Article

PLD-SLAM: A New RGB-D SLAM Method with Point and Line Features for Indoor Dynamic Scene

Chenyang Zhang ^{1,*}, Teng Huang ¹, Rongchun Zhang ² and Xuefeng Yi ¹

¹ School of Earth Sciences and Engineering, Hohai University, Nanjing 211100, China; 20191003@hhu.edu.cn (T.H.); 180209020007@hhu.edu.cn (X.Y.)

² School of Geographic and Biologic Information, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; rongchunzhang@njupt.edu.cn

* Correspondence: zcynj@hhu.edu.cn

Abstract: RGB-D SLAM (Simultaneous Localization and Mapping) generally performs smoothly in a static environment. However, in dynamic scenes, dynamic features often cause wrong data associations, which degrade accuracy and robustness. To address this problem, in this paper, a new RGB-D dynamic SLAM method, PLD-SLAM, which is based on point and line features for dynamic scenes, is proposed. First, to avoid under-over segmentation caused by deep learning, PLD-SLAM combines deep learning for semantic information segmentation with the K-Means clustering algorithm considering depth information to detect the underlying dynamic features. Next, two consistency check strategies are utilized to check and filter out the dynamic features more reasonably. Then, to obtain a better practical performance, point and line features are utilized to calculate camera pose in the dynamic SLAM, which is also different from most published dynamic SLAM algorithms based merely on point features. The optimization model with point and line features is constructed and utilized to calculate the camera pose with higher accuracy. Third, enough experiments on the public TUM RGB-D dataset and the real-world scenes are conducted to verify the location accuracy and performance of PLD-SLAM. We compare our experimental results with several state-of-the-art dynamic SLAM methods in terms of average localization errors and the visual difference between the estimation trajectories and the ground-truth trajectories. Through the comprehensive comparisons with these dynamic SLAM schemes, it can be fully demonstrated that PLD-SLAM can achieve comparable or better performances in dynamic scenes. Moreover, the feasibility of camera pose estimation based on both point features and line features has been proven by the corresponding experiments from a comparison with our proposed PLD-SLAM only based on point features.

Keywords: RGB-D; Simultaneous Localization and Mapping; dynamic scenes; deep learning; target detection; K-means

Citation: Zhang, C.; Huang, T.; Zhang, R.; Yi, X. PLD-SLAM: A New RGB-D SLAM Method with Point and Line Features for Indoor Dynamic Scene. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 163. <https://doi.org/10.3390/ijgi10030163>

Academic Editors: Yasir Latif, Pulak Purkait and Wolfgang Kainz

Received: 6 December 2020

Accepted: 12 March 2021

Published: 13 March 2021

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

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

1. Introduction

Simultaneous Localization and Mapping (SLAM) [1] was first proposed by researchers in 1986 and has been used to estimate the pose of a mobile robot and construct an incremental map of unknown surroundings simultaneously [2,3]. SLAM is the key technology for robots navigating in unknown environments and has become a hotspot of the robotics research community [4]. Currently, various sensors (e.g., radar, ultrasonic, laser) have been applied in SLAM to help a robot perceive information in an unknown scene [5]. In comparison to the ranging type of sensors, a camera with advantages of smaller size and low power consumption can provide abundant texture information for robots in unknown environments, especially in GNSS-denied (Global Navigation Satellite System) environments (e.g., lunar, Martian surface, and underground environments). A SLAM system based on the data input of a vision camera is usually called visual SLAM. Coinciding with the development of computer technology, visual SLAM has attracted extensive attentions and has been applied successfully to planetary exploration missions such as the NASA's

Mars Exploration Rover 2003 (MER) mission [6], and China's Chang'E-3 mission [7] and Chang'E-4 mission [8].

Over the past few decades, some well-known and robust SLAM schemes have been successively released and have achieved better performances and comparable location accuracies. In general, these visual SLAM algorithms are roughly divided into two categories: Feature-based SLAM (e.g., PTAM [9], ORB-SLAM2 [10]) and direct SLAM (e.g., LSD-SLAM [11], DVO [12]) methods. However, most of these visual SLAM methods are implemented under the basic premise of static environmental conditions. In other words, a static environmental condition means that the geometric distribution of scenes is static and invariant in the world coordinate system. At present, these classical feature-based SLAM systems [9,10] and direct SLAM system [11,12] are incapable of distinguishing whether the scene features are static or dynamic. If some feature matches have direct data associated with the motion features, the wrong data association will inevitably occur and lead to the degradation of pose estimation accuracy or even failure. Meanwhile, most robust visual SLAM systems are incapable of recognizing and rejecting the motion features, and they are greatly restricted in most applications of dynamic scenes. Accordingly, the relevant research on visual SLAM for dynamic scenes have attracted extensive attention in recent years.

In the past few years, visual SLAM methods for dynamic scenes have been proposed to improve the robustness and performance of visual SLAM. For instance, some SLAM methods combined with deep learning for dynamic scenes were used to improve the performances of SLAM [13]. Models included in these methods always bring about over-segmentation of dynamic features and losses of static features. Moreover, fewer point features will reduce the robustness of the camera pose estimation. To calculate the camera pose with higher precision in dynamic scenes, line features can also be utilized, except for point features. Compared to a monocular camera or stereo camera, an RGB-D camera can provide color and depth images simultaneously, and it has been applied widely in the visual SLAM community. Therefore, in this paper, a new RGB-D SLAM scheme based on point and line features for dynamic scenes called PLD-SLAM is proposed. First, as the RGB and depth images can be obtained by an RGB-D camera simultaneously, we mainly combine the RGB and depth information to detect these dynamic points and line features. The MobileNet model [13] is initially used to segment the semantic information. Next, we regarded the semantic segmentation region as underlying dynamic regions and check whether these extraction points and lines are in the regions. Then, by means of the depth image, the K-Means clustering algorithm is employed to filter out as many potentially dynamic point and line features as possible. The line features that have been considered as potentially dynamic features are regarded as the dynamic line features. The remaining line features are saved and treated as static features. For detecting and seeking real dynamic point features, another two strategies are utilized to identify the real dynamic point features among these potentially dynamic point features. Finally, combining the depth image with point and line features, an optimization model based on points and lines is used to calculate the camera pose. The main contributions of our work are summarized as follows:

- We propose a new RGB-D SLAM method for the dynamic scenes called PLD-SLAM that combines point features with line features. PLD-SLAM applies the MobileNet model and K-Means clustering of depth information to obtain the dynamic feature points and lines. Then, PLD-SLAM applies two strategies to identify dynamic feature points and remains abundant of static features as much as possible at the same time.
- We combine this with depth information and adopt an optimization model based on points and lines to achieve a higher-accuracy camera pose and robustness performance for dynamic scenes.
- We evaluate our PLD-SLAM on the public TUM RGB-D datasets [14] that contain typical dynamic scene sequences and real scenes. Compared to the state-of-the-art visual SLAM (e.g., ORB-SLAM2 [10]) and some classical dynamic SLAM systems, PLD-

SLAM outperforms the performance of most dynamic SLAM systems and achieves comparable accuracy.

