Article

DGFlow-SLAM: A Novel Dynamic Environment RGB-D SLAM without Prior Semantic Knowledge Based on Grid Segmentation of Scene Flow

Fei Long , Lei Ding * and Jianfeng Li

College of Computer Science and Engineering, Jishou University, Jishou 416000, China
* Correspondence: dinglei@jsu.edu.cn

Abstract: Currently, using semantic segmentation networks to distinguish dynamic and static key points has become a mainstream designing method for semantic SLAM systems. However, the semantic SLAM systems must have prior semantic knowledge of relevant dynamic objects, and their processing speed is inversely proportional to the recognition accuracy. To simultaneously enhance the speed and accuracy for recognizing dynamic objects in different environments, a novel SLAM system without prior semantics called DGFlow-SLAM is proposed in this paper. A novel grid segmentation method is used in the system to segment the scene flow, and then an adaptive threshold method is used to roughly detect the dynamic objects. Based on this, a deep mean clustering segmentation method is applied to find potential dynamic targets. Finally, the results of grid segmentation and depth mean clustering segmentation are jointly used to find moving objects accurately, and all the feature points of the moving objects are removed on the premise of retaining the static part of the moving object. The experimental results show that on the dynamic sequence dataset of TUM RGB-D, compared with the DynaSLAM system with the highest accuracy for detecting moderate and violent motion and the DS-SLAM with the highest accuracy for detecting slight motion, DGflow-SLAM obtains similar accuracy results and improves the accuracy by 7.5%. In addition, DGflow-SLAM is 10 times and 1.27 times faster than DynaSLAM and DS-SLAM, respectively.

Keywords: simultaneous localization and mapping; no prior semantic knowledge; dynamic objects; grid segmentation method

check for updates

Citation: Long, F.; Ding, L.; Li, J. DGFlow-SLAM: A Novel Dynamic Environment RGB-D SLAM without Prior Semantic Knowledge Based on Grid Segmentation of Scene Flow. *Biomimetics* **2022**, *7*, 163. https://doi.org/10.3390/biomimetics7040163

Academic Editor: Ameer Hamza Khan

Received: 25 August 2022
Accepted: 9 October 2022
Published: 13 October 2022

1. Introduction

Currently, visual SLAM has been widely applied in many areas, such as unmanned driving [1,2], AR [3], and intelligent robots [4]. The method of the feature-based visual SLAM systems uses a camera sensor to acquire the image and reconstruct the key points into 3D points based on static assumption. Then, the pose of the camera or robot is solved under strong static constraints, which provides the robot with the ability to map the surrounding environment and the ability to display self-orientation. Currently, some advanced SLAM algorithms have achieved remarkable results, such as ORB-SLAM2 [5] and DSO [6]. However, most of the dynamic objects existing in the real environment are in complex or harsh environments, and the dynamic objects can easily lead to the failure of camera pose estimation because of a lack of reliable static key points. Therefore, reducing the negative impact of dynamic objects on the visual SLAM systems and ensuring the robustness and real-time performance of visual SLAM in complex environments are of great significance for the application of robots in complex dynamic environments.

Currently, the main methods for solving the operation of SLAM systems in dynamic environments are divided into three categories: the pure geometry-based methods [7,8], the pure semantic-information-based methods [9], and the hybrid methods combining semantic information and geometry [10].

The geometric methods have reliable confidence in general cases but are less effective in some special cases. The epipolar geometry method will make the algorithm ineffective when the camera moves along the epipolar direction [11]. Another is the RANSAC method, which requires that most of the key points are static points. When the dynamic points reach a certain number, the RANSAC method cannot obtain the correct result [12].

With the development of deep learning, methods based on semantic information have become the mainstream methods for identifying dynamic objects in SLAM systems [9,10,13]. The semantic segmentation network can find a great quantity of possible moving objects, such as people, cars, and animals, and can give pixel-level masks. However, semantic methods can only provide potential motion information and cannot really identify whether there are moving objects in the found objects. A static book may also be in motion, and a dynamic human body may also be at a stationary moment. According to the paper [10], potentially dynamic objects are those objects that are sometimes in a static state and sometimes in a dynamic state. Moreover, the potential dynamic targets can be simply divided into two categories. One category of potential dynamic targets is considered to be moveable targets based on prior knowledge, which may move now or in the future, such as people, cats, dogs, cars, etc. The other category of potential dynamic targets is considered to be static targets based on prior knowledge, which have the possibility of movement, such as books, computers, chairs, etc. To enhance the performance of recognizing dynamic objects, the current methods combining geometric and semantic methods have been widely used in SLAM systems [10,14–16].

In these systems, the geometric method is generally used as a supplementary verification method for the semantic method. Namely, the semantic method is firstly used to find the potential dynamic objects, and the geometric method is then used to accurately find the dynamic objects. This method performs very well in some special scenarios. Currently, the methods combining geometric and semantic information have become mainstream for SLAM systems [10,14–17]. However, the semantic segmentation networks have two problems: The first problem is that most effective semantic segmentation networks need expensive computation and possess bad real-time performance, while the lightweight semantic segmentation networks possess poor segmentation accuracy, although they have good real-time performance. The second problem is that the semantic segmentation networks cannot detect the dynamic objects without their prior semantic knowledge [15].

Most of the current methods using semantic information perform semantic segmentation on the key frames and compute dynamic or static probability of the key points on the non-key frames and transfer them to the next key frame. However, this method may reduce the accuracy, although it meets the real-time requirements of SLAM systems. In contrast, our proposed method combing k-means clustering segmentation and grid segmentation can simultaneously perform dense segmentation in each frame under the premise of ensuring the real-time operation of SLAM systems. On the other hand, to solve the recognition of moving objects without their prior semantic knowledge, we propose a scene-flow-based method to distinguish dynamic–static regions, which enables visual SLAM systems to work well without relying on prior semantic segmentation networks.

The main contributions of this paper can be summarized as:

(1) A novel grid segmentation method is proposed to segment the scene flow, then an adaptive threshold method is designed to roughly detect the dynamic objects. Based on this, a deep mean clustering segmentation method is applied to find potential dynamic targets. This proposed segmentation method can guarantee that each frame can be executed in real time.

(2) The results of grid segmentation and deep mean clustering segmentation are jointly used to find the moving objects accurately, and all the feature points of the moving part of the moving objects are removed on the premise of retaining the static part of the moving object.

(3) A novel SLAM system called DGFlow-SLAM is proposed by integrating our approach into the RGBD-SLAM system and verified on TUM RGB-D datasets. Experimental

results show that the system works well in real time in dynamic environments without relying on prior semantic knowledge.

The remaining parts are organized as follows. Section 2 details the related work. Section 3 introduces our specific theory for solving the SLAM operation in dynamic scenarios. Section 4 presents the analysis of the experimental results. Section 5 is the final conclusion.

2. Related Work

Currently, SLAM systems that adopt three solutions in dynamic environments can be divided into two categories, namely the SLAM systems based on camera motion [18–25] and the SLAM systems independent of camera motion [26–28]. The former systems need to first calculate the motion of the camera itself, and then judge and distinguish the dynamic target. However, this system fell into the "chicken and egg" problem, so the researchers tried to find the dynamic target before calculating the camera's own motion.

2.1. SLAM Systems Based on Camera's Own Motion in Dynamic Environment

In this kind of system, the main work can be divided into three categories, which are methods based on improved RANSAC [12], algorithms based on foreground/background model segmentation, and algorithms based on semantic segmentation or target detection.

The first representative algorithm obtains the camera's own motion through improving the RANSAC [12]. Lu et al. designed an enhanced RANSAC method based on grid division and key point distribution, which assumes that background points are widely distributed throughout the image and the dynamic points are relatively concentrated [18].

The second representative algorithm is based on front/background models. This algorithm requires initial assumption about the environment. Kushwaha et al. proposed a background subtraction technique based on optical flow [19]. Sun et al. proposed an RGB-D-based dynamic environment SLAM method to remove moving objects [20]. In the learning process, the LMedS algorithm [29] and the reprojection error are used to derive the foreground. A foreground model is constructed through the codebook model [30]. Jaimez et al. proposed a joint VO–SF system to find dynamic regions based on energy minimization [21]. On this basis, Scona et al. proposed a StaticFusion system through an energy residual function [22].

