect of next clustering step, we needed to fill them in the pre-processing. Here, we used an algorithm based on morphological reconstruction [42] to fill these holes. As can be seen from Figure 3b, several obvious holes were filled in the depth image. The purpose of this is to better cluster in the subsequent steps, because these invalid holes are sometimes marked as a separate category by the clustering process and may mislead the judgement of moving objects.

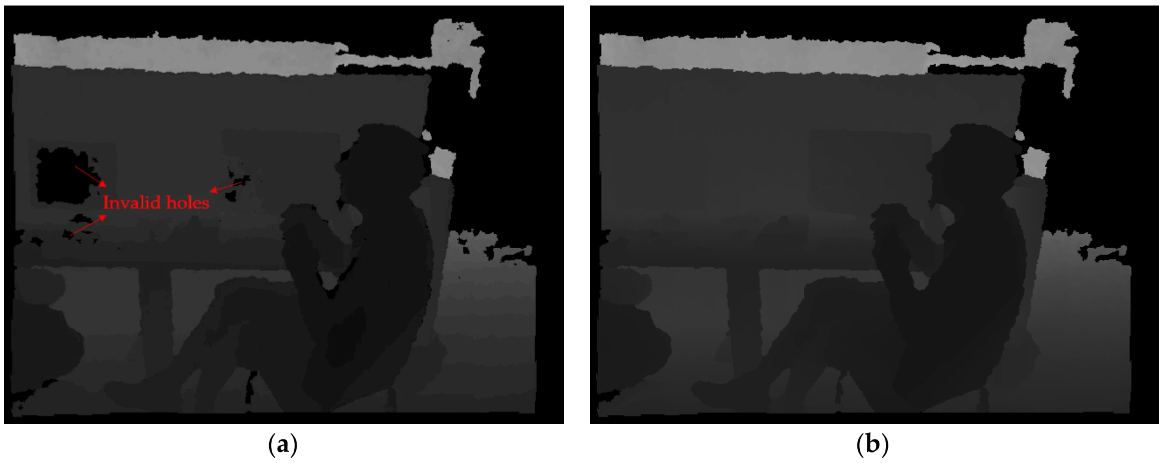


Figure 3. Original depth image and filled depth image. (**a**) There were some no data holes in the original depth image; (**b**) filled depth image. No data holes in the depth image were filled after this step.

2.2. Moving Object Detection and Elimination

In this sub-section, we introduce the principle and pipeline of our moving object detection method using example images taken from the TUM RGB-D dataset. Our RGB-D SLAM takes the proposed moving object detection method as a data processing front-end. This approach is based on two assumptions. The first one is that enough feature points, i.e., more than twenty pairs of matching points, exist in the static environment. These feature points are used to calculate the camera's pose. Common indoor scenes have many artificial objects, which can basically meet this assumption. The other one is that the parallax between two consecutive frames of the static environment captured by the RGB-D camera is small. The parallax here refers to the difference between the image coordinates of the same feature point on different frames. The reason for this assumption is that we can better find out the inconsistency of feature points caused by the motion of the moving objects in this situation, and then use our method to detect and eliminate the outliers. Large parallax between two consecutive frames will result in tracking loss. Commonly used RGB-D cameras, such as the Microsoft Kinect V1, can provide a high rate of 30 frames per second (FPS), which ensures that the parallax between two consecutive frames is sufficiently small (i.e., less than ten pixels). Therefore, these two assumptions can be satisfied when tested in an indoor environment using a common RGB-D sensor.

The first part of our moving object detection method is shown in Figure 4. We used previous and current RGB images for the extraction and matching of point features. The corresponding feature points are connected with yellow lines. As can be seen from the picture (Figure 4a), dense feature points were extracted from the plaid shirt of the right experimenter, and some points were extracted from the torso of the left experimenter wearing a black T-shirt. Feature points on moving objects will cause errors in the later estimation of camera pose, so we need to eliminate them. Before judging the moving areas, we need to cluster the depth image. In Figure 4b, the k-means clustering algorithm was used to cluster the filled current depth image. The number of clusters was set to ten in this example. We can see that the two experimenters in the clustered filled depth image are almost divided into a single category, and distinguished from the static environment.

Figure 5a shows the position of feature points on the current RGB images. From Figure 5a, we can see that feature points are spread among moving objects and static environments. We then mapped the positions of these feature points onto the clustered depth map, as Figure 5b shows. We can now count the distribution of feature points in the c-th cluster areas. Note that we have not yet excluded outliers in this step, so we use a statistic of the number and percentage of all feature points, N^c_{ini} and P^c_{ini}.

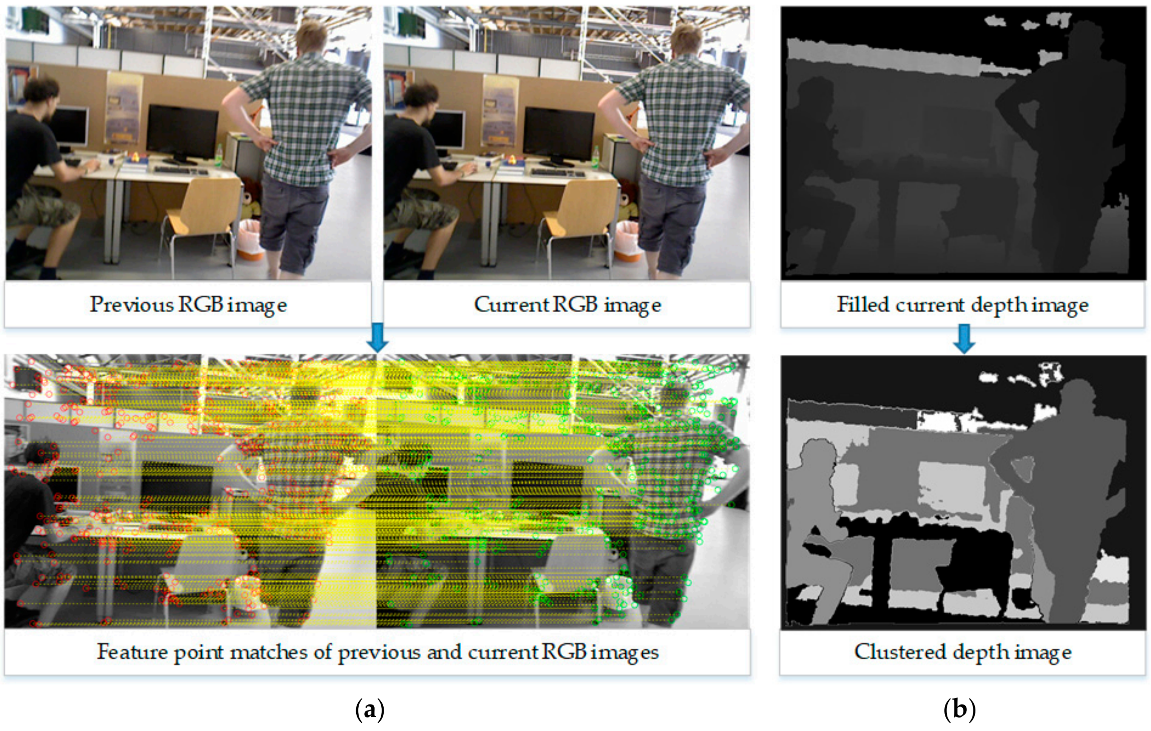

Previous RGB image
Current RGB image
Filled current depth image
Feature point matches of previous and current RGB images
Clustered depth image
(a)
(b)

Figure 4. The first step of the moving object detection method. (**a**) Extraction and matching of point features; (**b**) clustering filled depth image. Different gray values in the clustered depth image represent different clusters.