The rest of this paper is organized as follows. Section 2 presents a review of related work. Section 3 describes the details of PLD-SLAM. The experimental results of the public TUM datasets and real scene and the discussion are presented in Section 4. Finally, conclusions and future work are presented in Section 5.

2. Related Works

The basic idea of dynamic SLAM is that the dynamic features are detected and considered as outliers [15]. The dynamic features will be eliminated and not be utilized to calculate the camera pose. In addition to these dynamic SLAM schemes based on the conventional outlier rejection algorithms of the math model, these methods of integrating deep learning with visual SLAM have also been extensively studied and applied for dynamic SLAM in past few years. The related research works of dynamic SLAM based on the traditional math model are introduced in Section 2.1, and these works of dynamic SLAM methods combined with deep learning are presented in Section 2.2.

2.1. Dynamic SLAM Based on Conventional Models

The first visual SLAM algorithm for dynamic scenes was proposed in 2007, which mainly included the detection and elimination of dynamic objects [15]. Since then, the general idea of a SLAM algorithm for dynamic scenes was formed where dynamic objects need to be detected and eliminated in the image sequence. Moratuwage [16] applied a stochastic finite element model to represent the visual features and measurement results in the angle field of the sensor and used the Bayesian regression model to denote the static and dynamic map points. Zou [17] proposed a Co-SLAM system for the dynamic scene with multiple cameras and applied a 2D re-projection error of point features to distinguish whether the point features are static or dynamic. Lee [18] used the pose graph to represent the robot pose and eliminated the scale dynamic feature correlation according to the error threshold of the node in the pose graph. In general, as the dynamic scenes are relatively complex, the above-mentioned SLAM methods cannot detect and check these dynamic and static features accurately. In consideration of the assumption that most visual features in the scene are static and a few features are dynamic, Wang [19] integrated the assumption into dynamic and static feature segmentation and raised a dense RGB-D SLAM algorithm. Similarly, Bakkay [20] assumed that adjacent frames moved at an almost constant speed and the velocities of dynamic features would be higher than those of static features. Therefore, for the moving object in the image sequence, dynamic features can be detected and removed according to this criterion. However, because of the complexity of motion objects in dynamic scenes and the effect of camera motion, the method referred to in [20] is not suitable for most motion objects. Sun [21] put forward a motion removal method that combined the Maximum Posterior Estimation to identify and eliminate dynamic features. He integrated the proposed method into the dense visual odometer DVO [12] and tested it on the RGB-D datasets. As an RGB-D camera can provide the depth image, Kim [22] proposed to reduce the influence of moving objects by estimating the background model from the depth scene and calculating the camera pose of adjacent frames. Wang [23] proposed a new RGB-D SLAM by adopting the fundamental matrix constraint and depth clustering algorithm to get rid of these wrong associations with these motion features. Generally, these SLAM systems can eliminate these point features with evident motion well, while a few features with slight motion are not detected and are rejected effectively normally.

In addition to the outlier rejection algorithms established on the math model directly, the optical flow field with the ability to recognize dynamic features is also utilized to check and eliminate the motion features. Some dynamic SLAM methods based on the optical flow have been proposed. Shimamura [24] used the optical flow field and Gaussian Mixture model to identify and reject the dynamic features. Cheng [25] combined the optical

flow field and fundamental matrix to eliminate dynamic features. As the visual camera is also in motion, the dynamic SLAM combined with the optical flow field usually cannot detect dynamic features effectively in some low-dynamic scenes. To detect and eliminate dynamic features effectively, a dynamic SLAM based on the dense optical flow has also been proposed. Alcantarilla [26] used the dense scene flow to check moving objects within a range of 5 m, but the method has a range limit for detecting dynamic objects, and it is easy to misjudge static point features as dynamic point features in low-texture scenes. In the actual dynamic scene, due to the motion effect of the visual camera, the optical flow method-based vision SLAM often has poor effects and is easily affected by illumination and other factors. In addition, the additional sensors are used to help detect and eliminate dynamic point features in dynamic scenes. Liu et al. [27] used different view fields of sensors to detect and reject outlier points in dynamic scenes. The inertial measurement unit (IMU) was also applied to dynamic SLAM for outlier point elimination in the dynamic scene. Kim [28] utilized an RGB-D camera and IMU to achieve SLAM in dynamic scenes, where IMU measurement information was served as the prior information of pose estimation to eliminate the wrong visual association in the dynamic scene.

2.2. Dynamic SLAM Based on Deep Learning

Recently, as the deep learning network can segment the semantic information well, some dynamic SLAM systems integrated with deep learning networks were proposed to improve the performance of SLAM systems. Bescos [29] was the first to propose Dyna-SLAM by combining the Mask R-CNN [30] and multiple-view geometry to detect dynamic objects, and the source code was released. As deep learning could segment semantic information effectively, most dynamic SLAM research based on deep learning has attracted extensive attention. Yang [31] combined this with the convolutional neural network of Fast-RCNN [32] to propose a dynamic RGB-D SLAM. He used the convolutional neural network to detect the semantic information and presented it by a rectangular box, and then considered the point features located in the rectangular box as outliers and rejected them. Likewise, Wang [33] proposed to use the deep learning model named YOLO (You Only Look Once) to identify moving objects in the image sequences. Due to the lack of identification and judgment of dynamic features, the proposed method referred to in [31,33] removes this possible dynamic semantic information. Yu et al. [34] used the SegNet [35] to accomplish a pixel-wise semantic segmentation of the image and combined it with a motion consistency checking strategy to filter out dynamic portions of the scene. Yu [34] could also produce a dense semantic octree map, which could be used for high-level tasks but with less positioning accuracy in tracking. Besides, on the basis of Dyna-SLAM [29], Cheng [36] applied the optical flow field to eliminate the dynamic feature points as soon as possible and raised a dynamic SLAM system called DM-SLAM for dynamic scenes.