The third representative algorithm is to filter the moving objects based on semantic information. This method using object detection or semantic segmentation can easily detect potential moving objects and directly remove these regional feature points. The currently widely used semantic segmentation networks include the highly accurate Mask R-CNN [31], the lightweight SegNet [32], etc. The widely used target detection networks include the very fast YOLO series [33,34], the SSD [35], etc. A representative example is the DS-SLAM proposed by Yu [9]. This system adds a semantic segmentation thread for the SegNet network [32] on the basis of ORB-SLAM2 [5] and directly removes the key points in moving objects. A DynaSLAM system [10] was proposed by Bescos et al., which adds a Mask R-CNN [31] to segment the target object and detect objects that may move or are moving by combining geometric methods. The latest RDS-SLAM system [23] proposed by Liu et al. is based on ORB-SLAM3 [24]. This system adds a semantic thread and a semantic-based optimization thread. On the basis of the former method, RDMO-SLAM [25] adds two threads to, respectively, calculate optical flow and velocity estimation.

2.2. SLAM Systems Independent of the Camera Motion in Dynamic Environment

Previous studies tried to find the dynamic regions without relying on camera pose, and the general method is to directly use semantic information or geometric constraints to determine whether the regions are dynamic regions or not. The first representative algorithm is a semantic-based approach. Sheng et al. proposed a Dynamic DSO system [26]. This system first uses Mask R-CNN [31] to find the high dynamic targets and checks whether the four neighborhoods of the pixels in the target area are all static points or not. If so, the target area will be determined to be a static area, and this static background area will be directly placed into the DSO system [6]. The second representative algorithm is the method based on geometric constraints. In [27], Dai et al. used the Delaunay triangulation algorithm [36] to obtain the correlation between the points and determine which target region the points belongs to by comparing the distance changes in the triangle edges in two different frames. The edges connecting in two different property areas will be directly removed, and the area with the largest connected area will be regarded as the static area. In [28], Huang et al. proposed a CluterSLAM based on the prior assumption of motion consistency of the feature points of the same rigid body. First, the motion distance matrix of points on the map is calculated. Next, the sparse motion distance matrix is clustered using the Hierarchical Clustering Algorithm (HAC) [37]. Then, the largest rigid cluster block is regarded as the static background region.

3. Design of DGFlow-SLAM

3.1. Framework Overview

As shown in Figure 1, our framework is improved based on the RGB-D SLAM of ORB-SLAM2, which has three threads of Tracking, Local Mapping, and Loop Closing. Our main work is the preprocessing process indicated by the red box. In the preprocessing process, we use grid segmentation and k-means clustering segmentation to remove the key points of dynamic objects. Compared with the ORB-SLAM2 system, our framework greatly reduces the negative impact of error points. In addition, we use an acceleration strategy instead of semantic neural networks, so our framework runs faster than other systems. As shown in Figure 1, the system first extracts the ORB feature points at the t-th and $t-1$-th frames. In the Motion Removal of the preprocessing process, the dynamic feature points are mainly removed by joint k-means clustering segmentation and grid segmentation, and the remaining static feature points are employed to obtain rough poses in the low-cost tracking module. The Mapping thread and Loop Closing thread are used to optimize the rough poses. We can obtain a transformation matrix through the optimized pose. Then, we can use the optimized transformation matrix $\mathbf{T}_{t-1,t-2}$ at the previous moment to speed up the preprocessing process at the current moment. This paper adopts the Low-Cost Tracking of DynaSLAM [10]. Compared with ORB-SLAM2, our framework can operate more accurately in dynamic environments on the premise of meeting real-time requirements.

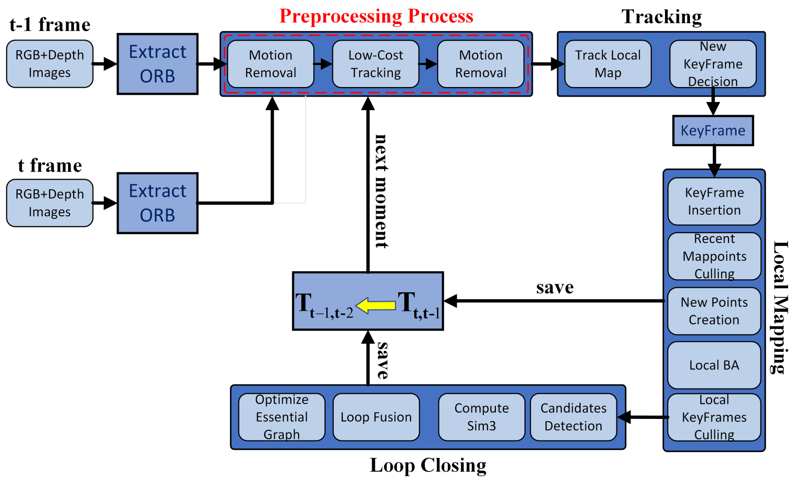


Figure 1. The DGFlow-SLAM frame proposed in this paper. The red dashed box means the preprocessing process part. The yellow arrow indicates that the optimized transformation matrix $\mathbf{T}_{t,t-1}$ at the current moment is saved and used as $\mathbf{T}_{t-1,t-2}$ at the next moment to speed up the preprocessing process.

3.2. Preprocessing Process

The whole preprocessing process is shown in the red dashed box in Figure 2. In this part, we divide the preprocessing process into two cases: non-initial frame and initial frame.

Figure 2 shows the detailed implementation flow of removing the dynamic key point in the non-initial frame state. It can be seen that the process is divided into three modules, namely Segmenting dynamic–static Areas, Moving Object Removal and Low-Cost Tracking. The Segmenting Dynamic–Static Areas module is first executed, and outputs two kinds of dynamic–static segmentation mask images, namely the dynamic–static grid segmentation mask image and the clustering dynamic–static segmentation mask image. Then, the Moving Object Removal module uses the obtained two kinds of dynamic–static mask images to remove the key points in the dynamic area, and the Low-Cost Tracking module calculates the rough camera pose using the static key points. Finally, the Segmenting Dynamic–Static Areas module and the Moving Object Removal module will be executed again to obtain the precise static key points, and the final result will be sent to the Tracking process. This detailed process is: We first compute the rough scene flow of the t-th frame by warping the $t-1$-th frame using the precise transformation matrix $\mathbf{T}_{t-1,t-2} \in \mathbb{R}^4$, where $\mathbf{T}_{t-1,t-2}$ means the transformation matrix between the $t-2$-th frame and the t-1-th frame. Then, we can partition clearly the grid mask image into dynamic or static region parts through statistically operating on the scene flow and use the segmented grid mask image to clearly distinguish the dynamic–static attribute of the clustering blocks. Next, both the dynamic–static grid mask image and the deep mean clustering dynamic–static mask image are used to remove the key points of the dynamic objects. A more accurate transformation matrix $\mathbf{T}_{t,t-1}$ is calculated in the Low-Cost Tracking process using the remaining static environment key points, and a more accurate scene flow is generated. Now, the obtained precise scene flow can be segmented into foreground and background. Finally, the accurate camera positioning can be obtained using the static matching points in the background.