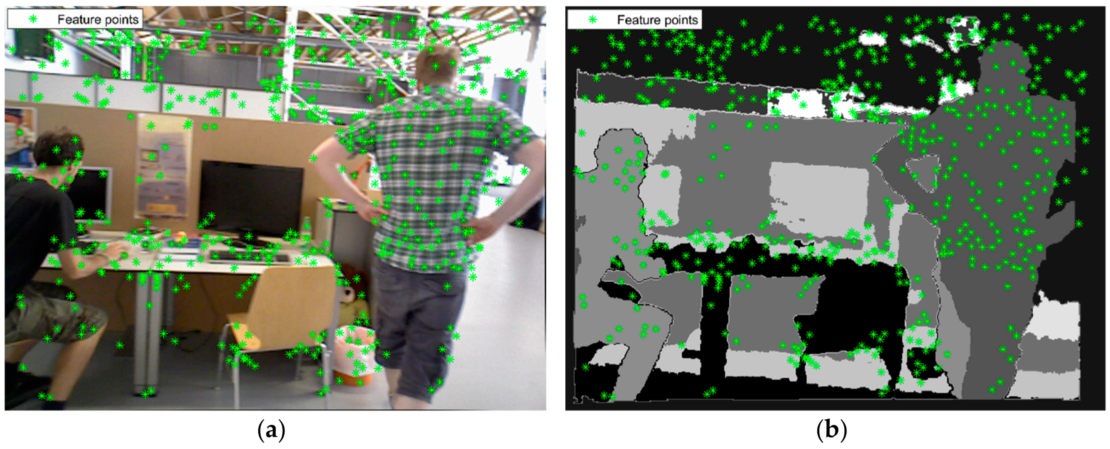


Figure 5. The second step of the moving object detection method. (**a**) The position of feature points on the current RGB images; (**b**) mapping the positions of feature points onto the clustered depth map. The star symbols represent the feature points. Different gray values represent different clusters.

We then used the fundamental matrix constraint [43] to identify inconsistency of feature points caused by the motion of the moving objects, and then eliminate the mismatched points. The fundamental matrix $\mathbf{F}_{3\times3}$ describes the relationship between the point $m_{3\times1}$ and its corresponding epipolar line l'_m, which can be expressed as follows:

$$l'_m = \mathbf{F}_{3\times3} \cdot m_{3\times1}. \tag{1}$$

Here, $m_{3\times1}$ is a homogeneous coordinate by padding with one in the third row. Since the corresponding point $m'_{3\times1}$ of point $m_{3\times1}$ is on the epipolar line l'_m, there must be:

$${m'_{3\times1}}^{\mathrm{T}} \cdot \mathbf{F}_{3\times3} \cdot m_{3\times1} = 0. \tag{2}$$

That is, if the point m of the first image is obtained, its corresponding point on the second image will be constrained on the epipolar line l'_m, as Figure 6 shows.

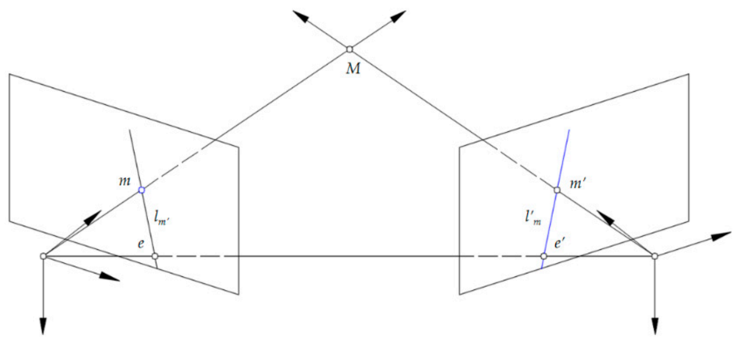


Figure 6. Illustration of the fundamental matrix constraint.

Eight points are randomly selected to calculate a fundamental matrix $\mathbf{F}^i$ in the i-th iteration. We can then compute the distance d^i_j of the j-th point according to $\mathbf{F}^i$:

$$d^i_j = \left(m'^{\mathrm{T}}_j \cdot \mathbf{F}^i \cdot m_j\right)^2. \tag{3}$$

Sampson distance weight is used to divide inliers and outliers to improve the robustness of the algorithm. It can be expressed as follows:

$$w^i_j = \frac{1}{\left(\mathbf{F}^i \cdot m_j\right)_1^{\;2} + \left(\mathbf{F}^i \cdot m_j\right)_2^{\;2} + \left(\mathbf{F}^{i\mathrm{T}} \cdot m'_j\right)_1^{\;2} + \left(\mathbf{F}^{i\mathrm{T}} \cdot m'_j\right)_2^{\;2}}. \tag{4}$$

We can then get the weighted distance:

$$wd^i_j = w^i_j \cdot d^i_j. \tag{5}$$

We took wd^i as the array of all distances in the i-th iteration and computed the median of wd^i. After reaching the maximum number of iterations, the least median of squares method [44] was used to select the best fundamental matrix with the smallest median. The smallest median was taken as the threshold. If the distance wd^i_j of the j-th point was less than the threshold, the j-th point was divided as an inlier, otherwise it was divided as an outlier.

In the third step, we used the fundamental matrix constraint mentioned above to determine the inliers and outliers. We then mapped the positions of these inliers and outliers onto the clustered depth map, as Figure 7a shows. Here, the distribution of inliers in the c-th cluster area can be counted. Note that we excluded outliers in this step, so we employed the number N^c_{first} and percentage P^c_{first} here as the statistical properties of the first fundamental matrix constraint. In this process, the eliminated outliers contain a small number of mismatching points and some points located in moving parts. As is shown in Figure 7a, the number of outliers removed from moving cluster areas is larger than that in the static environment. This is because feature points on a moving object are more inconsistent than points on a static area. Next, for the remaining inliers, the fundamental matrix constraint was used again to further eliminate the outliers. The positions of the inliers and outliers on the clustered depth map obtained this time are shown in Figure 7b, and the distribution of inliers in the c-th cluster area can be counted. Note that we have excluded outliers again, so we call the number N^c_{second} and percentage P^c_{second} here the statistical properties of the second fundamental matrix constraint. As is shown in Figure 7, since the number of feature points removed in the moving cluster areas is more than that in the static areas, the proportion of points in the moving areas will decrease.

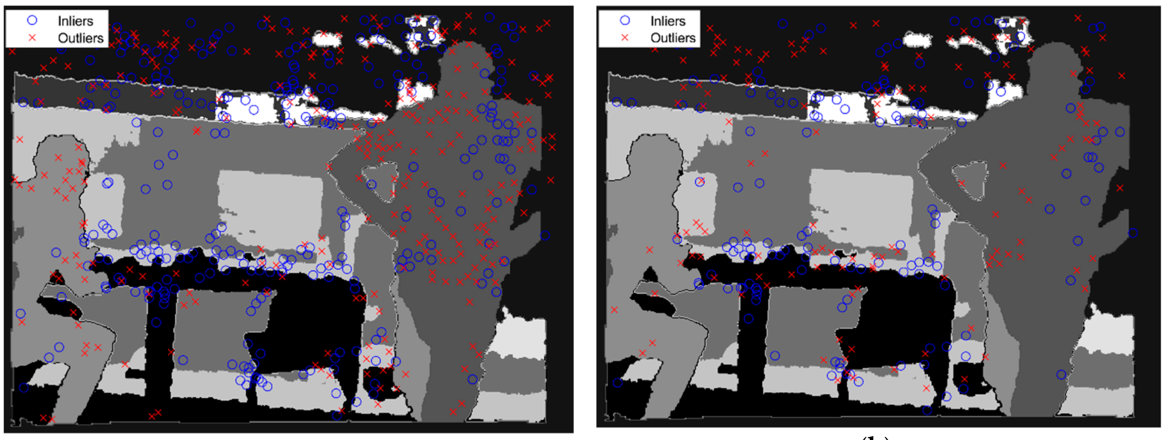


Figure 7. The third step of the moving object detection method. (**a**) The positions of the inliers (blue circle marks) and outliers (red x marks) on the clustered depth map after the first fundamental matrix constraint; (**b**) the positions of the inliers (blue circle marks) and outliers (red x marks) on the clustered depth map after the second fundamental matrix constraint. Different gray values represent different cluster areas.