As an RGB-D camera provides RGB and depth images simultaneously, in addition to the SLAM systems using deep learning above directly, there are also proposals combining deep learning and depth to filter out dynamic features. Zhong [37] determined the dynamic objects based on ORB-SLAM2 combined with depth values provided by the RGB-D camera and removed dynamic objects to perform simultaneous positioning and mapping. Li [38] and Cui [39] utilized the depth information to establish a depth filter and combined semantic information to identify moving objects. To date, the camera pose estimation accuracy of these dynamic SLAM algorithms based on deep learning is higher than those of most dynamic SLAM methods based on the mathematical model and the optical flow. However, most of them are prone to cause the over-elimination of dynamic features, which will lead to the loss of partial static features and degradation of location accuracy, eventually even the tracking loss of SLAM. Moreover, the line feature is another type of primary visual feature that is not utilized to estimate camera pose in most dynamic SLAM systems.

Generally, the dynamic SLAM methods based on deep learning theory can achieve a high camera positioning accuracy. However, as the under-over semantic segmentation caused by the deep learning algorithm commonly exists, it can lead to the loss of

visual semantic information or wrong association, which can eventually result in the accuracy degradation of camera pose estimation. Therefore, we proposed the PLD-SLAM to overcome the challenges in dynamic scenes. In detail, to avoid the under-over semantic segmentation of the deep learning network, we applied the K-Means algorithm to acquire as many static features as possible. Moreover, as the semantic segmentation occupied mass pixels, the point features were usually utilized in most existing dynamic SLAM, and line features were also used to achieve the camera pose estimation. While the majority of dynamic SLAM methods were solely conducted based on feature points, our PLD-SLAM additionally used another kind of primary feature: Lines. Thus, point and line features were utilized in our proposed RGB-D SLAM creatively, which was a major difference compared with other dynamic SLAM methods based on point features only.

3. System Introduction

In this section, PLD-SLAM will be introduced at length. This section mainly contains four aspects. First, the overall architecture framework of PLD-SLAM is presented. Second, a brief introduction of the real-time semantic segmentation model that was adopted in PLD-SLAM is presented. Then, the semantic segmentation model and K-Means clustering considering depth information for checking these potentially dynamic points and line features are described. Furthermore, the outlier detection method, which applies the epipolar constraint and depth information constraint of adjacent frames to detect and filter out dynamic feature points, is described. Finally, to obtain preferable camera pose estimation results, an optimization model with points and lines is introduced and utilized to calculate the camera pose.

3.1. Overview of PLD-SLAM

An overview of PLD-SLAM is shown in Figure 1, and Figure 2 shows the details of PLD-SLAM. The input frames of PLD-SLAM are RGB and depth images. The point features are extracted by the outstanding ORB [40] algorithm, and the line features are detected by using the LSD [41] algorithm. Then, the RGB images are segmented by the MobileNet model first, and the segmented results (e.g., walking people) are considered as potentially dynamic. All extraction point and line features that are located in the semantic regions are regarded as potentially dynamic features. To refine the potentially dynamic features reasonably and retain more static information, the K-Means clustering of depth images in the segmentation region is utilized. These line features that are not in the regions will be considered as static features. After finishing this, two checking strategies (i.e., epipolar constraint, depth difference) are utilized for identifying and rejecting the dynamic point features. Subsequently, to acquire a more accurate camera pose estimation, the remaining static feature points and lines would be used to construct the optimization model and estimate the camera pose.

Figure 1. A brief pipeline of Simultaneous Localization and Mapping based on point and line features for dynamic scenes (PLD-SLAM).

Figure 2. The details of PLD-SLAM.

The details of our PLD-SLAM are shown in Figure 2. PLD-SLAM mainly contains three threads: Tracking, Local Bundle Adjustment, and Looping Detection. In the tracking thread, segmentation and rejection of dynamic features are carried out. Then, it determines if the frame is a keyframe. If so, the frame is inserted into the keyframe sequences, and then local bundle adjustment and looping detection are carried out to reduce the accumulation of errors and drifting. If not, the frame and the next frame are utilized to calculate the camera pose. Finally, after accomplishing the above works, PLD-SLAM will judge if the frame is the last one. If the current frame is the last frame of an image sequence, the trajectory of the RGB-D camera will be output. If the current frame is not the last one, PLD-SLAM continues performing the above three threads.

3.2. Segmentation of Potentially Moving Objects

The MobileNet model is a lightweight network model put forward by Google. It is trained for object detection on COCO data based on the recent work that won the 2016 COCO challenge [42]. From the description in [13], MobileNet can achieve comparable results to other networks with only a fraction of the computational complexity and model size. Moreover, the MobileNet can be applied to various recognition tasks for efficient on-device intelligence, and it has a wide range of applications including object detection, face attributes, and large-scale geo-localization. Therefore, PLD-SLAM adopts the MobileNet model as the semantic information extraction network in the tracking thread. It is based on the depthwise separable convolutions, which is a form of factorized convolutions that factorize a standard convolution into a depthwise convolution and a pointwise convolution. For the MobileNet model, the depthwise convolution applies a filter to an input, and the pointwise convolution applies a 1×1 convolution to the output depthwise convolution. The depthwise convolution divides the standard convolution into two layers: The individual layer for filtering and another separate combination layer. This factorization significantly reduces the computation and size of the model. In most real applications, a person is often most likely to be a moving object, and such point and line features located in the person should be regarded as underlying dynamic features. Figure 3a shows the semantic segmentation result generated by the MobileNet model. The features located in the semantic segmentation region are considered as potentially dynamic features and are presented in Figure 3b.

Figure 3. (a) The pixel-wise semantic segmentation result of RGB image. (b) The underlying dynamic feature points (+) and lines (—). (c) The underlying dynamic feature points (+) and lines (—) are refined by the MobileNet model and K-Means.

The K-Means algorithm is a simple iterative clustering algorithm, which applies distance as the similarity index to search for K classes in a given data set. The center of each clustering class is obtained according to the mean value of all elements belonging to this class iteratively. For a given dataset X containing n dimensions or one-dimensional data points, as well as the number of clustering categories k , the Euclidean distance or Mahalanobis distance is selected as the similarity indicator. After finishing the K-Means clustering simultaneously, the sum of the squared distance from each element to the corresponding center of the clustering class should be minimized. According to the above descriptor, we assume that a dataset X will be divided into k classes, and each clustered x_i belongs to the category t_j . The center of the j -th clustering class is u_j . The cost function is defined as follows:

$$L = \sum_{j=1}^k \sum_{i=1}^n (x_i - u_j)^2 (t_j = j) \quad (1)$$

To identify the potential dynamic features accurately and avoid over-segmentation, we combine the K-Means algorithm with a depth image to segment the features located in the red rectangular in Figure 3a. As shown in Figure 3a, as the person in the red rectangle occupies considerable pixels, the K-Means clustering algorithm is used to segment the person and background according to the depth information. In this paper, the clustering classes are divided into two categories (i.e., the potentially dynamic features and static features). The pixel number of potentially dynamic information (e.g., person) is much larger than that of static background information. Based on this basic assumption, we divide the class with the larger number of elements into potentially dynamic features, and the rest into static features. After finishing the semantic information segmented by the MobileNet model, the interval of pixel coordinates related to the semantic information (e.g.,

person) in the RGB image is obtained. Then, these point and line features located in the pixel coordinate range are considered as the underlying dynamic features. The underlying dynamic features directly using the MobileNet are shown in Figure 3b. According to the combination MobileNet model with the K-Means algorithm, the segmentation result of dynamic features can be seen in Figure 3c.