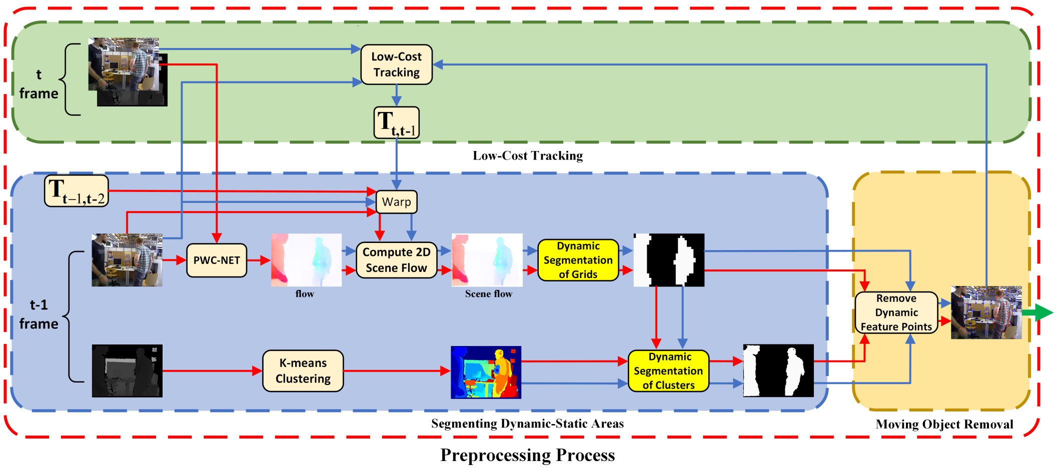


Figure 2. The red dashed line is an overview of the preprocessing process. Different color dashed lines indicate different modules. The blue module is used as the Segmenting Dynamic–Static Areas module to produce a grid segmentation dynamic–static mask image and a k-means clustering segmentation dynamic–static mask image. The yellow module is used as the Moving Object Removal module to remove dynamic keypoints using the two mask images jointly. The green module is used as a Low-Cost Tracking module to calculate rough poses. The blue and yellow modules together form the Motion Removal module in Figure 1, and the green module is the Low-Cost Tracking module in Figure 1. In this figure, we have two iterative processes, the red arrows indicate the first iterative process and the blue arrows indicate the second iterative process. The green arrow indicates the dynamic–static feature points obtained from the preprocessing process are passed to the tracking module. The highlighted parts are grid dynamic–static segmentation and k-means clustering dynamic–static segmentation, respectively.

Unlike the case of non-initial frames, a rough transformation matrix $\mathbf{T}_{t_0,t_1}$, where t_0 means the initial time and $\mathbf{T}_{t_0,t_1}$, means the transform matrix from the t_1-th frame to t_0-th frame is obtained by the Low-Cost Tracking algorithm in the case of the initial frame, and the rest process in the case of the initial frame is the same as the non-initial frame's rest process.

In this Figure 2, the preprocessing process, the Segmenting Dynamic–Static Areas module, and the Moving Object Removal module jointly complete the Motion Removal process in Figure 1. The red arrow path means the first Motion Removal process, and the blue arrow path means the Low-Cost Tracking and the second Motion Removal process. When the red arrow process is completed, the blue arrow process is then executed. Note that the result generated by the first Motion Removal process is used as an initial input of the blue arrow process in the Moving Object Removal module. The green arrows indicate the final output of the preprocessing process.The following sections give the details of our approach.

3.3. Calculation of Scene Flow

Theoretically, the optical flow of an image contains scene flow and camera self-motion flow, which are caused by the dynamic objects and the camera motion, respectively. Now, we can segment the moving objects using the optical flow with an appropriate threshold. However, the self-motion flow of the camera will interfere with the optical flow of moving objects, which makes it difficult to give the appropriate threshold. So, separating the scene flow from the optical flow will be helpful for the segmentation of moving objects.

FlowFusion [38] proposes a 2D scene flow solution method. Optical flow is the coordinate difference of a same point under a camera coordinate system at different moments. The estimation of the optical flow $\delta\mathbf{x}^{of}_{t-1\to t} \in \mathbb{R}^2$ from time t -1 to time t is as follows [38]:

$$\delta\mathbf{x}^{of}_{t-1\to t} = \pi(\mathbf{T}_t \cdot (\mathbf{x}_{t-1} + \delta\mathbf{x}_{t-1\to t})) - \pi(\mathbf{T}_{t-1} \cdot \mathbf{x}_{t-1}) \tag{1}$$

where $\mathbf{x}_{t-1} \in \mathbb{R}^3$ represents the coordinate vector of the 3D point in the world coordinate system at time t-1, $\delta\mathbf{x}_{t-1\to t}$ represents the difference of the 3D coordinate vector of a certain point between time $t-1$ and t, $\pi(\cdot)$ represents the mapping from 3D points to 2D points, and $\mathbf{T}_t \in \mathbb{R}^4$ represents the camera pose at time t. The optical flow consists of the scene flow and the camera self-motion flow. The relationship between 2D optical flow and scene flow can be described as follows [38]:

$$\delta\mathbf{x}^{sf}_{t-1\to t} = \delta\mathbf{x}^{of}_{t-1\to t} - \delta\mathbf{x}^{ef}_{t-1\to t} \tag{2}$$

where $\delta\mathbf{x}^{sf}_{t-1\to t}$ and $\delta\mathbf{x}^{ef}_{t-1\to t}$ represent 2D scene flow and camera self-motion flow, respectively. Camera self-motion flow is an optical flow due to the camera's own motion. The camera transformation matrix is used to warp the original coordinate values to other camera coordinate systems to obtain new coordinates, and the new coordinates are subtracted from the original coordinates to obtain the camera self-motion flow. Now, the camera self-motion flow is calculated as follows [38]:

$$\delta\mathbf{x}^{ef}_{t-1\to t} = W(\mathbf{x}, \boldsymbol{\xi}) - \mathbf{x} \tag{3}$$

$$W(\mathbf{x}, \boldsymbol{\xi}) = \pi(\mathbf{T}(\boldsymbol{\xi})\pi^{-1}(\mathbf{x}, D(\mathbf{x}))) \tag{4}$$

where $W(\cdot)$ is a warp calculation and is represented by Equation (4), which warps the coordinates of a pixel point in one image coordinate system to another image coordinate system to obtain its new coordinates. $\mathbf{x} \in \mathbb{R}^2$ are the pixel coordinates on the image and $D(\mathbf{x})$ represents the depth of point $\mathbf{x}$. $\mathbf{T}(\boldsymbol{\xi}) \in SE(3)$ represents the transformation matrix between two different frames, and $\boldsymbol{\xi} \in se(3)$ represents the Lie algebra transformation which contains rotation and translation.

It is very time-consuming to compute the dense optical flow, so this paper uses the PWC-NET neural network [39] to estimate the optical flow. In order to avoid too much error caused by the depth camera during measurement, the threshold value is set to 7 m. The error caused by the depth camera will be less than 2 cm in the range of 4 m, but it will rise rapidly when the range is more than 4 m and can reach almost 4 cm when the range is 7 m. Excessive errors are harmful to our method, so the threshold is set to 7 m. As can be seen from the scene flow image in Figure 2, the static background pixels are almost close to white because the camera's self-motion flow greatly weakens the static part of the optical flow. In contrast, the foreground pixels are brightly colored because the camera's self-motion flow does not have much effect on the dynamic part.

In the process of calculating the scene flow, we use the transformation matrix $\mathbf{T}_{t-1,t-2}$ from the $t-1$-th frame to t-th frame to replace the roughly calculated transformation matrix between the $t-1$-th frame and t-th frame by the low-cost tracking algorithm. Our approach has two advantages: (1) the iteration speed is accelerated because it reduces one time pose solution process; (2) the information between two sequential frames is effectively correlated.

3.4. Algorithm Design for Dynamic Grid Segmentation

It is very time-consuming to directly execute scene flow statistics for each pixel, and the discrete error points will also affect the dynamic–static segmentation of the scene flow. To solve this problem, a grid containing 20×20 units is used to divide the scene flow image into N grid regions. The reason for setting the grid region as 20×20 is as follows: A large grid will result in too many unnecessary pixels and easily lead to wrongly distinguish the dynamic–static regions. On the contrary, a small grid will consume too much calculation time. In our experiment, we found that the size of 20×20 is most suitable for the grid. It can ensure the accuracy of the algorithm in distinguishing the dynamic–static regions without consuming too much calculation time. Then, a 2-norm sum of a grid can be obtained by calculating the total size of the 2-norm of the scene flow of all pixels within a grid, and finally a statistical computing is executed based on the 2-norm sums of all grids. This method can greatly speed up the operation and eliminate the influence of outliers. It is worth noting that the moving objects will cause a larger depth difference than the static area, so we only calculate the depth difference between the pixels within the 2×2 range of a grid center and their matching points and incorporate this information into the dynamic–static segmentation calculation.