The last step of our moving object detection method is to input the statistical characteristics obtained from the previous three steps into the moving objects judgment model designed by us. If a cluster area satisfies our model, it will be judged as a moving objects area. Otherwise it will be judged as a static area. The model consists of three modules, and the first module of our model can be expressed as follows:

$$\text{Module 1}: \ (N^c_{ini} - N^c_{first})/N^c_{ini} \times 100 > T_{reduce}. \tag{6}$$

T_{reduce} is the threshold that we set to find the cluster areas, where a large number of outliers were removed after the first fundamental matrix constraint. However, if a cluster area has a few feature points at the beginning, it will also meet T_{reduce}, even if a small number of outliers are removed. Therefore, we needed to set up the second module, which can be expressed as follows:

$$\text{Module 2}: \ P^c_{ini} > T_{ini}. \tag{7}$$

This threshold T_{ini} is to reduce the interference of small areas with a few feature points, which is easy to satisfy the first module. Normally, because the number of outliers removed from moving cluster areas is larger than that from the static cluster areas, the proportion of points in the moving cluster areas will decrease and the proportion of points in the static cluster areas will increase. Based on this, we set up the third module:

$$\text{Module 3}: \ (N^c_{ini} - N^c_{second}) > 0. \tag{8}$$

If the c-th cluster area satisfies all modules of our moving objects judgement model, the area will be judged to be a moving object. The judgement result of this example is shown in Figure 8a, and the result is a binary image. We can then use this binary image to filter the remaining feature points. As can be seen from Figure 8b, feature points on the moving persons have been eliminated, and the remaining feature points are basically located in the static environment.

In this part, we have illustrated the steps of our moving object detection method through example data from the TUM RGB-D dataset, and the final result in Figure 8 also shows the effectiveness of our method in detecting moving objects. Our RGB-D SLAM takes the proposed moving object detection method as a data filtering process, and estimates the camera pose more robustly and accurately.

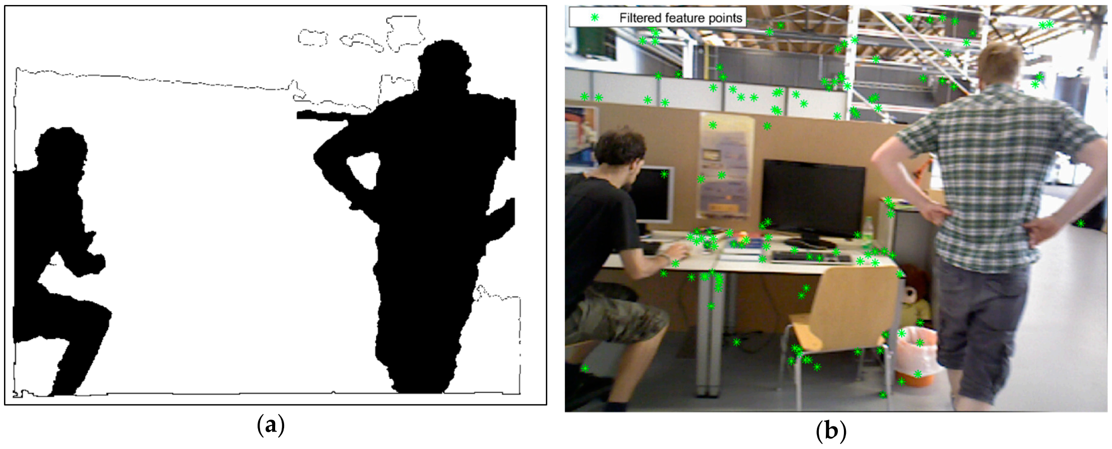


Figure 8. The final step of the moving object detection method. (**a**) The moving object judgement result of this example; (**b**) the remaining feature points on the current RGB image. The black areas in the binary image represent moving objects detected by our method. The star symbols in the RGB image represent the remaining feature points.

2.3. Camera Pose Estimation

After moving object detection and elimination, most of the remaining feature points were located in the static environment. We took the first frame as an initial keyframe and set its position to be the center of a local 3D coordinate system. The relationship between the 3D feature points obtained by the depth sensor, and their image points by the imaging sensor, can be expressed as follows:

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1_i \end{bmatrix} = \mathbf{K} \cdot (\mathbf{R} \cdot P_i + \mathrm{t}) = \mathbf{K} \cdot \left(\mathbf{T} \cdot \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}\right)_{(1:3)} . \tag{9}$$

Here, the $\mathbf{R} \in SO(3)$ and $\mathrm{t} \in \mathbb{R}^3$ represent the orientation and position of the camera, respectively. The matrix $\mathbf{T}$, containing camera exterior parameters $\mathbf{R}$ and t, transforms the homogeneous world coordinate system $P_i(X_i, Y_i, Z_i, 1)^{\mathrm{T}}$ to the current camera coordinate system $P'_i(X'_i, Y'_i, Z'_i)$. $\mathbf{K}$ represents the interior parameters of the camera, which was obtained from the camera calibration. The homogeneous pixel coordinate $p_i(u_i, v_i, 1)^{\mathrm{T}}$ is the corresponding re-projected point of $P_i(X_i, Y_i, Z_i, 1)^{\mathrm{T}}$. We then used the efficient perspective-n-point method (EPnP) to estimate the camera pose [45]. Due to the unknown camera pose and noise in the observation, there was a re-projection error between world map points and matched feature points, as Figure 9 shows.

We then used bundle adjustment (BA) to optimize the camera pose by minimizing the re-projection error of all matches. It can be defined as follows:

$$\{\mathbf{R}, \mathrm{t}\} = \underset{\mathbf{R},\mathrm{t}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\left\| p_i - \frac{1}{\lambda_i} \cdot K \cdot (T \cdot P_i)_{(1:3)} \right\|^2 \right). \tag{10}$$

In order to construct the unconstrained error equation, we used a six-dimensional vector $\xi \in \mathbb{R}^6$ of Lie algebra [46] to represent the camera pose $\mathbf{R}$ and t. Because the above equation is a least squares problem, we needed to calculate the derivative of the re-projection error $\mathbf{e}_i$ with respect to the optimization variables:

$$\frac{\partial \mathbf{e}_i}{\partial \delta\xi} = \frac{\partial \mathbf{e}_i}{\partial p'_i} \frac{\partial p'_i}{\partial P'_i} \frac{\partial P'_i}{\partial \delta\xi} = -\begin{bmatrix} f_x \frac{1}{Z'_i} & 0 & -f_x \frac{X'_i}{Z'^2_i} & -f_x \frac{X'_i Y'_i}{Z'^2_i} & f_x (1 + \frac{X'^2_i}{Z'^2_i}) & -f_x \frac{Y'_i}{Z'_i} \\ 0 & f_y \frac{1}{Z'_i} & -f_y \frac{Y'_i}{Z'^2_i} & -f_y (1 + \frac{Y'^2_i}{Z'^2_i}) & f_y \frac{X'_i Y'_i}{Z'^2_i} & f_y \frac{X'_i}{Z'_i} \end{bmatrix}. \tag{11}$$

Here, the f_x and f_y are the camera interior parameters. In addition to camera pose, we also needed to optimize the 3D world map point $P_i(X_i, Y_i, Z_i,)^{\mathrm{T}}$. The partial derivative of the re-projection error $\mathbf{e}_i$ with respect to $P_i(X_i, Y_i, Z_i,)^{\mathrm{T}}$ can also be obtained via the chain rule of derivative calculation:

$$\frac{\partial \mathbf{e}_i}{\partial P_i} = \frac{\partial \mathbf{e}_i}{\partial p'_i}\frac{\partial p'_i}{\partial P'_i}\frac{\partial P'_i}{\partial P_i} = -\begin{bmatrix} f_x\frac{1}{Z'_i} & 0 & -f_x\frac{X'_i}{Z'^2_i} \\ 0 & f_y\frac{1}{Z'_i} & -f_y\frac{Y'_i}{Z'^2_i} \end{bmatrix}\mathbf{R}. \tag{12}$$

Through Equation (11) and (12), we linearize the 2D pixel re-projection error $\mathbf{e}_i$ with respect to camera pose and 3D world map point. We can then use the Levenberg–Marquardt method implemented in g2o [47] to solve the least squares optimization problem described in Equation (10). Finally, a robust result of camera pose $\mathbf{R}$ and t can be obtained. In the following experimental section, we detail experiments we performed with a testing dataset to verify the effectiveness of our RGB-D SLAM system in common dynamic indoor scenes.