To retain more static features and avoid over-segmentation as soon as possible, the K-Means clustering algorithm combined with depth information is used to obtain more appropriate potentially dynamic features. Given the clustering class $\mathbf{M}'$ with the maximum number in ascending order first, the depth threshold of S and B are obtained according to Equation (2):

$$\begin{cases} S = \mathbf{M}'[m * 0.1] \\ B = \mathbf{M}'[m * 0.9] \end{cases} \quad (2)$$

where m is the number of elements in $\mathbf{M}'$, and $[\cdot]$ represents the element value corresponding to the element index in the set. By using the value of S and B , the potentially dynamic and static features are distinguished by the corresponding depth information within points and lines. If the depth value is located in the interval of the preset thresholds (i.e., $[S, B]$), its corresponding point feature or line feature is considered a potentially dynamic feature. If the opposite is true, the extraction point or line is regarded as the static feature. Therefore, the above potentially dynamic feature points and lines can be refined and filtered according to the above principle.

An overview of the aforementioned algorithm is shown in Figure 4. As shown in Figure 4, the RGB image is utilized to segment and acquire the semantic information first (i.e., potentially dynamic region), and these point and line features are extracted by the ORB and LSD algorithm. Then, the K-Means clustering considering depth information is applied to divide the known semantic segmentation region into two categories. As stated above, the potentially dynamic region occupies many more pixels than the static feature in the semantic region. The depth threshold values of the potentially dynamic region can be determined by applying the clustering category with a larger number of elements. When extracting these point and line features, the depth values within the point and line features can be extracted from the depth image. Finally, it will be determined whether the depth values of potentially dynamic points and lines confirm the interval range of threshold values. If so, these point and line features will be treated as potentially dynamic features. If not, these points and lines will be regarded as static features.

Figure 4. Extraction of potentially dynamic point and line features.

3.3. Dynamic Feature Points Extraction

After completing the segmentation of potentially dynamic feature points and lines, we assume that the potentially dynamic features are dynamic. However, the potentially dynamic features may not be real dynamic features. In this paper, as the inherent characters of line features, the remaining line features filtered out by Section 3.2 will be considered as the static features. As for the potentially dynamic point features, two consistency checking strategies need to be done to determine whether the potentially dynamic features are the motion point features. The epipolar constraint of frames is used to detect if the potentially dynamic feature points are real dynamic features first. The depth information consistency of feature points in adjacent frames is able to detect the dynamic feature points again when taking the depth information provided by an RGB-D camera into account.

3.3.1. Epipolar Constraint Consistency Check

After finishing the segmentation of potentially dynamic objects and obtaining the initial static features, the system matches the static feature points in the current frame and reference frame, and obtains the matching feature points set Q . Then, the fundamental matrix of camera motion is calculated by using the matching points set of Q . For each of the above potentially dynamic feature points, the epipolar constraint (shown in Figure 5) is used to identify the dynamic feature points first. In Figure 5, o - xyz and o' - $x'y'z'$ denote the coordinate system in the reference frame and current frame, respectively, and l and l' are the polar lines. The epipolar constraint model is expressed in Equation (3), where p_i and q_i represent the homogeneous coordinates of feature points in the current frame and the reference frame, respectively. F represents the fundamental matrix, which is a 3×3 matrix and calculated by the OpenCV library.

$$p_i^T \cdot F \cdot q_i = 0 \quad (3)$$

Figure 5. Epipolar constraint of adjacent frames.

For each match of potentially dynamic point features in the current frame and reference frame, the corresponding polar line equation is calculated by using the fundamental matrix and denoted as $[a, b, c]^T$. The distance from the image point to the polar line is calculated as follows:

$$D = \frac{|p_i^T \cdot F \cdot q_i|}{\sqrt{(a^2 + b^2)}} \quad (4)$$

where D represents the distance and $|\cdot|$ is the norm of a vector. We preset a certain threshold value as 1.0 and calculate the distance from each potentially dynamic feature point to the polar line according to Equation (4). If the distance exceeds the preset threshold value, the potentially dynamic point feature pair is regarded as a real dynamic feature point pair. If not, it is classified as a static feature point pair.

3.3.2. Depth Constraint Consistency Check

As an RGB-D camera can provide RGB and depth images simultaneously, we also use the depth difference of the matching feature points of adjacent frames to identify the feature points as dynamic or static. If the depth value of the feature point in the reference frame is equal to that of the feature point in the current frame under a common coordinate system, the feature point is considered a static feature point. The assumption is shown in Figure 6, where $depth$ is the depth information of a feature point in the reference frame and $depth'$ is the corresponding depth information in the current frame. In Figure 6, $o-xyz$ and $o'-x'y'z'$ denote the coordinate system in the reference frame and current frame, respectively, and p and p' are the matching feature points. First, we still use the previous static feature points of the reference frame and corresponding depth image to acquire the corresponding 3D points of the reference frame. Then, according to the space resection of photogrammetry, the acquired 3D points of the reference frame and the matched feature points of the current frame are applied to calculate the pose of the current frame and the reference frame. According to the calculated pose and depth image of the current frame, this paper recovers the 3D points of the current frame and transforms the 3D matching point pair to the reference frame. The distance difference of the matching point in 3D space is calculated by Equation (5), where X_{ref} is the reference frame 3D coordinates and X_{curr} is the current frame 3D coordinates.

Figure 6. The depth constraint of adjacent frames.

$$Error = \sqrt{\|X_{ref} - X_{curr}\|} \quad (5)$$

The matching point features error set $Error_p$ is obtained and sorted in descending order according to the coordinate difference of each pair of matching points. In this paper, 20% of the number of elements in the matching point error set $Error_p$ are regarded as the outliers, i.e., the element value index of the set $Error_p$ is considered to be deleted when it ranks in the first 20% of the total. From those described above and the epipolar constraint mentioned in Section 3.3.1, the whole process of the dynamic feature point recognition and elimination algorithm is achieved, and the final obtaining of static feature points and lines is shown in Figure 7.

Figure 7. Extraction result of point and line features. Raw color image: (a,d,g); the extraction of lines and points: (b,e,h); inlier point and line features: (c,f,i).