First of all, since we only perform statistical calculations for the scene in the range of 7 m, all grids which only contain the scenes outside the range of 7 m will be removed. The grids containing the scenes both within and outside the range of 7 m will be determined whether they are valid grids or not, according to Equation (5). Let the grid be $G_j = \{p_i \in \mathbb{R}^2 | p_1, p_2, \ldots, p_m\}$, and the valid grid calculation method is as follows:

$$\begin{cases} if \quad N(G_j(D_{p_i} < 7))/N(G_j) \geq 0.8, \quad then \quad G_j \in GC_{valid} \\ if \quad N(G_j(D_{p_i} < 7))/N(G_j) < 0.8, \quad then \quad G_j \in GC_{invalid} \end{cases} \tag{5}$$

where $N(G_j(D_{p_i} < 7))$ represents the number of pixels with a depth of less than 7 m in the grid G_j, and $N(G_j)$ represents the total number of pixels in the grid G_j. Equation (5) means that when the ratio of the number of valid pixels to the total number of pixels is less than 80%, the grid G_j will be regarded as an invalid grid. Considering that some grids are likely the edge grids, and we need to retain those edge grids to contain enough information, we set the threshold to 0.8. Moreover, if the threshold is too small, the grids containing less information will also be retained, which is not conducive to the segmentation of dynamic regions.

In our method, the grid-based process of dynamic–static segmentation is as follows:

Step 1: Remove the invalid grids and keep the valid grids.

Step 2: Calculate the 2-norm sum for each grid by calculating the total size of the 2-norm of the scene flow of all pixels within the grid and find the grid with the smallest 2-norm sum among the valid grids.

Step 3: The dynamic and the static grids are segmented through the segment model, which is built based on a given threshold value and the grid with the smallest 2-norm sum. The detailed implementation process is given below.

Now, we can calculate the 2-norm size of the scene flow for each pixel as follows:

$$d_{i,j} = \sqrt{(d_x^{sf})_{i,j}^2 + (d_y^{sf})_{i,j}^2} \tag{6}$$

where d_x^{sf} and d_y^{sf} are the elements of the scene flow vector, and i and j mean the i-th pixel and the j-th grid, respectively. The scene flow of the j-th grid is the sum of the scene flow of the pixels in the j-th grid, namely $\sum_{i=1}^{m} d_{i,j}$. We select the grid G_{min} with the smallest scene flow in the valid grids GC_{valid} as the static background grid. Finally, We add the depth difference in the process of grid division statistics.

We derive our depth difference equation according to the principle of the literature [13]. Let (v_i, u_i) be the coordinate of the upper left corner of the center 2×2 area of grid G_j, and the 2×2 depth difference at the center of the grid can be calculated as:

$$V_{G_j} = \sum_{m=0}^{1}\sum_{n=0}^{1}\left| |ExtD(\mathbf{T}(\boldsymbol{\xi})\pi^{-1}((v_i+m, u_i+n), D^t_{(v_i+m,u_i+n)}))| - |D^t_{(v'_i+m,u'_i+n)}| \right| \tag{7}$$

where $ExtD(\cdot)$ is to extract the depth value of the map point, and (v_i, u_i) and (v'_i, u'_i) are matched by optical flow [40]:

$$\begin{cases} v'_i = v_i + d_x^{of} \\ u'_i = u_i + d_y^{of} \end{cases} \tag{8}$$

where d_x^{of} and d_y^{of} are the elements of the optical flow vector.We use scene flow S_{G_j} and grid center depth difference together to find the dynamic grid. Then, we can compute the sum of scene flow and the center depth difference for grid $G_j = \{p_i \in \mathbb{R}^2 | p_1, p_2, \ldots, p_m\}$ by the following equation:

$$S_{G_j} = \sum_{l}^{m} d_{i,j} + \gamma V_{G_j} \tag{9}$$

The grid with the smallest scene flow value in the valid grid is used as the smallest static background grid G_{min}. Now, based on the obtained static background grid G_{min}, we can divide all grids into two different parts, namely the dynamic grid part and static grid part by the following equation:

$$\begin{cases} if \quad S_{G_j} \geq \tau_1 \sum_{i=1}^{m} d_{i,min}^{G_{min}} + \gamma V_{G_{min}}, \quad then \quad G_j \in GC_{dynamic} \\ if \quad S_{G_j} < \tau_1 \sum_{i=1}^{m} d_{i,min}^{G_{min}} + \gamma V_{G_{min}}, \quad then \quad G_j \in GC_{static} \end{cases} \tag{10}$$

Among them, τ_1 is 3.3∼3.9, and γ is 10∼20. The reason to set the values of τ_1 and γ is that using the transformation matrix $\mathbf{T}_{t-1,t-2}$ at the previous moment as the initial transformation matrix $\mathbf{T}_{t,t-1}$ to segment the current frame may lead to a high error and affect the gap between G_{min}, the two moving scene flows, which makes it difficult to judge the distinction between dynamic–static regions. In this case, when the threshold is set too small, the static background will be wrongly segmented into dynamic regions. Furthermore, if there is no valid grid, namely $\forall G_j \in GC_{invalid}$, a threshold must be set to distinguish dynamic–static grids:

$$\begin{cases} if \quad S_{G_j} \geq \tau_2, \quad then \quad G_j \in GC_{dynamic} \\ if \quad S_{G_j} < \tau_2, \quad then \quad G_j \in GC_{static} \end{cases} \tag{11}$$

where τ_2 is set to 1500∼2000. Considering a special case that most of the pixels in the picture are moving, to retain some moving points and ensure that there are enough feature points to help SLAM run stably, the threshold is chosen in the range between 1500 and 2000. The reason for setting the threshold to 1500∼2000 is as follows: The scene flow of one pixel point of a vigorously moving object is at least 3, so the value of the scene flows of one grid is at least $3 * (20 * 20) = 1200$. Moreover, the inaccurate transformation matrix will make the error range of the scene flows of one grid be about 0.4∼0.8. Then, the minimum error value of the scene flows of one grid is about $0.4 * (20 * 20) + 1200 = 1360$, and the maximum error value of the scene flows of one grid is about $0.8 * (20 * 20) + 1200 = 1520$. Furthermore, the influence of depth difference needs to be considered, and the depth difference between two consecutive frames is about 4∼6cm. According to the value range of γ is 10∼20, the minimum depth difference threshold is $4 * (2 * 2) * 10 = 160$, and the maximum depth difference threshold is $6 * (2 * 2) * 20 = 480$. Then, we have the threshold range $(1360 + 160 \approx 1500, 1520 + 480 = 2000)$. The method to judge whether a grid is a dynamic or static grid is shown in Algorithm 1.

Algorithm 1 Dynamic–static grid discrimination.

```
Input: scene flow image and depth image
Output: Dynamic–static grid sets $Array_{GC_{dynamic}}$ and $Array_{GC_{static}}$
 1: $Array_{GC_{dynamic}}, Array_{GC_{static}}, Array_{GC_{valid}}, Array_{GC_{invalid}} \leftarrow 0$
 2: $F_{min} = +\infty, threshold = +\infty$
 3: for each $G_j$ do
 4:     $c_1 = N(G_j(D_{p_i} < 7))/N(G_j)$
 5:     if $c_1 \geq 0.8$ then
 6:         $Array_{GC_{valid}} \leftarrow G_j$
 7:         if $F_{min} > \sum_{i=1}^{m} d_{i,j}$ then
 8:             $F_{min} = \sum_{i=1}^{m} d_{i,j}, G_{min} = G_j$
 9:         end if
10:     else
11:         $Array_{GC_{invalid}} \leftarrow G_j$
12:     end if
13: end for
14: if $Array_{GC_{valid}} \neq \emptyset$ then
15:     $threshold = \tau_1 * F_{min} + \gamma V_{G_{min}}$
16: else
17:     $threshold = \tau_2$
18: end if
19: for each $G_j$ do
20:     $S_{G_J} = \sum_{l}^{m} d_{i,j} + \gamma V_{G_j}$
21:     if $S_{G_J} \geq threshold$ then
22:         $Array_{GC_{dynamic}} \leftarrow G_j$
23:     else
24:         $Array_{GC_{static}} \leftarrow G_j$
25:     end if
26: end for
```

3.5. Dynamic Segmentation Algorithm Based on Deep K-Means Clustering

The simple grid segmentation method cannot segment all key points of the dynamic target region, and there are some potential dynamic points that need to be removed together. For example, people may move the chair they are sitting on when they get up. Then, the moving chair is a potential dynamic object [9,10,13]. According to the suggestion proposed in the paper of Joint VO–SF [21], we selected 24 classes to segment the image. The number of k-means clusters is set according to the image size, for example, if the image size is $640 * 480$, then the number of k-means clusters is $[640/10] * [480/10] = 6 * 4 = 24$. This is given empirically by the Joint VO–SF authors, mainly to provide medium-sized clustering blocks.