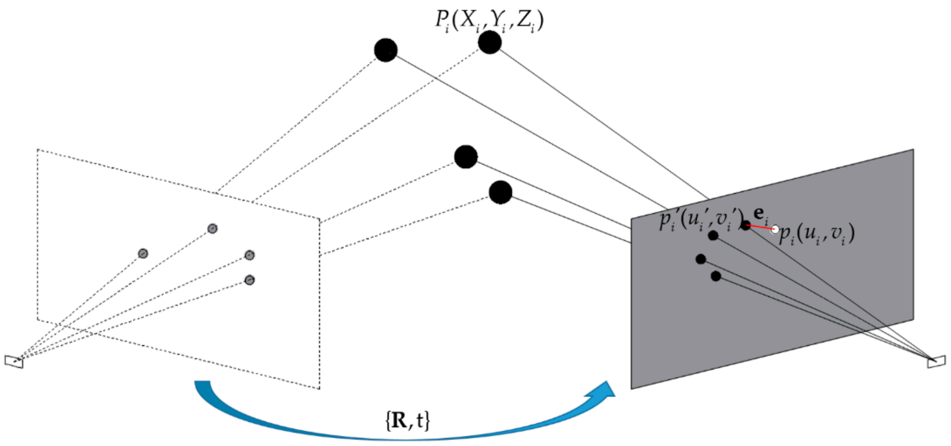


Figure 9. Illustration of re-projection error.

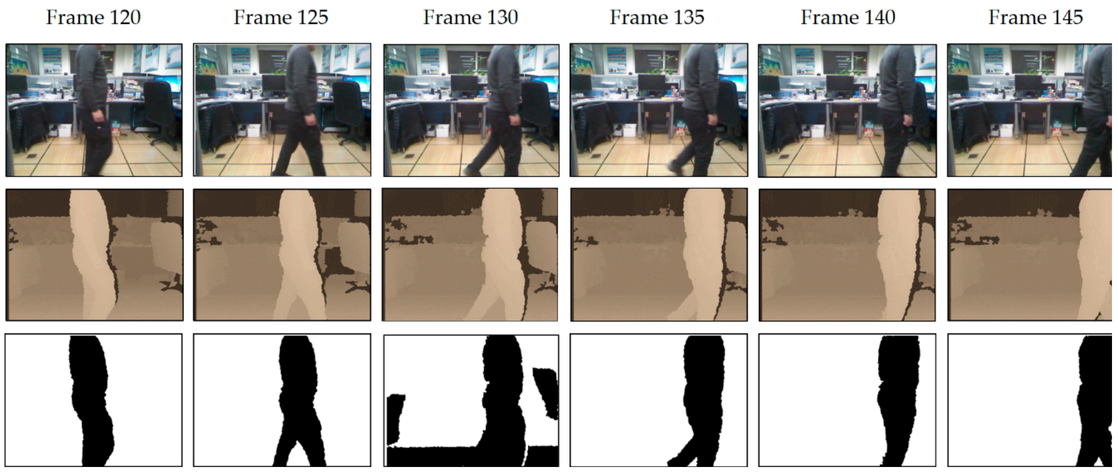
3. Experimental Results

In order to evaluate the effectiveness of our new SLAM method, we conducted a series of experiments using two datasets: sequence images captured by our RGB-D camera, and the public TUM RGB-D datasets with ground-truth. In the first experimental part, we qualitatively evaluated the proposed moving object detection method, the data filtering front-end of our improved RGB-D SLAM, with our homemade sequence images in typical dynamic indoor scenes like *corridor* and *office*. In the second part, we tested our improved RGB-D SLAM with the public TUM RGB-D datasets. We conducted experiments to analyze the performance of our improved RGB-D SLAM system with moving object detection method and the one without moving object detection method in dynamic environments. For the purpose of comparison, we also compared our SLAM system with the DVO and the BaMVO, both of which are the state-of-the-art RGB-D SLAM methods that can run robustly in a dynamic indoor environment. All the experiments were conducted on a computer with an Intel Core i7-6820HQ CPU with 2.7 GHz and 16 GB RAM. The details of the experimental results are described below.

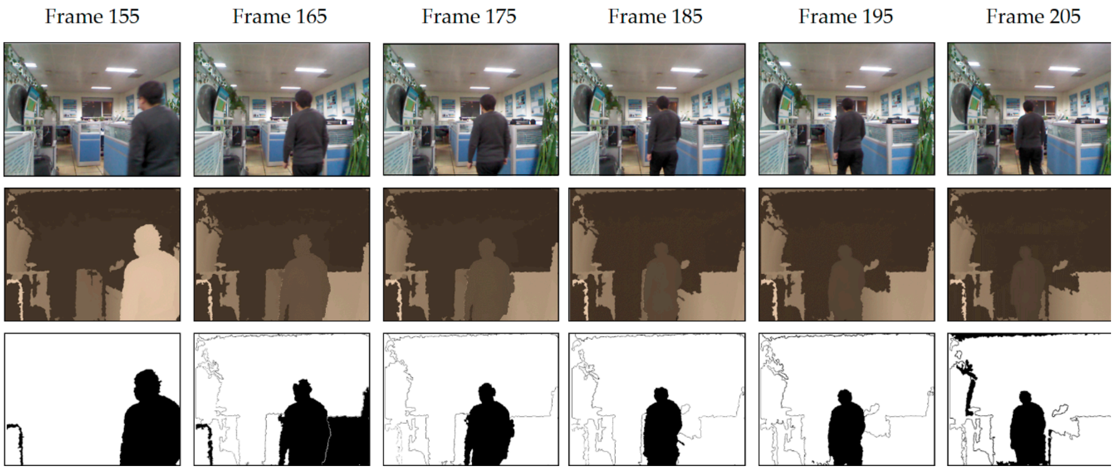
3.1. Testing with Sequence Images Captured by Our RGB-D Camera

In order to test the effectiveness of the proposed method, we recorded some RGB-D sequence images at 640 × 480 resolution with a Microsoft Kinect V1 sensor. All the data was recorded at a frame rate of 30 Hz in common scenarios, including the *office* and the *corridor*. The experimenter walked at normal speed as a moving object. The RGB-D camera was calibrated using the method described in Section 2.1. Using calibration parameters, we were able to project the depth image onto the RGB image.

The first scenario is an *office* environment, where we captured two types of sequence images. Frames selected evenly from the experimental results are shown in Figure 10. In the first sequence images of *office*, the camera was fixed and the experimenter passed through the view with a direction parallel to the camera plane. The experimenter was about 1.5 m away from the camera and walked at a normal speed. In the second sequence images of *office*, as shown in Figure 10b, the experimenter walked away from the camera with a direction perpendicular to the camera plane at a normal speed, and the interval of presented images is 10 frames. The three rows from top to bottom are RGB images, depth images, and motion detection results. The black parts in the binary images in the third row are the moving object detected by our method, and the white part is the background. As can be seen from Figure 10, our method can detect and remove the moving objects well in the frames with obvious pedestrian characteristics.