3.4. Optimization Model of Points and Lines

After acquiring the static feature points and lines, these feature points and lines will be used to establish a new geometric optimization model of solving camera pose. In this section, we describe the details of the geometric constraint optimization model of points and lines.

3.4.1. Optimization Model of Static Feature Points

For the static feature points, as an RGB-D camera can provide depth information, depth information is considered a geometric constraint in our paper. Unlike most RGB-D SLAM methods based on point features, in addition to the 2D re-projection error of point features, the constraint term of per-depth is applied to calculate camera pose. As shown in Figure 6, the depth information of matching point features in adjacent frames should be equal to each other in a unified coordinate frame system. Based on this thought, the pose solution model of re-projection and per-depth can be established. First, according to matching point features, the error functions related to the re-projection error of point features are established. Then, the constraint related to depth value is established too.

Using two terms of geometric constraints, our geometric optimization model of point features is established and expressed as Equation (6):

$$\begin{cases} err_p = K \cdot T(\zeta) \cdot P - p \\ err_{dep} = depth - P_z \end{cases} \quad (6)$$

where ζ is a six-dimensional vector of Lie algebra that represents the camera pose and $T(\zeta)$ represents the transformation matrix from the current coordinate system to the reference coordinate system. K represents the intrinsic matrix of the RGB camera and p is the pixel coordinate of the extraction point feature. P is the 3D coordinate of the point in the current coordinate frame and $depth$ is the depth value of the point feature in the reference frame. P_z is the z component of the 3D point converted to the reference coordinate system. err_p and err_{dep} are the re-projection error of point features and per-depth information, respectively.

3.4.2. Optimization Model of Static Feature Lines

For the static line features, the optimization model related to lines includes three terms of geometric constraints. The first is the traditional 2D re-projection error. As the 3D lines can be recovered by using depth images, the plane normal vector and the distance from the camera optical center to a 3D line are considered as the other two constraint terms. First, the cross product of the plane normal vector containing the 3D line in the adjacent frames is close to 0. Meanwhile, the distance from the camera optical center to a 3D line should be equal to each other in frames under a unified coordinate system. The mentioned geometric constraints of lines constitute our pose optimization model of line features.

For the first constraint condition, we still adopt the minimization of the distances from the re-projected endpoints of a 3D line to the 2D line features. The constraint of 2D line features is established as follows: First, the 2D line equation in the image plane is computed by using pixel coordinates of line features and is denoted in Equation (7):

$$\lambda_{1 \times 3} = \frac{[p]^\wedge q}{|[p]^\wedge q|} \quad (7)$$

where λ_i ($i = 0, 1, 2$) are the coefficients of the line equation and λ is a unit vector. p and q are the homogeneous coordinates of line features endpoints and $[\cdot]^\wedge$ is a skew-symmetric matrix of a three-dimensional vector. According to the computed line equation in the image plane and the pixel coordinate of re-projection line features, the error equation is established using Equation (8), where err_s and err_e denote the re-projection error of the line feature. (x_p, y_p) and (x_q, y_q) are the pixel coordinates of re-projected line endpoints.

$$\begin{cases} err_s = \lambda_0 \cdot x_p + \lambda_1 \cdot y_p + \lambda_2 \\ err_e = \lambda_0 \cdot x_q + \lambda_1 \cdot y_q + \lambda_2 \end{cases} \quad (8)$$

Next, another two constraint conditions are introduced briefly and shown in Figure 8. In Figure 8, P and Q are endpoints of the 3D lines, while p and q are endpoints of the 2D line feature of the reference frame. p' and q' are the 2D line features of the current frame. $o\text{-}xyz$ is the reference frame coordinate system and $o'\text{-}x'y'z'$ is the current frame coordinate system. oPQ is the plane containing the camera optical center and 3D line (PQ), and $o'PQ$ is another plane that contains the camera optical center and 3D line PQ . n' and n are the normal of the two planes, respectively. d and d' denote the distance from the camera optical center to the 3D line PQ . As described in Figure 8, according to space geometric knowledge, the cross product of n and n' should be a zero vector under a unified coordinate system, and d should be equal to d' . According to the description, the two terms of constraints are expressed in Equation (9). For details, please refer to our previous work in [43].

$$\begin{cases} [n']^\wedge n = 0 \\ d' = d \end{cases} \quad (9)$$

Figure 8. The illustration of the optimization model of line features.

4. Experimental Results and Discussion

In this section, to verify our method, we have performed a series of experiments using the public datasets with ground-truth. For the quantitative evaluation of visual SLAM, the easy-to-use evaluation tool of python (<https://vision.in.tum.de/data/datasets/rgbd-dataset/tools>, accessed on 19 August 2020) is used to evaluate the SLAM systems and PLD-SLAM. The metric index of the absolute camera trajectory error is used to evaluate our SLAM method. The root-mean-square errors (RMSE) of the absolute translation error (ATE) is considered as the main evaluation criterion and applied to measure the difference between the ground-truth and estimated trajectory. RMSE is an important SLAM evaluation indicator, and the RMSE of the ATE reflects the accuracy and the performance of SLAM. For the well-known SLAM systems (e.g., ORB-SLAM2 [10]), the RMSE is usually utilized to evaluate the overall performance of SLAM.

4.1. TUM Dataset

The TUM dataset is the well-known RGB-D benchmark to evaluate the SLAM system [41]. The datasets consist of various indoor scene sequences recorded using an RGB-D camera of Microsoft Kinect1. The image sequences are recorded in real indoor scenes at a frame rate of 30 Hz with a 640×480 size and provide the ground-truth trajectories captured by using a motion-capture system with higher accuracy. As a result, many well-known SLAM schemes have adopted the TUM dataset as their benchmark dataset, such as ORB-SLAM2 [10]. In the *fr3* sequences (e.g., *sitting_xyz*, *sitting_half_sphere*, *sitting_rpy*, *sitting_static*, *walking_xyz*, *walking_half_sphere*, *walking_rpy*, *walking_static*), the RGB-D camera is always facing the table. The camera movement trajectory and the person's movement mode are different in the sequences, which are typical dynamic scene sequences. Among the sequences of *fr3*, “*sitting*” belongs to the low-dynamic sequences and “*walking*” belongs to the high-dynamic sequences. All experiments are performed on a desktop computer with an Intel®Core™ i5-8250U@CPU at 1.6 GHz. The software utilized in our proposed PLD-SLAM includes the Linux operating system of Ubuntu 16.04, OpenCV 3.2.0, Eigen 3.0, g2o, and C++ language.