Next, the segmented grid mask image is used to make a distinction between dynamic–static attributes of the cluster blocks. Let the dynamic overlapping area between a cluster block image and a segmented grid mask image be S, and the entire cluster block is considered to be dynamic if the ratio of S to the cluster block is great than τ_3. The corresponding method can be shown as:

$$\begin{cases} if \quad N(GC_{dynamic} \cap B_i)/N(B_i) \geq \tau_3, \quad then \quad B_i \in BC_{dynamic} \\ if \quad N(GC_{dynamic} \cap B_i)/N(B_i) < \tau_3, \quad then \quad B_i \in BC_{static} \end{cases} \tag{12}$$

where $N(Z)$ represents the number of elements in the set Z and B_i represents the pixel point set of the i-th mean cluster. We found that similar regions are in the same motion, so we set τ_3 to be 0.3∼0.5. Finally, we eliminate the potential key points in the dynamic clustering regions. The above implementation process can be shown as Algorithm 2. We did not set up 24 clustering blocks for all experiments but only for the experiments with $640 * 480$ images. Only 24 clustering blocks are available for $640 * 480$ image sizes. If the image size is $640 * 480$, then the number of k-means clusters is $[640/10] * [480/10] = 6 * 4 = 24$. So, the 24 clusters are not constant numbers in Algorithm 2. When working with larger images or smaller images, one has to go for modifying the number of clustering blocks. Bigger images have more clustering blocks and vice versa.

Algorithm 2 Dynamic–static K–means clusters discrimination.

```
Input: Grid mask image and depth image
Output: Dynamic–static cluster block sets $Array_{BC_{dynamic}}$ and $Array_{BC_{static}} \leftarrow 0$
    K-means clustering was carried out on the depth image, and the clustering was 24 blocks
2:  for each $B_j$ do
        $c = N(GC_{dynamic} \cap B_i)/N(B_i)$
4:      if $c \geq \tau_3$ then
            $Array_{BC_{dynamic}} \leftarrow B_i$
6:      else
            $Array_{BC_{static}} \leftarrow B_i$
8:      end if
    end for
```

3.6. Acceleration Strategy

As shown in Figure 3, we use a two-layer pyramid for k-means clustering acceleration. First, we use a 4×4 Gaussian kernel to reduce the image to 1/4 of the original image. Then, we perform k-means clustering on the upper image, and the iterative times are no more than 10. The upper image clustering results in 24 clustering blocks, and we map the centroids of these clustering blocks back into the original image. As shown in Figure 3, a brown centroid of one cluster on the upper image is mapped to a blue pixel block with the size of 4×4 in the original image. Within the range of blue pixel blocks, the coordinate closest to the average depth is the center of the cluster block. We perform the following calculation for each pixel point (v_i, u_i) in the pixel block:

$$CenterError = \|D_{(v_i,u_i)} - \frac{\sum_{j=0}^{4} D_{(v_j,u_j)}}{4}\|_2 \tag{13}$$

where $D_{(v_i,u_i)}$ denotes the depth of pixel point (v_i, u_i). The pixel point (v_i, u_i) with the smallest CenterError is used as the mean clustering center of the original image, and then the original pixel points are assigned to each clustering center.

The running time of our method is mainly consumed in the process of k-means clustering iteration. The time complexity of k-means clustering is $O(NKt)$. Where N is the number of data objects, K is the number of clustering blocks, and t is the number of iteration times. In layer 1, N, K, and t are $160 * 120$, 24, and 10, respectively. In layer 0, N, K, and t are $640 * 480$, 24, and 1, respectively.

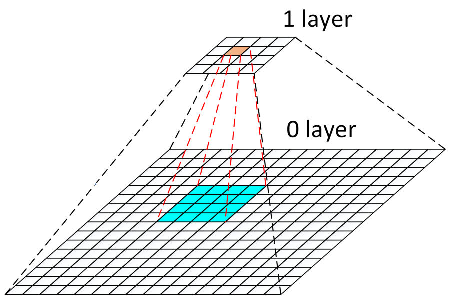


Figure 3. A two-level image pyramid to help speed up computing. The blue pixel block is the mapping result of the brown pixel block in layer 0.

3.7. Joint Removal of Dynamic Feature Points

Furthermore, to ensure the small objects can be successfully eliminated, DGFlow-SLAM is proposed by combing the segmenting grid mask image and the segmenting mean clustering mask image. DGFlow-SLAM means that both Algorithms 1 and 2 are first used to obtain the corresponding segmenting mask images, respectively, and then the two different segmenting mask images are superimposed to eliminate the dynamic key points.

4. Experiment Results

4.1. Experimental Setup

The DGFlow-SLAM proposed in this paper is verified on the public dataset TUM, and the following experiments are carried out on the s/static, w/static w/half, w/rpy, and w/xyz datasets. The s/static dataset contains only slight movement caused by the head or partial limbs of a human body. The w/static dataset contains moderate-intensity movement caused by slowly getting up or the walking of human beings, and we can find the camera moves slightly at the same time by observing this dataset. The rest of the datasets, such as the w/half, w/rpy, and w/xyz datasets, contain violent movement caused by human beings, and we can find these cameras move violently at the same time by observing the corresponding datasets. Moreover, these cameras move along a semicircle on the w/half dataset, rotate along the main axis on the w/rpy dataset, and move along the three axes of xyz on the w/xyz dataset, respectively. Finally, the following experiments are completed using Intel Core i7 and 8G memory.

We employ absolute trajectory error (ATE) and relative pose error (RPE) to conduct quantitative evaluation [41]. Among them, ATE represents the global consistency of errors in the metric system and the overall performance of the system, and RPE represents translational and rotational drift errors and reflects the drift of the system. Finally, the system evaluation is divided into two parts: accuracy evaluation and time evaluation.We used the following enhancement values [10] to measure the effect of our system on ORB-SLAM2:

$$\eta = \frac{o-r}{o} \times 100\% \tag{14}$$

where o represents the size of the error value of ORB-SLAM2 [5], and r represents the size of the error value generated by our method. This enhancement value indicates how much we improved the ORB-SLAM2 improvements. The enhancement values are shown in the last two columns of Tables 1–3. On the other hand, we use ablation experiments to validate the performance of our grid segmentation, k-means clustering segmentation, and the combination of both methods. Finally, we analyze the operational effects on a public dataset, as well as on a real dataset.

Table 1. Result of absolute trajectory error (ATE).

Sequence	ORB-SLAM2		DS-SLAM		DynaSLAM		DGFlow-SLAM		Enhancements	
	RMSE	**S.D.**	**RMSE**	**S.D.**	**RMSE**	**S.D.**	**RMSE**	**S.D.**	**RMSE**	**S.D.**
w/half	0.5380	0.2374	0.0303	0.0159	**0.0296**	0.0157	0.0489	0.0284	90.91%	88.04%
w/rpy	1.0061	0.5496	0.4442	0.2650	**0.0354**	0.019	0.2789	0.1856	71.28%	66.28%
w/static	0.2036	0.0994	0.0081	0.0033	**0.0068**	0.0032	0.0086	0.0046	95.58%	95.37%
w/xyz	0.6445	0.2676	0.0247	0.0161	**0.0164**	0.0086	0.0280	0.0187	95.66%	93.01%
s/static	0.0080	0.0040	0.0065	0.0033	0.0108	0.0056	**0.0059**	**0.0029**	26.25%	27.5%

Table 2. Results of translation relative pose error (RPE translation).