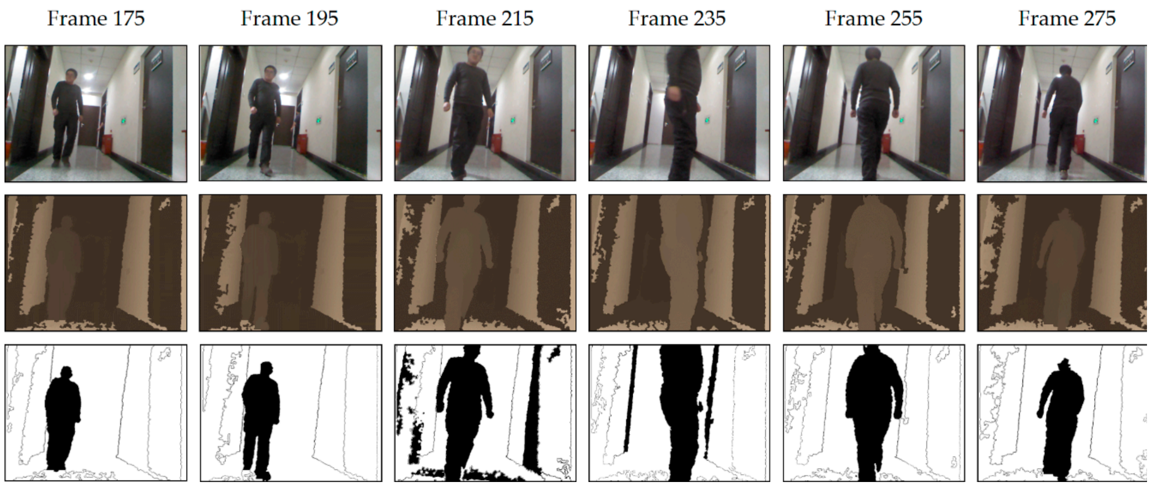
(**a**)



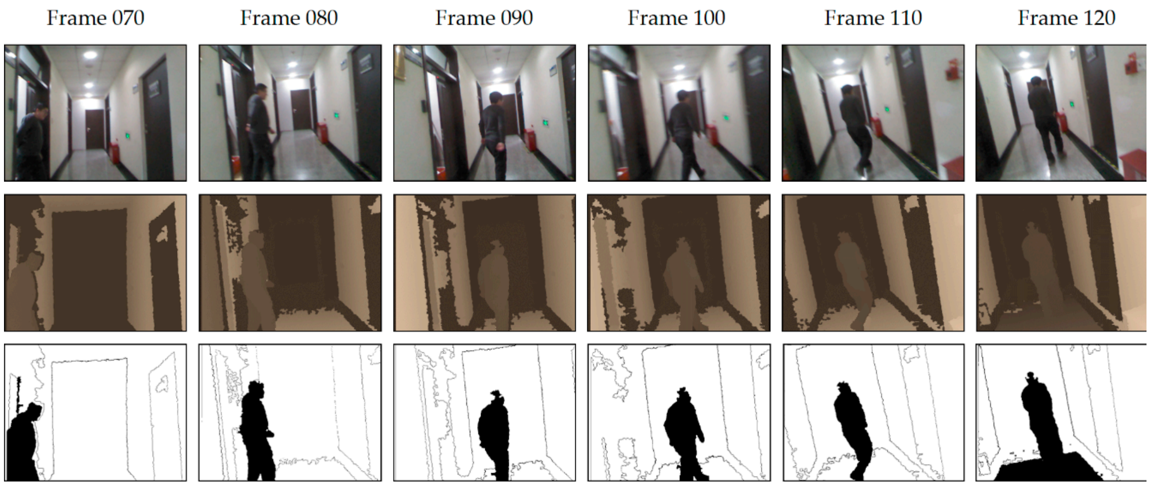
(**b**)

Figure 10. Selected experimental results of the proposed method using sequence images captured in an *office* environment. The three rows from top to bottom are RGB images, depth images, and moving object detection results. (**a**) The experimenter passed through the view with a direction parallel to the camera plane. (**b**) The experimenter walked away from the camera with a direction perpendicular to the camera plane. The moving objects detected by our method are colored in black, as shown in the third row.

The second scenario was a *corridor* environment. There are two types of sequence images in this scenario. Figure 11 shows some frames evenly selected from the experimental results. In the first sequence images, the camera was fixed. The experimenter approached the camera and turned around. As shown in Figure 11a, the spacing is 20 frames. As for the second sequence images, the camera was unfixed and swung from side to side. The experimenter just walked away from the camera with a normal speed. In Figure 11b, the spacing of the selected frames is 10 frames. According to our moving object detection method, we can get the corresponding binary images as shown in third row of each sub-figure. It is worth mentioning that some of the background parts were treated as moving objects, like Frame 215 in Figure 11a and Frame 120 in Figure 11b. Nonetheless, it can be seen from the Figure 11 that these misclassified parts are mainly white walls and floor tiles. Few feature points will be extracted in these areas, so, even if they are removed, this situation will have little influence on the subsequent camera pose estimation. Overall, our method can detect moving objects well in most frames. After the moving objects are detected, the feature points belonging to the moving objects are eliminated in the subsequent process.



(**a**)



(**b**)

Figure 11. Some experimental results of the proposed method using sequence images captured in a *corridor* environment. (**a**) RGB-D camera was fixed and the experimenter approached the camera and then turned away from it; (**b**) RGB-D camera swung from side to side and the experimenter just walked away from the camera plane. The moving objects detected by our method are colored in black, as shown in the third row of each sub-figure.

3.2. *Evaluation Using TUM RGB-D SLAM Datasets*

In this part, the TUM RGB-D SLAM datasets were used to evaluate the proposed RGB-D SLAM method. The RGB and depth images were recorded at frame rate of 30 Hz and a 640×480 resolution. Ground-truth trajectories obtained from a high-accuracy motion-capture system are provided in the

TUM datasets. Evaluation tools are available for trajectory accuracy estimation with the ground-truth data. The TUM datasets have a Dynamic Objects category containing three typical dynamic indoor scenes: *desk*, *sitting*, and *walking*. In the *desk* sequence, a person sits at a desk in an office. In the *sitting* sequences, two persons sit at a desk with a little gesture. These two scenarios can be considered lowly dynamic environments. In the *walking* sequences, two persons walk through an office scene. *Walking* sequences can be used to evaluate the robustness of the proposed method in highly dynamic scenes with quickly moving objects. As is described on the website of the TUM datasets, the RGB-D camera has four different states of motion in the *sitting* and *walking* sequences, which are *static*, *xyz*, *halfsphere*, and *rpy*. The camera is fixed on a position when it is *static* state, and moves along three directions at the *xyz* state. For the *halfsphere* state, the camera moves on a 1 m diameter hemisphere. As for the *rpy* state, the camera rotates along the principal axes (row-pitch-yaw) at the same position.

Firstly, we conducted a quantitative evaluation of our new SLAM method on dynamic TUM datasets. The ATE, which represents the difference between the estimated trajectory and the ground-truth, was used as the evaluation metric. In particular, an easy-to-use open-source package evo was employed for the evaluation (github.com/MichaelGrupp/evo). Figure 12 shows the accuracy on two highly dynamic datasets. The blue lines represent the ATE of the RGB-D SLAM without moving object detection, and the lines in orange represent the ATE of our new SLAM method with moving object detection. As we can see in blue lines of Figure 12a,b, when there are pedestrians in the FOV of the camera, the ATE values increase dramatically. However, the ATE values of orange lines at those same moments were greatly reduced, because the influence of pedestrians was eliminated with our new RGB-D SLAM.