In experiments, the number of point and line features are preset as 1000 and 300, respectively. The experimental results of using the TUM RGB-D datasets are shown in Table 1. We compared PLD-SLAM with dynamic SLAM methods (e.g., DS-SLAM [34], DM-SLAM [36], Dyna-SLAM [29], the optical-flow dynamic SLAM [25] abbreviated as OPF-SLAM, and the state-of-the-art ORB-SLAM2 [10]). The quantitative experimental results are presented in Table 1. All statistics are from [29,32,36] and our real experiments. “×” means tracking loss at some point or a significant portion of the sequences not being processed by the SLAM scheme. “—” means the SLAM methods do not provide the related experimental results. The smallest value marked in bold denotes the optimal result of camera pose estimation.

Table 1. Quantitative comparison results of the absolute trajectory error (ATE: m) in meters for the experiments. It can be seen that our method exhibits a significantly improved performance compared to ORB-SLAM2 in high-dynamic scenes in terms of the ATE.

	Sequence	DS-SLAM	OPF-SLAM	DM-SLAM	Dyna-SLAM	ORB-SLAM2	PLD-SLAM
low dynamic scenes	fr3_s_static	0.0065	0.0134	0.0063	0.0064	0.0083	0.0063
	fr3_s_xyz	—	0.0130	—	0.013	0.0095	0.0092
	fr3_s_rpy	0.0187	0.0160	0.0230	0.0302	0.019	0.0220
	fr3_s_half	0.0148	0.0257	0.0178	0.0191	0.035	0.0145
high dynamic scenes	fr3_w_static	0.0081	0.041	0.0079	0.0080	0.390	0.0065
	fr3_w_xyz	0.0247	0.306	0.0148	0.0158	0.614	0.0144
	fr3_w_rpy	0.4442	0.104	0.0328	0.0402	0.937	0.2212
	fr3_w_half	0.0303	0.307	0.0274	0.0274	0.789	0.0261

To fully prove the effectiveness of camera pose calculation using point and line features, we also conducted verification tests on TUM RGB-D datasets by using our RGB-D SLAM with point features only. The quantitative experimental results are presented in Table 2. We mainly adopted the RMSE of ATE to compare the accuracy and robustness of our RGB-D SLAM based on point and line features. The smallest values marked in bold denote the most accurate result in the camera pose estimation. “×” still means that tracking is lost at some point or a significant portion of the sequences is not processed. According to the experimental comparison results presented in Table 2, our PLD-SLAM based on point features and line features achieved improvements in 7 of 8 dynamic sequences. According to the quantitative experimental results shown in Tables 1 and 2, it can be concluded that our proposed SLAM based on point and line features is feasible and can perform better than or comparably to the outstanding SLAM methods for dynamic scenes.

Table 2. Quantitative comparison results of the absolute trajectory error (ATE: m) in meters for the experiments. It can be seen that PLD-SLAM based on point and features exhibits significant robustness and improvements compared to PLD-SLAM based merely on point features.

KERRYPNX	Sequence	PLD-SLAM(Point Only)	PLD-SLAM(Point+Line)
low dynamic scenes	fr3_s_static	0.007	0.0063
	fr3_s_xyz	0.010	0.0092
	fr3_s_rpy	0.014	0.0220
	fr3_s_half	0.0146	0.0145
high dynamic scenes	fr3_w_static	0.008	0.0065
	fr3_w_xyz	0.015	0.0144
	fr3_w_rpy	×	0.2212
	fr3_w_half	0.027	0.0261

For visual comparison, some intuitive results and the corresponding ATE of trajectories estimated by ORB-SLAM2 [10], OPF-SLAM [25], and PLD-SLAM are shown in Figures 9 and 10. In the two groups of figures, the black lines represent the ground-truth trajectories. The blue lines denote the estimated trajectories, while the red lines represent the error metric of the translation of ATE. The yellow rectangles indicate that the tracking loss of features occurs in partial frames of data sequences. Figures 9 and 10 show the RGB-D camera trajectories of ORB-SLAM2, OPF-SLAM, and PLD-SLAM with reference to the ground truth of three sequences. Figure 9 shows the selected ATE plots for high-dynamic scenes. From the intuitive visual comparison, camera tracking loss of frames occurred in the OPF-SLAM. Although ORB-SLAM2 performed more robustly in high-dynamic scenes than OPF-SLAM, the accuracy of camera pose estimation was lower. According to the visual comparison in Figure 9, as PLD-SLAM can calculate camera poses accurately by eliminating the negative effects of dynamic features, the accuracy of camera trajectories

improved significantly by comparison with ORB-SLAM2 and OPF-SLAM. Figure 10 shows the selected ATE plots in the low-dynamic scene, and our PLD-SLAM can still achieve robust and better performances (e.g., *fr3_s_half*). Considering the quantitative results in Table 1 and the visual comparisons in Figures 9 and 10 comprehensively, PLD-SLAM can achieve robust performance in both high- and low-dynamic sequences.

Figure 9. Plots of the ATE [m] for high-dynamic scenes: *fr3_w_xyz*, *fr3_w_half*. (a,b) The experiments performed with ORB-SLAM2; (c,d) the experiments performed with OPF-SLAM; (e,f) the experiments performed with PLD-SLAM.

Figure 10. Plots of the ATE [m] for high-dynamic scenes: *fr3_s_xyz*, *fr3_s_half*. (a,b) The experiments performed with ORB-SLAM2; (c,d) the experiments performed with OPF-SLAM; (e,f) the experiments performed with PLD-SLAM.

4.2. Real Scene

To prove the effectiveness of PLD-SLAM in real scenes, we also performed the relevant experiments. The real scene dataset recorded the common office scenarios in which people work naturally and the RGB-D camera did not follow certain motion patterns. We adopted an opti-track motion capture system to generate these ground truths of RGB-D camera trajectories, and utilized the X-Tion of the RGB-D camera to generate the image sequences [44]. These image sequences were recorded at a frame rate of 30 Hz with a size of 640×480 pixels.

The quantitative experimental results are presented in Table 3. We tested our proposed dynamic SLAM on two image sequences captured in a real scene. The smallest value marked in bold denotes the optimal result of the camera pose estimation. According to the experimental results of our dynamic SLAM in the real scenes, tracking loss of partial frames occurred in the OPF-SLAM and is marked in a yellow rectangle in Figure 11a. Although

ORB-SLAM2 performed experiments in the two image sequences, the accuracy of camera pose estimation is low. According to the results of Table 3 and Figure 11, it is easy to find that our proposed method does achieve higher accuracy in camera pose estimation and robust performance in the real scene.

Table 3. Quantitative comparison results of the absolute trajectory error (ATE: m) in meters for the experiments. It can be seen that our method exhibits a significantly improved performance compared to ORB-SLAM2 and OPF-SLAM in the real scenes in terms of the ATE.