Sequence	ORB-SLAM2		DS-SLAM		DynaSLAM		DGFlow-SLAM		Enhancements	
	RMSE	**S.D.**	**RMSE**	**S.D.**	**RMSE**	**S.D.**	**RMSE**	**S.D.**	**RMSE**	**S.D.**
w/half	0.3728	0.2873	0.0297	0.0152	**0.0284**	0.0149	0.0447	0.0284	88.01%	94.81%
w/rpy	0.4058	0.2964	0.1503	0.1168	**0.0448**	0.0262	0.1970	0.1492	51.54%	49.66%
w/static	0.1858	0.1614	0.0102	0.0038	**0.0089**	0.0044	0.0114	0.0062	93.86%	96.16%
w/xyz	0.4214	0.2923	0.0333	0.0229	**0.0217**	0.0119	0.0398	0.0266	90.56%	90.90%
s/static	0.0084	0.0041	0.0078	0.0038	0.0126	0.0067	**0.0076**	**0.0037**	9.52%	9.76%

Table 3. Results of rotation relative pose error (RPE rotation).

Sequence	ORB-SLAM2		DS-SLAM		DynaSLAM		DGFlow-SLAM		Enhancements	
	RMSE	**S.D.**	**RMSE**	**S.D.**	**RMSE**	**S.D.**	**RMSE**	**S.D.**	**RMSE**	**S.D.**
w/half	7.6591	5.8202	0.8142	0.4101	0.7842	0.4012	1.0696	0.6817	86.03%	88.29%
w/rpy	7.8952	5.7114	3.0042	2.3065	0.9894	0.5701	3.7283	2.7804	52.18%	51.32%
w/static	3.4029	2.9116	0.2690	0.1215	0.2612	0.1259	0.2800	0.1307	91.54%	95.11%
w/xyz	8.0446	5.5260	0.8266	0.2626	0.6284	0.3848	**0.6187**	0.3754	92.31%	93.21%
s/static	0.2729	0.1179	0.2735	0.1215	0.3416	0.1642	**0.2698**	**0.1150**	**1.14%**	**2.46%**

4.2. Accuracy and Runtime Evaluation

4.2.1. Accuracy Evaluation

As shown in Tables 1–3, we quantitatively analyzed the ATE and RPE of the four systems. RMSE represents the deviation between the observed value and the true value and reflects the robustness of the system. The standard deviation S.D. represents the degree of deviation and reflects the stability of the system. Tables 1–3 show the ATE, the RPE translation, and the RPE rotation, respectively. Compared with the classical ORB-SLAM2 [5], the DGFlow-SLAM decreases the RMSE of ATE, RPE translation, and RPE rotation by 95.66%, 90.56%, and 91.54%, respectively, when they detect the fast motion sequence on the dataset w/xyz, and decreases the RMSE of ATE, RPE translation, and RPE rotation by 26.25%, 9.52%, and 1.14%, respectively, when they detect the static sequence on the dataset s/static. Compared with the rest of the SLAM systems, the DGFlow-SLAM achieves the best results on the dataset s/static. This result shows that the DGFlow-SLAM is more suitable to detect the slow motion objects than the rest of the SLAM systems. In addition to this, compared with the methods with prior semantic knowledge, such as the DS-SLAM [9] and the DynaSLAM [10], the DGFlow-SLAM performances slightly worse on the datasets w/static and w/xyz, but the errors of the three SLAM systems are very close to each other.

We compare the algorithm performance of ORB-SLAM2 [5], DS-SLAM [9], DynaSLAM [10], and DGFlow-SLAM through their estimated camera trajectories. Among them, ORB-SLAM2 is a classic SLAM system without prior semantic knowledge, and DS-SLAM [9] and DynaSLAM [10] are the classic semantic SLAM systems. The data of DS-SLAM [9] and DynaSLAM [10] are derived from RDS-SLAM [23]. In Figure 4, black lines, blue lines, and red lines represent true trajectories, estimated trajectories, and difference between estimated and true values, respectively.

Figure 4 displays the comparison results. We know that ORB-SLAM2 [5] performs poorly on all data sets. Compared with ORB-SLAM2 [5], DS-SLAM [9] shows a great enhancement on the rest of the datasets, except the w/rpy dataset. Compared with ORB-SLAM2 [5] and DS-SLAM [9], DynaSLAM [10] shows a better performance on all the datasets. However, DynaSLAM discards much pose data along the camera's estimated trajectories when it operates on the w/rpy and w/static datasets. So, the DynaSLAM [10] is not good for the loop detection and dense mapping of camera pose. The estimated trajectories by the DGFlow-SLAM are generally consistent with the true trajectories, and most points are close to the true trajectories on the w/half, w/static, and w/xyz datasets. However, the performance of the DGFlow-SLAM is much worse on the w/rpy dataset. Furthermore, we can see that for the most estimated camera poses, DGFlow-SLAM has fewer errors than DS-SLAM [9] and DynaSLAM [10], and for a few estimated camera poses, DGFlow-SLAM has many more errors than DS-SLAM [9] and DynaSLAM [10]. Compared with DynaSLAM [10], DGFlow-SLAM preserves all the camera poses. Compared with ORB-SLAM2 [5], DGFlow-SLAM greatly enhances the estimated camera poses.

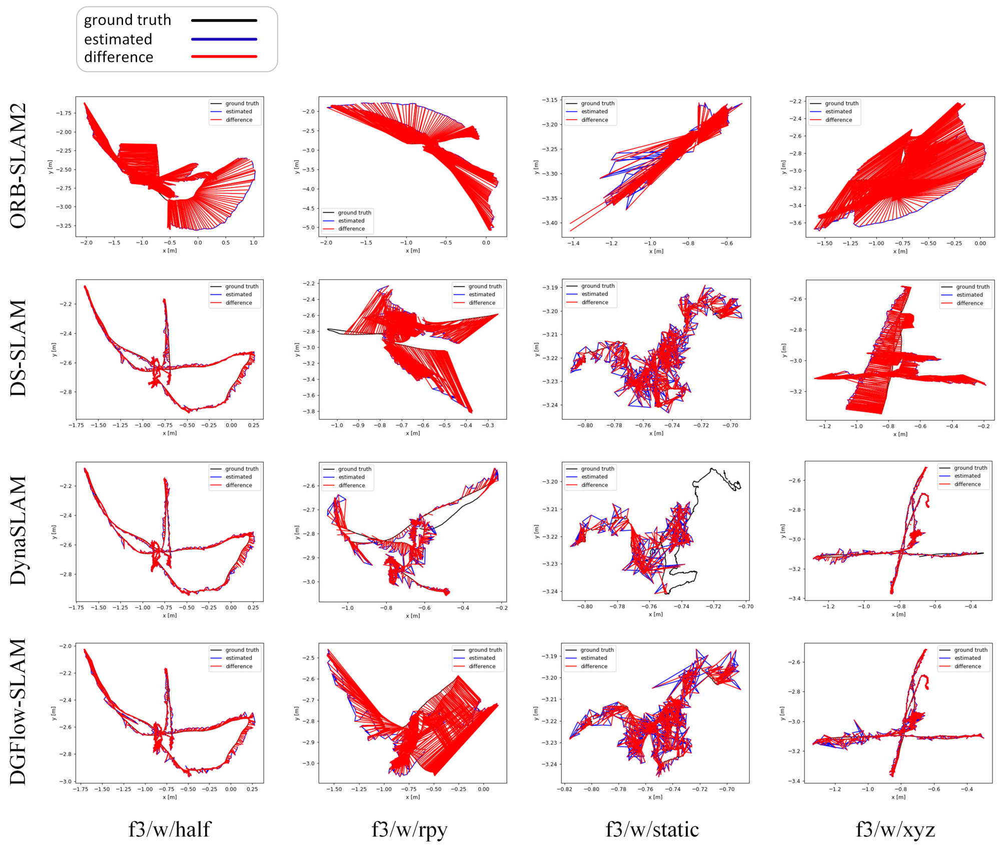


Figure 4. This is the camera trajectory image. Black lines are the true trajectory and blue lines are our estimated camera trajectory. Except for w/rpy, on the other three datasets, our estimated trajectory is basically around the real trajectory.