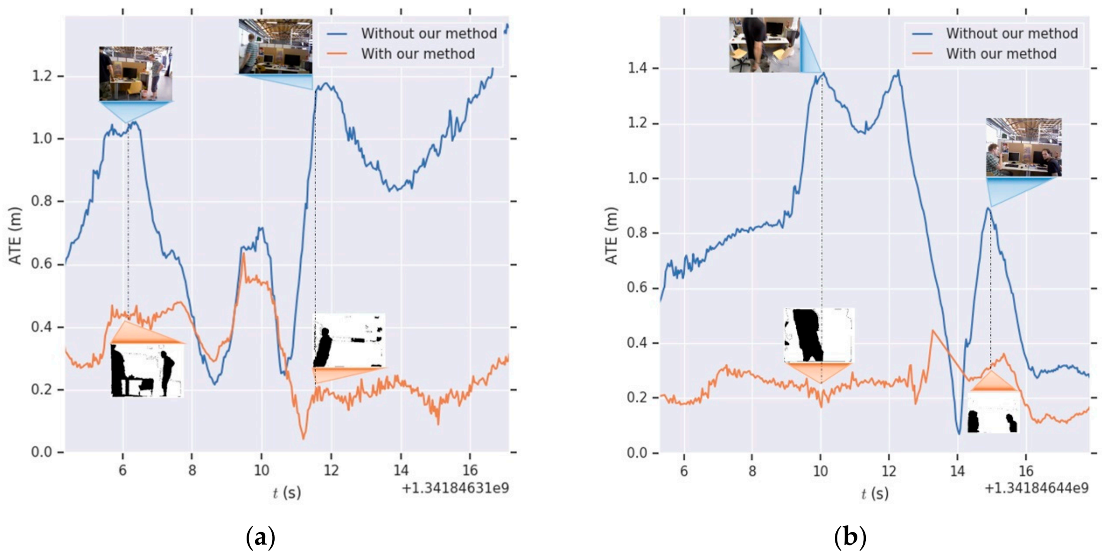


Figure 12. Accuracy of the experiment without moving object detection (blue) and the experiment with our new method (orange) on two highly dynamic datasets. The horizontal axis represents time in seconds, and the longitudinal axis represents the absolute trajectory error (ATE) in meters. (**a**) 1341846315–1341846327 s part of the *fr3/walking_xyz* dataset; (**b**) 1341846446–1341846457 s part of the *fr3/walking_halfsphere* dataset. The RGB images of blue lines indicate the presence of pedestrians in the FOV of the camera. The black areas in the binary images of orange lines indicate that our improved RGB-D SLAM excluded the influence of pedestrians at these moments when ATE values increased dramatically.

Table 3 shows the ATE on the *sitting* and *walking* dynamic datasets of RGB-D SLAM with and without the proposed moving object detection. From the root mean square error (RMSE) column, we can see that the proposed RGB-D SLAM system with moving object detection method had a lower RMSE than the one without moving object detection in these dynamic environments. In the first two *sitting* datasets, the new RGB-D SLAM with moving object detection improved the accuracy slightly. This is because the *sitting* datasets are lowly dynamic scenes where pedestrians are sitting on chairs, so the pedestrians have little impact on the camera's ego-motion estimation. The next four *walking* datasets represent highly dynamic scenes with quick moving objects. By using our moving object detection method, the proposed RGB-D SLAM greatly improved the accuracy in these datasets. The results clearly demonstrate the effectiveness and robustness of our RGB-SLAM in both lowly and highly dynamic indoor scenes.

Table 3. ATE on the *sitting* and *walking* dynamic datasets of the proposed RGB-D SLAM with and without the moving object detection. The numbers in bold indicate that these terms are better than those of another method.

Dynamic TUM RGB-D Datasets	**Without Moving Object Detection ATE (m)**				**With Moving Object Detection ATE (m)**			
	Max.	**Avg.**	**Min.**	**RMSE**	**Max.**	**Avg.**	**Min.**	**RMSE**
fr3/sitting_static	0.0381	0.0078	0.0007	0.0087	**0.0255**	**0.0059**	**0.0002**	**0.0066**
fr3/sitting_halfsphere	0.0971	0.0194	0.0018	0.0230	**0.0919**	**0.0152**	**0.0017**	**0.0196**
fr3/walking_static	0.6391	0.3513	0.0587	0.3843	**0.6161**	**0.2691**	**0.0231**	**0.3080**
fr3/walking_xyz	1.3661	0.6309	0.0612	0.7328	**0.8196**	**0.2768**	**0.0430**	**0.3047**
fr3/walking_halfsphere	1.5683	0.5831	**0.0690**	0.6891	**0.6246**	**0.2972**	0.1088	**0.3116**
fr3/walking_rpy	1.1136	0.4975	0.1118	0.5523	**0.9351**	**0.4637**	**0.0708**	**0.4983**

[1] The unit of each term is the meter. RMSE = root mean square error.

To demonstrate the results intuitively, several accuracy heat maps of the proposed RGB-D SLAM with and without our moving object detection method are shown in Figure 13. The gray dotted line represents the reference ground-truth. The color solid lines represent the estimated trajectories. The color bar represents the size of the ATE. Each row shows the results of the two methods with the same dataset, and the two color bars of each row have the same maximum error and minimum error. Comparing the three pairs of trajectories, we can observe that the proposed RGB-D SLAM with moving object detection method showed better accuracy in both lowly and highly dynamic indoor scenes.

We then compared our new method with the state-of-the-art RGB-D SLAM methods DVO and BaMVO for comparison. The DVO method proposed by Kerl et al. [37] uses a robust error function that reduces the influence of large residuals caused by moving objects. The BaMVO proposed by Kim et al. [39] detects moving objects through accumulation of the warped difference images between the current and remapped depth images over a period of time. These two RGB-D SLAM methods can run robustly in a dynamic indoor environment. For a fair comparison of the three methods, we closed the loop closure detection of our RGB-D SLAM in the experiment and used the RMSE results of these two reference methods in their papers. We adopted the relative pose error (RPE) as another evaluation metric, which described the error with a distance of one second in the estimated trajectory file. Meanwhile, we calculated both the translation and the rotation part of RPE as an evaluation of six DoFs.

Table 4 shows the RPE results of DVO, BaMVO, and our RGB-D SLAM. We calculated the RMSE of three methods in nine scenes, including lowly and highly dynamic indoor environments. As we can see in Table 4, our RGB-D SLAM system performed better in most scenes of TUM dynamic datasets in terms of the translation RMSE and rotation RMSE. In the lowly dynamic environment, compared with the other two methods, our system had a lower RPE in the translation and rotation parts. As for the highly dynamic environment, our system improved the precision greatly when compared with DVO, and had better accuracy in most scenes when compared with BaMVO. This shows that the proposed RGB-D SLAM system with a moving object detection method can yield smaller errors than the reference methods, and performs better in common dynamic environments.