Sequences	PLD-SLAM		ORB-SLAM2		OPF-SLAM	
	RMSE	Std	RMSE	Std	RMSE	Std
Sequences 01	0.0279	0.1439	0.582	1.557	0.088	0.1984
Sequences 02	0.0458	0.0189	0.361	0.185	0.305	0.223

Figure 11. Plots of the ATE [m] for real dynamic scenes. (a,d) The results performed with OPF-SLAM; (b,e) the results performed with ORB-SLAM2; (c,f) the results performed with PLD-SLAM.

4.3. Runtime Analysis

The efficiency is an important indicator for SLAM, and we also investigate the efficiency performance of PLD-SLAM. The average time of different components on the TUM datasets was calculated. The semantic segmentation of the MobileNet model and K-Means clustering are most time-consuming. What is more, as PLD-SLAM utilized the point and line features to construct an optimization model and optimize the camera pose, it also consumed a little more time. We tested the time consumption for some major modules on our laptop with the general configuration CPU. The calculation results of several parts are shown in Table 4. The average time consumed of the semantic segmentation and K-Means algorithm in each frame is 248 ms, and the check of dynamic features and pose optimization of points and lines need 32 and 61 ms, respectively. As our PLD-SLAM operated on an inefficient CPU and incorporated the computationally expensive MobileNet network, it did not implement in real-time. Because of this limitation of PLD-SLAM, the efficiency of PLD-SLAM will be improved in our future work. There is another feasible solution

to address this problem in the future. An efficient approach is to deploy the semantic segmentation module on a server with a higher hardware configuration to accelerate the running speed of SLAM.

Table 4. The average computational time of different components in PLD-SLAM.

Module	Dynamic Features Detection	Optimization of Points and Lines	Potentially Dynamic Segmentation
Time (unit: ms)	32	61	248

4.4. Discussion

In the experimental results of Table 1, PLD-SLAM and other dynamic SLAM methods all mainly perform the experiments. In comparison to these classical SLAM systems for dynamic scenes [29,34,36], PLD-SLAM achieved some improvements or acquired comparable accuracy in the low-dynamic and high-dynamic scenes. Most dynamic SLAM methods (e.g., DS-SLAM, DM-SLAM, Dyna-SLAM) are based on point features to calculate the camera pose, while PLD-SLAM achieved the pose estimation using the point and line features. Differing from most dynamic SLAM using point features only, PLD-SLAM adopted an optimization model with points and lines to calculate a more accurate camera pose. Therefore, PLD-SLAM could perform better in most sequences. As the visual camera is in motion, the dynamic SLAM based on the optical flow field cannot detect the dynamic point features effectively. Therefore, the dynamic SLAM based on the optical flow cannot acquire a better performance, leading to tracking loss. Compared to the ORB-SLAM2 [10], PLD-SLAM obtained an obvious improvement in low-dynamic and high-dynamic scenes. As ORB-SLAM2 cannot distinguish static and dynamic point features, ORB-SLAM2 could achieve a comparable accuracy in some low-dynamic scenes (e.g., *fr3_s_static*, *fr3_s_rpy*). Nevertheless, for the high-dynamic scenes, ORB-SLAM2 could not perform better generally. For the real scenes, we also designed and performed the comparison experiments. As shown by the experimental results in Figure 11 and Table 3, PLD-SLAM achieved a better performance than ORB-SLAM2 and OPF-SLAM. However, for the sequences with camera rotation (e.g., *fr3_s_rpy*, *fr3_w_rpy*), PLD-SLAM had a mildly poor performance, and improvement work will be done in the future.

The following conclusions from the experimental results of Tables 1–3 and Figures 9–11 can be made: PLD-SLAM utilized the MobileNet and K-Means to identify dynamic feature points and lines, and it is also different from most dynamic SLAM based on feature points only. As PLD-SLAM adopted an optimization model of points and lines to calculate the camera pose, PLD-SLAM obtained more remarkable improvements than other dynamic SLAM methods. In addition, compared to ORB-SLAM2 and other dynamic SLAM algorithms in the low-dynamic scene, although PLD-SLAM achieved some slight improvements in location accuracy, it also achieved better or comparable performances. Finally, according to the experimental results above, it was demonstrated that PLD-SLAM obtained a higher accuracy with strong robustness and could be applied in the pose estimation model of an RGB-D camera for dynamic scenes.

5. Conclusions

In this paper, we presented a new RGB-D SLAM based on point and line features in the dynamic scenes. We combined the semantic segmentation network MobileNet and K-Means algorithm to filter out dynamic features in the scene as far as possible. The epipolar constraint and depth difference of frames were applied to eliminate dynamic point features. To obtain a more accurate and robust camera pose, points and lines were utilized to calculate the camera pose. After obtaining static point and line features, we adopted the optimization model of points and lines and improved the performance of robustness and accuracy in the dynamic scenes. These comprehensive experiment results proved that PLD-SLAM improved the accuracy and robustness of camera pose estimation

in dynamic scenes. We also compared PLD-SLAM with classic dynamic SLAM methods, and the results showed that PLD-SLAM achieved an obvious improvement and robustness in most of the dynamic image sequences. Although our proposed PLD-SLAM showed some poor performances on some extra dynamic scene sequences, it achieved comparable accuracy and performances too.

However, despite the good performance, PLD-SLAM also requires some ongoing work. For instance, the types of objects that can be recognized in the mobile network are restricted, which limits its accuracy of pose estimation. In future work, we will improve the real-time performance of PLD-SLAM by optimizing the operation speed of PLD-SLAM. Moreover, we will use the plane feature of the indoor scene and extend it into PLD-SLAM (point-line-plane), which can acquire a more robust pose estimation in dynamic scenes.

Author Contributions: Chenyang Zhang and Teng Huang conceived the idea and designed the methods; Chenyang Zhang and Xuefeng Yi performed the experiments; Chenyang Zhang and Teng Huang analyzed the data; Chenyang Zhang, Teng Huang, and Rongchun Zhang wrote this paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Postgraduate Research and Practice Innovation Program of Jiangsu Province (KYCX20_0485) and supported by the Fundamental Research Funds for the Central Universities (B200203106). This research was also part-funded by the National Natural Science Foundation of China (41901401).

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: <http://vision.in.tum.de/data/datasets/rgbd-dataset> (accessed on 12 March 2021).

Acknowledgments: We thank the Computer Vision Group from the Technical University of Munich and Imperial College London and National University for making the RGB-D SLAM dataset publicly available.