It is worth noting that to improve the accuracy, some static parts of a dynamic object are preserved, which is not conducive to dense map construction. On the other hand, the optical flow method is sensitive to light changes, and the current framework cannot be applied to outdoor areas.

4.2.2. Runtime Evaluation

The real-time performance of a SLAM system is an important indicator. DynaSLAM [10] has the best detection accuracy on almost all dynamic datasets compared with the other SLAM systems. However, DynaSLAM [10] takes 500 ms to complete the detection per frame and cannot meet the real-time requirements of SLAM. DGFlow-SLAM using the real-time PWC-NET network [39] module for optical flow estimation is much faster than DynaSLAM [10] using the Mask-RCNN network [31] module for optical flow estimation. Table 4 shows that the total time of the three modules of the DGFlow-SLAM is about 51 ms, less than the working time of DS-SLAM (65 ms) [9] and far less than t DynaSLAM (500 ms) [10]. The reason for the fast speed is that we use an acceleration strategy. The total time for 10 iterations of k-means clustering runs on 1/4 image is 4 ms, and one k-means clustering calculation at 640 ∗ 480 is 3 ms. Therefore, the total time for k-means clustering is 7 ms, which is more than 50% of the total time of the preprocessing process. Compared with the consumption time of 37.57 ms for the Seg-Net network [32] used in DS-SLAM [9] and 195 ms for the Mask-RCNN network [31] used in DynaSLAM [10], our algorithm is much faster. So, DGFlow-SLAM meets the real-time requirements of SLAM.

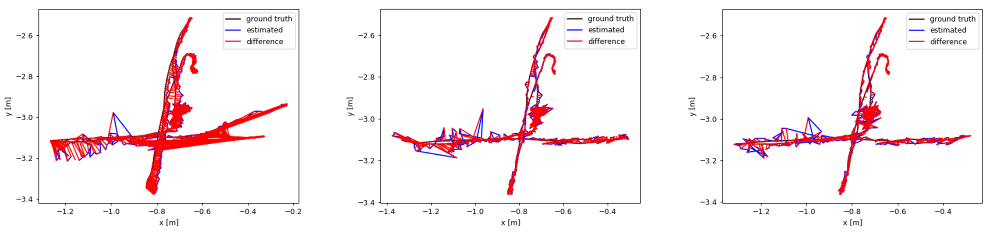
Table 4. The time cost of each module of our system.

Sequence	ORB Extraction and Matching	Removal of Outliers	PWC-NET	Total Time
Times (ms)	21	18	12	51

In summary, our scheme outperforms the RGB-D mode in the original ORB-SLAM2 [5] and achieves similar accuracy as DS-SLAM [9] and DynaSLAM [10]. Compared with DynaSLAM [10], on the dataset w/xyz, DGFlow-SLAM performances slightly worse, but its detection speed is greatly enhanced. In Table 1, the enhancement value of DGFlow-sSLAM is less than 1.8% of the enhancement value of 97.46% of DynaSLAM [10]; however, the detection speed of DGFlow-SLAM is 10 times faster than DynaSLAM [10]. Unlike DS-SLAM [9] and DynaSLAM [10], we do not use semantic segmentation networks, and our proposed method is much faster than semantic segmentation. Our method performs optimally in the case of very slight movements.

4.3. Ablation Experiment

We use grid segmentation mask image, k-means clustering segmentation mask image, and their joint segmentation mask image to remove dynamic objects and compare their effects on the dataset w/xyz, respectively. As shown in Figure 5, the grid segmentation mask image is less effective in removing dynamic objects, mainly because it does not remove many potential static parts. The joint image is the best in removing dynamic objects, which is because the grid segmented mask image has some accuracy improvement by removing the small object moving parts. The data in Figure 5 and Table 5 are the median values selected by running DGFlow-SLAM three times.



(**a**) grid mask (**b**) cluster mask (**c**) joint mask

Figure 5. Camera trajectory image of the removal effect of three mask images on dynamic objects.

Table 5 shows the data of the RMSE of the ATE for the ablation experiment. The joint mask image works the best, and the RMSE value is only 0.0234.

Table 5. Absolute trajectory error obtained by the DGFlow-SLAM system after removing dynamic points using three mask images separately

Mask Image	Grid Mask	Kmean Mask	Joint Mask
RMSE	0.1259	0.0275	0.0234

4.4. Run on a Public Dataset

Figure 6 shows our performance on four TUM datasets, where red and blue points represent dynamic key points and static key points, respectively. From the comparison between scene flow image and optical flow image, we can see that the optical flow is greatly weakened, and the picture in the first column of the second row shows the most obvious change. The whole image is very blurry and the boundary of objects cannot be seen clearly, and the reason is that the camera's self-motion flow covers the whole image. So, we can see all objects clearly when we remove the camera's self-motion flow. From the picture in the first column of the last row, we can see that most of two human bodies can be found. However, there are still some areas that are not considered to be dynamic masks, such as the shin of the left person and the leg of the right person when they are static. The method regarding the unmoved human body region as static region will help to solve the camera pose. As shown in the last row of Figure 6, most of the feature points of the dynamic region can be found.

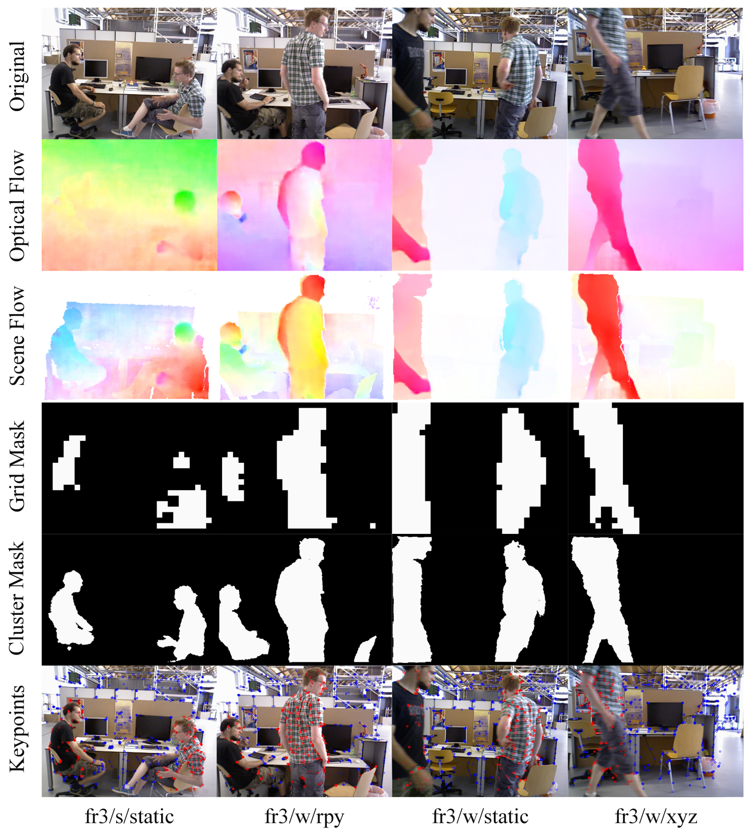


Figure 6. Comparison of the four datasets. The first row represents the input image, and the second row is the optical flow result produced by PWC-NET [39]. Our computed scene flow graph is shown in the third row. The fourth and the fifth rows are the dynamic–static grid mask images and the dynamic–static mean clustering mask images, respectively. The last row shows the distinction between the dynamic points and the static key points of ORB, and the red key points and the blue key points represent dynamic points and background points, respectively.

Figure 7 shows the detection of small object motion. In Figure 7a, only one head appears in the picture, and the head is well identified and segmented in Figure 7b using the

Algorithm 1. In Figure 7c, the white area indicates that the object is completely segmented using Algorithm 2. Figure 7d shows the corresponding dynamic–static key points obtained by the DGFlow-SLAM. Another example is the ball rolling experiment. Figure 7e is the original image. Figure 7f shows the result of Algorithm 1, and the white area represents the rolling ball. Figure 7g shows the result of Algorithm 2. It can be seen that because the moving objects are too small, the k-means clustering method cannot segment the dynamic targets. Figure 7h shows the dynamic and static key points obtained by DGFlow-SLAM. The above experiments show that Algorithm 1 or Algorithm 2 may fail to segment the small objects in different environments, while the DGFlow-SLAM system can segment the small objects in different environments successfully.