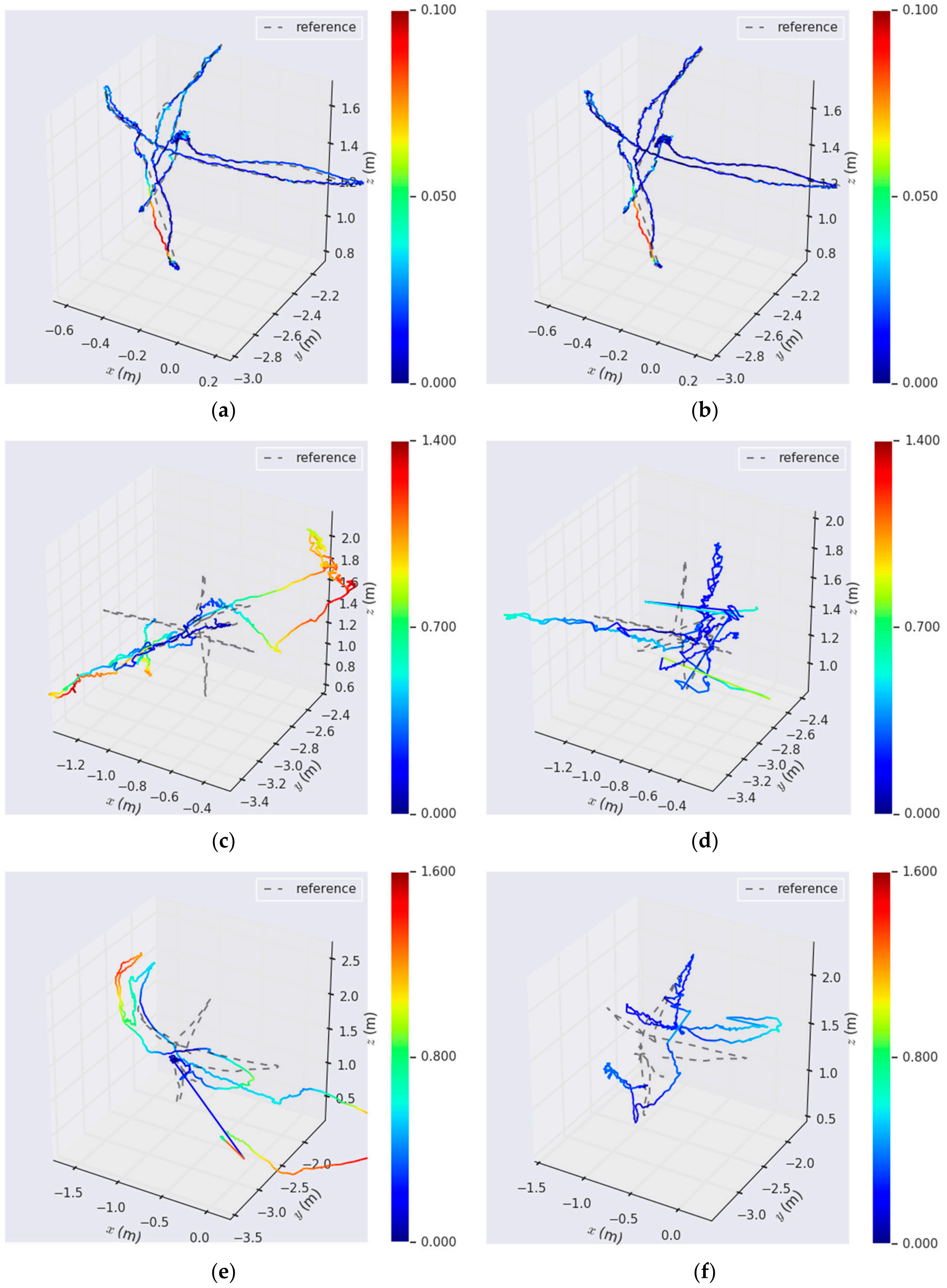


Figure 13. Several accuracy heat maps of the proposed RGB-D SLAM with and without our moving object detection method. The three accuracy heat maps of the left column were estimated without using the moving object detection method on the (**a**) *fr3/sitting_halfsphere*, (**c**) *fr3/walking_xyz*, and (**e**) *fr3_walking_halfsphere* sequences. The three accuracy heat maps of the right column were estimated using the method on the (**b**) *fr3/sitting_halfsphere*, (**d**) *fr3/walking_xyz*, and (**f**) *fr3_walking_halfsphere* sequences. The two color bars of each row have the same maximum error and minimum error. The redder the color is, the larger the ATE is.

Table 4. Translation RMSE and rotation RMSE of relative pose error (RPE) for TUM RGB-D datasets in the dynamic objects category. The numbers in bold indicate that these terms are better than those of another method.

TUM RGB-D Datasets Category: Dynamic Objects		**Translation RMSE (m/s)**			**Rotation RMSE (°/s)**		
		DVO	**BaMVO**	**Ours**	**DVO**	**BaMVO**	**Ours**
Low Dynamic Environment	fr2/desk_with_person	0.0354	0.0352	**0.0069**	1.5368	1.2159	**0.4380**
	fr3/sitting_static	0.0157	0.0248	**0.0077**	0.6084	0.6977	**0.2595**
	fr3/sitting_xyz	0.0453	0.0482	**0.0117**	1.4980	1.3885	**0.4997**
	fr3/sitting_halfsphere	0.1005	0.0589	**0.0245**	4.6490	2.8804	**0.5643**
	fr3/sitting_rpy	0.1735	0.1872	**0.0234**	6.0164	5.9834	**0.7838**
High Dynamic Environment	fr3/walking_static	0.3818	**0.1339**	0.1881	6.3502	**2.0833**	3.2101
	fr3/walking_xyz	0.4360	0.2326	**0.2158**	7.6669	4.3911	**3.6476**
	fr3/walking_halfsphere	0.2628	**0.1738**	0.1908	5.2179	4.2863	**3.3321**
	fr3/walking_rpy	0.4038	0.3584	**0.3270**	7.0662	6.3398	**6.3215**

[1] Relative pose error for pose pairs with a distance of 1 second.

4. Conclusions

In this paper, we present a new RGB-D SLAM method with moving object detection for dynamic indoor scenes. The proposed RGB-D SLAM has two main innovations. First, we developed a new method to detect and eliminate moving objects based on mathematical models and geometric constraints. The proposed moving object detection method can be incorporated into the SLAM system as a data filtering process. Second, we clustered the filled depth images and used them to segment moving objects. Therefore, our improved RGB-D SLAM can detect moving objects and alleviate the impact of them better than those methods for which the moving object detection is based on subtraction between consecutive images.

In the experimental section, the proposed RGB-D SLAM was qualitatively evaluated with our homemade sequence images in some typical dynamic indoor scenes like *corridor* and *office*. We then conducted experiments to demonstrate that the proposed RGB-D SLAM can greatly improve the accuracy in dynamic indoor scenes by using our moving object detection method. We also presented the quantitative evaluation results of our improved RGB-D SLAM by comparing with DVO and BaMVO, which both are the state-of-the-art RGB-D SLAM algorithms in dynamic environments by incorporating data processing methods or moving object detection into the SLAM process. These quantitative evaluations were carried out using the TUM RGB-D datasets. All the experimental results show that the proposed RGB-D SLAM was able to run effectively and stably in common dynamic indoor scenes. The attained accuracy of pose estimation was generally higher than that of reference methods.

In the future, we plan to continue improving our RGB-D SLAM method by introducing other clustering methods, such as the Gaussian mixture model. In addition, we intend to combine our moving object detection model with a lightweight deep learning method, so as to achieve robust results of moving object detection in challenging dynamic environments.

Author Contributions: R.W., K.D. and W.W. conceived the idea and designed the methods and experiments; R.W. and Y.W. performed the experiments; K.D. and W.W. analyzed the data; R.W., K.D. and W.W. wrote the paper.

Funding: This research was funded by National Key Research and Development Program of China (No. 2016YFB0502102 and No. 2018YFB1305004).

Acknowledgments: We thank Computer Vision Group from Technical University of Munich for making the RGB-D SLAM dataset publically available.

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

Abbreviations

The following abbreviations are used in this manuscript:

SLAM	Simultaneous localization and mapping
LiDAR	Light detection and ranging
GPS	Global positioning system
IMUs	Inertial measurement units
3D	Three-dimension
SURF	Speeded-Up Robust Features
ORB	Oriented FAST and rotated BRIEF
ORB-SLAM	ORB feature based SLAM
DoFs	Degree of freedom
DVO	Dense visual odometry
RFS	Random finite set
FOV	Field of view
BaMVO	Background model-based dense-visual-odometry
IAICP	Intensity assisted iterative closest point
FPS	Frames per second
EPnP	Efficient perspective-n-point method
BA	Bundle adjustment
ATE	Absolute trajectory error
RMSE	Root mean square error
RPE	Relative pose error