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

References

1. Mouats, T.; Aouf, N.; Chermak, L.; Richardson, M.A. Thermal stereo odometry for UAVs. *IEEE Sens. J.* **2015**, *11*, 6335–6347. [CrossRef]
2. Smith, R.; Self, M.; Cheeseman, P. Estimating uncertain spatial relationships in robotics. In Proceedings of the IEEE International Conference on Robotics and Automation, Raleigh, NC, USA, 31 March–3 April 1987; p. 850.
3. Durrant, W.; Bailey, T. Simultaneous localization and mapping: Part I. *IEEE Robot. Autom. Mag.* **2006**, *13*, 99–110. [CrossRef]
4. Lu, S.; Liu, D. A Survey of Research Situation on Navigation by Autonomous Mobile Robot and Its Related Techniques. *Trans. Chin. Soc. Agric. Mach.* **2002**, *33*, 112–116.
5. Kaichang, D.I.; Wenhui, W.A.; Hongying, Z.H.; Zhaoqin, L.I.; Runzhi, W.A.; Feizhou, Z.H. Progress, and Applications of Visual SLAM. *Acta Geod. Cartogr. Sin.* **2018**, *47*, 770–779.
6. Di, K.; Xu, F.; Wang, J.; Agarwal, S.; Brodyagina, E.; Li, R.; Matthies, L. Photogrammetric processing of rover imagery of the 2003 Mars Exploration Rover mission. *ISPRS J. Photogramm. Remote Sens.* **2008**, *63*, 181–201. [CrossRef]
7. Wang, B.; Zhou, J.; Tang, G.S. Research on visual localization method of lunar rover. *Sci. China Inf. Sci.* **2014**, *44*, 452–460.
8. Liu, Z.; Di, K.; Li, J. Landing site topographic mapping and rover localization for Chang’e-4 mission. *Sci. China Inf. Sci.* **2020**, *63*, 140901. [CrossRef]
9. Klein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 225–234.
10. 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]
11. Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-scale direct monocular SLAM. In Proceedings of the European Conference Computer Vision (ECCV), Zurich, Switzerland, 6–12 September 2014; pp. 834–849.
12. 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.
13. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobilevision application. *arXiv* **2017**, arXiv:1704.04861.
14. 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, Vilamoura, Portugal, 7–12 October 2012; pp. 573–580.

15. Wang, C.C.; Thorpe, C.; Thrun, S.; Hebert, M.; Durrant-Whyte, H. Simultaneous Localization, Mapping and Moving Object Tracking. *Int. J. Robot. Res.* **2007**, *26*, 889–916. [\[CrossRef\]](#)
16. Moratuwage, D.; Vo, B.-N.; Wang, D. Collaborative multi-vehicle SLAM with moving Object tracking. In Proceedings of the 2013 IEEE International Conference Robotics & Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 5702–5708.
17. Zou, D.; Tan, P. CoSLAM: Collaborative visual SLAM in dynamic environments. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 354–366. [\[CrossRef\]](#)
18. 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. [\[CrossRef\]](#)
19. 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.
20. 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.
21. 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. [\[CrossRef\]](#)
22. 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. [\[CrossRef\]](#)
23. Wang, R.; Wan, W.; Wang, Y.; Di, K. A New RGB-D SLAM Method with Moving Object Detection for Dynamic Indoor Scenes. *Remote Sens.* **2019**, *11*, 1143. [\[CrossRef\]](#)
24. Shimamura, J.; Morimoto, M.; Koike, H. Robust vSLAM for Dynamic Scenes. In Proceedings of the MVA2011 IAPR Conference on Machine Vision Applications, Nara, Japan, 13–15 June 2011; pp. 344–347.
25. Cheng, J.; Sun, Y.; Meng, M.Q.H. Improving monocular visual SLAM in dynamic environments: An optical-flow-based approach. *Adv. Robot.* **2019**, *33*, 576–589. [\[CrossRef\]](#)
26. 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.
27. Liu, G.; Zeng, W.; Feng, B.; Xu, F. DMS-SLAM: A General Visual SLAM System for Dynamic Scenes with Multiple Sensors. *Sensors* **2019**, *19*, 3714. [\[CrossRef\]](#) [\[PubMed\]](#)
28. Kim, D.H.; Han, S.B.; Kim, J.H. Visual odometry algorithm using an RGB-D sensor and IMU in a highly dynamic environment. In *Robot Intelligence Technology and Applications 3*; Springer: Cham, Switzerland, 2015; pp. 11–26.
29. Bescos, B.; Fàcil, J.M.; Civera, J.; Neira, J. DynaSLAM: Tracking, mapping, and in painting in dynamic scenes. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4076–4083. [\[CrossRef\]](#)
30. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
31. Yang, S.; Wang, J.; Wang, G.; Hu, X.; Zhou, M.; Liao, Q. Robust RGB-D SLAM in a 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.
32. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *6*, 1137–1149. [\[CrossRef\]](#)
33. Wang, R.; Wang, Y.; Wan, W.; Di, K. A Point-Line Feature based Visual SLAM Method in Dynamic Indoor Scene. In Proceedings of the 2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS), Wuhan, China, 22–23 March 2018; pp. 1–6.
34. Yu, C.; Liu, Z.; Liu, X.J.; Xie, F.; Yang, Y.; Wei, Q.; Fei, Q. Ds-slam: A semantic visual slam towards dynamic environments. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1168–1174.
35. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [\[CrossRef\]](#) [\[PubMed\]](#)
36. Cheng, J.; Wang, Z.; Zhou, H.; Li, L.; Yao, J. DM-SLAM: A Feature-Based SLAM System for Rigid Dynamic Scenes. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 202. [\[CrossRef\]](#)
37. 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.
38. Li, S.; Lee, D. RGB-D SLAM in Dynamic Environments using Static Point Weighting. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2263–2270. [\[CrossRef\]](#)
39. Cui, L.; Ma, C. SDF-SLAM: Semantic Depth Filter SLAM for Dynamic Environments. *IEEE Access* **2020**, *8*, 95301–95311. [\[CrossRef\]](#)
40. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2564–2571.
41. Grompone, V.; Jakubowicz, J.; Morel, J.; Randall, G. LSD: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 722–732.

-
42. Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Murphy, K. Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv* **2016**, arXiv:1611.10012.
 43. Zhang, C. PL-GM: RGB-D SLAM with a Novel 2D and 3D Geometric Constraint Model of Point and Line Features. *IEEE Access* **2021**, *9*, 9958–9971. [[CrossRef](#)]
 44. Cheng, J.; Zhang, H.; Meng, M.Q. Improving Visual Localization Accuracy in Dynamic Environments Based on Dynamic Region Removal. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1585–1596. [[CrossRef](#)]