References

1. Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: Part I. *IEEE Robot. Autom. Mag.* **2006**, *13*, 99–110. [CrossRef]
2. Bailey, T.; Durrant-Whyte, H. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robot. Autom. Mag.* **2006**, *13*, 108–117. [CrossRef]
3. Dissanayake, M.W.M.G.; Newman, P.; Clark, S.; Durrant-Whyte, H.F.; Csorba, M.A. Solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. Robot. Autom.* **2001**, *17*, 229–241. [CrossRef]
4. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278. [CrossRef]
5. Fuentes-Pacheco, J.; Ruiz-Ascencio, J.; Rendón-Mancha, J.M. Visual simultaneous localization and mapping: A survey. *Artif. Intell. Rev.* **2015**, *43*, 55–81. [CrossRef]
6. Ido, J.; Shimizu, Y.; Matsumoto, Y.; Ogasawara, T. Indoor Navigation for a Humanoid Robot Using a View Sequence. *Int. J. Robot. Res.* **2009**, *28*, 315–325. [CrossRef]
7. Celik, K.; Chung, S.J.; Clausman, M.; Somani, A.K. Monocular vision SLAM for indoor aerial vehicles. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (ICRA), St. Louis, MO, USA, 11–15 October 2009; pp. 1566–1573.
8. Davison, A.J.; Reid, I.D.; Molton, N.D.; Stasse, O. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1052–1067. [CrossRef] [PubMed]
9. Lemaire, T.; Lacroix, S. Monocular-vision based SLAM using Line Segments. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Roma, Italy, 10–14 April 2007; pp. 2791–2796.
10. Celik, K.; Chung, S.J.; Somani, A. Mono-vision corner SLAM for indoor navigation. In Proceedings of the 2008 IEEE International Conference on Electro/information Technology, Winsor, ON, Canada, 7–9 June 2008; pp. 343–348.
11. Wu, K.; Di, K.; Sun, X.; Wan, W.; Liu, Z. Enhanced monocular visual odometry integrated with laser distance meter for astronaut navigation. *Sensors* **2014**, *14*, 4981–5003. [CrossRef]

12. Jiang, Y.; Chen, H.; Xiong, G.; Scaramuzza, D. ICP Stereo Visual Odometry for Wheeled Vehicles based on a 1DOF Motion Prior. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 585–592.
13. Gomez-Ojeda, R.; Gonzalez-Jimenez, J. Robust stereo visual odometry through a probabilistic combination of points and line segments. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 2521–2526.
14. Gomez-Ojeda, R.; Moreno, F.A.; Scaramuzza, D.; Gonzalez-Jimenez, J. PL-SLAM: A Stereo SLAM System through the Combination of Points and Line Segments. *arXiv* **2017**, arXiv:1705.09479. [CrossRef]
15. Kaess, M.; Dellaert, F. Probabilistic structure matching for visual SLAM with a multi-camera rig. *Comput. Vis. Image Underst.* **2010**, *114*, 286–296. [CrossRef]
16. Moratuwage, D.; Wang, D.; Rao, A.; Senarathne, N. RFS Collaborative Multivehicle SLAM: SLAM in Dynamic High-Clutter Environments. *IEEE Robot. Autom. Mag.* **2014**, *21*, 53–59. [CrossRef]
17. He, Y.; Zhao, J.; Guo, Y.; He, W.; Yuan, K. PL-VIO: Tightly-Coupled Monocular Visual-Inertial Odometry Using Point and Line Features. *Sensors* **2018**, *18*, 1159. [CrossRef] [PubMed]
18. Bay, H.; Ess, A.; Tuytelaars, T.; Gool, L.V. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.* **2008**, *110*, 346–359. [CrossRef]
19. Rublee, E.; Rabaud, V.; Konolige, K. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 2564–2571.
20. Hu, G.; Huang, S.; Zhao, L.; Alempijevic, A.; Dissanayake, G. A robust RGB-D SLAM algorithm. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 7–12 October 2012; pp. 1714–1719.
21. Ji, Y.; Yamashita, A.; Asama, H. RGB-D SLAM using vanishing point and door plate information in corridor environment. *Intell. Serv. Robot.* **2015**, *8*, 105–114. [CrossRef]
22. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
23. Kerl, C.; Sturm, J.; Cremers, D. Dense visual SLAM for RGB-D cameras. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 2100–2106.
24. Wang, C.C.; Thorpe, C.; Thrun, S.; Hebert, M.; Durrant-Whyte, H. Simultaneous Localization, Mapping and Moving Object Tracking. *Inter. J. Robot. Res.* **2007**, *26*, 889–916. [CrossRef]
25. Alcantarilla, P.; Yebes, J.; Almazn, j.; Bergasa, L. On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA), Saint Paul, MN, USA, 14–18 May 2012; pp. 1290–1297.
26. Wang, Y.; Huang, S. Towards dense moving object segmentation based robust dense RGB-D SLAM in dynamic scenarios. In Proceedings of the 2014 13th International Conference on Control Automation Robotics& Vision (ICARCV), Singapore, 10–12 December 2014; pp. 1841–1846.
27. Bakkay, M.C.; Arafa, M.; Zagrouba, E. Dense 3D SLAM in dynamic scenes using Kinect. In Proceedings of the 7th Iberian Conference on Pattern Recognition and Image Analysis, Santiago de Compostela, Spain, 17–19 June 2015; pp. 121–129.
28. Sun, Y.; Liu, M.; Meng, Q.H. Improving RGB-D SLAM in dynamic environments: A motion removal approach. *Robot. Autom. Syst.* **2017**, *89*, 110–122. [CrossRef]
29. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Algarve, Portugal, 7–12 October 2012; pp. 573–580.
30. Yang, S.; Wang, J.; Wang, G.; Hu, X.; Zhou, M.; Liao, Q. Robust RGB-D SLAM in dynamic environment using faster R-CNN. In Proceedings of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 13–16 December 2017; pp. 5702–5708.
31. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection withregion proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1142. [CrossRef]

32. Zhong, F.; Wang, S.; Zhang, Z.; Zhou, C.; Wang, Y. Detect-SLAM: Making Object Detection and SLAM Mutually Beneficial. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision, Lake Tahoe, NV, USA, 12–15 March 2018; pp. 1001–1010.
33. Yang, D.; Bi, S.; Wang, W.; Yuan, C.; Wang, W.; Qi, X.; Cai, Y. DRE-SLAM: Dynamic RGB-D Encoder SLAM for a Differential-Drive Robot. *Remote Sens.* **2019**, *11*, 380. [CrossRef]
34. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
35. Moratuwage, D.; Vo, B.-N.; Wang, D. Collaborative multi-vehicle SLAM with moving object tracking. In Proceedings of the 2013 IEEE Int. Conf. Robotics &Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 5702–5708.
36. Zou, D.; Tan, P. CoSLAM: collaborative visual SLAM in dynamic environments. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 354–366. [CrossRef]
37. Kerl, C.; Sturm, J.; Cremers, D. Robust odometry estimation for RGB-D cameras. In Proceedings of the 2013 IEEE Int. Conf. Robotics &Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 3748–3754.
38. Lee, D.; Myung, H. Solution to the SLAM Problem in Low Dynamic Environments Using a Pose Graph and an RGB-D Sensor. *Sensors* **2014**, *14*, 12467–12496. [CrossRef]
39. Kim, D.H.; Kim, J.H. Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment. *IEEE Trans. Robot.* **2017**, *32*, 1565–1573. [CrossRef]
40. Li, S.; Lee, D. RGB-D SLAM in Dynamic Environments using Static Point Weighting. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2263–2270. [CrossRef]
41. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
42. Soille, P. Morphological Image Analysis: Principles and Applications. *Sens. Rev.* **1999**, *28*, 800–801.
43. Richard, H.; Andrew, Z. *Multiple View Geometry in Computer Vision*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2003; pp. 241–253.
44. Massart, D.L.; Kaufman, L. Least median of squares: a robust method for outlier and model error detection in regression and calibration. *Analytica Chimica Acta* **1986**, *187*, 171–179. [CrossRef]
45. Lepetit, V.; Moreno-Noguer, F.; Fua, P. EPnP: An accurate O(n) solution to the PnP problem. *Inter. J. Comput. Vis.* **2009**, *81*, 155–166. [CrossRef]
46. Hall, B.C. *Lie Groups, Lie Algebras, and Representations*, 2nd ed.; Springer: New York, NY, USA, 2015; pp. 49–75.
47. Kümmerle, R.; Grisetti, G.; Strasdat, H.; Konolige, K.; Burgard, W. G2o: A general framework for graph optimization. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 3607–3613.