Figure 8 shows that we can detect the slow moving objects by combining the grid and the adaptive threshold because most of the slight movements exceed more than 1 pixel distance, and the adaptive threshold is far less than 1. We define the movement change between 0.8∼2 of scene flow of one pixel in a frame as slight motion. The movement change between Figure 8a,b is very slight, and we can see that the foot of the right human body in the figure has very slight movement. We can see that there is a distinct red area in the lower right of Figure 8c, and this distinct red area is clearly found in Figure 8d,e. We can see that the corresponding dynamic key points are marked with red in Figure 8f, and we will remove the relevant dynamic key points in the system. The above experiments demonstrate that our method is highly sensitive to slight movements.

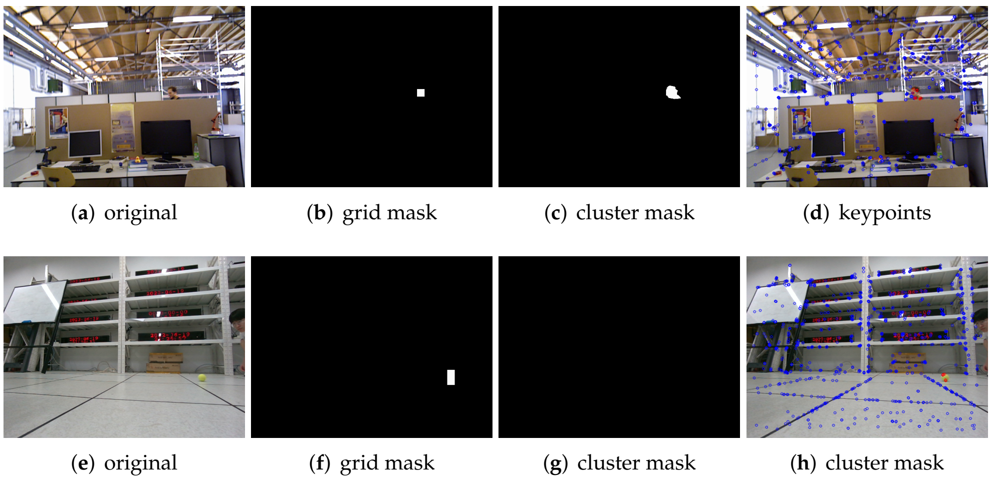


Figure 7. (**a**–**d**) show the cases of small object motion on the public data sets, as does Figure 5. (**e**,**f**) show the cases of small object motion taken with camera. In (**a**), only a head of the human body appears in the image. (**b**) shows the results using Algorithm 1. (**c**) shows the results using Algorithm 1. (**d**) shows the dynamical key points and the static key points using the algorithm DGFlow-SLAM. (**e**) shows an image of a small ball rolling, and (**f**) is the result of Algorithm 1. (**g**) is the result of Algorithm 2. (**h**) shows the dynamical key points and static key points using the algorithm DGFlow-SLAM.

In Figure 8a, most of the area on the right side of the person is in a static state, and only the feet have very slight movements, which is difficult to see from the figure. However, we are able to accurately segment the slightly moving parts based on the scene flow. We also need to segment and remove the ground part for two main reasons: the first reason is that the reflection of the human body affects the ground optical flow estimation; the second reason is that the optical flow estimation is prone to errors on the ground where the overall luminosity is almost the same. This illustrates the difficulty of applying our method to outdoor areas with strong photometric variations, as it is very sensitive to changes in optical flow. In addition, it is worth noting that a stationary chair will move when the left person stands up, which is a typical potential moving object that can be detected by our method.

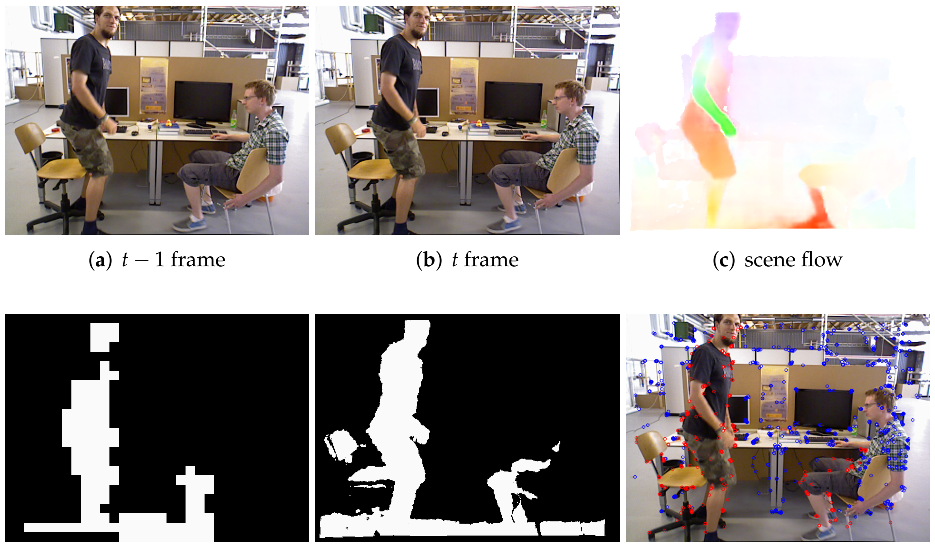


Figure 8. This figure shows an example of slight motion segmentation. (**a**,**b**) are frames at time $t-1$ and time t, respectively, in which the human body is in slow motion. (**c**) is a scene flow graph.

4.5. Run in Real Environment

To verify that our system can work properly in the real world, we conducted extensive experiments for DGFlow-SLAM in a laboratory environment. In these experiments, we used a RealSense D455 camera to capture a large number of 640 $\times$ 480-size RGB images and depth images by hand-held shooting. Figure 9 shows the results of our experiments. The first row is the case of a single person walking. From left to right are the captured RGB image, the calculated 2D scene flow, the grid segmentation mask, the k-means clustering mask, and the dynamic–static key point classification results, respectively. Almost all of the dynamic key points of the human body are correctly recognized and marked in red. The second row is a scene of two people moving towards each other, and almost all of the dynamic key points of the human body are also clearly marked.

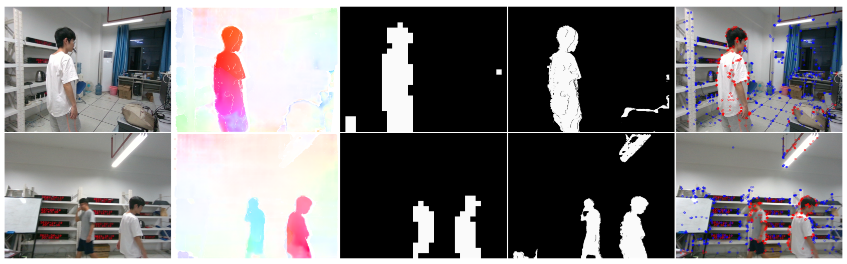


Figure 9. The experimental cases in our real environment. The first row is a scene of a single person exercising, and the second line shows two people walking in opposite directions. The white in the two mask images represents the dynamic part, and the black represents the background area.

5. Conclusions

In order to improve the adaptability of classical ORB-SLAM2 [5] in complex and unknown environments, we proposed a new framework DGFlow-SLAM to reduce the impact of dynamic feature points. In this framework, grid segmentation and k-means clustering segmentation are combined to remove dynamic regions, which greatly improves the segmentation performance of dynamic–static feature points. In addition, in order to speed up the iteration and strengthen the connection between two sequential frames, the framework uses the previous solution as the current initial solution to roughly segment the dynamic–static regions. Experiments show that DGFlow-SLAM can clearly detect violent motion and slight motion without prior semantic knowledge. Moreover, DGFlow-SLAM has the best accuracy in detecting slight motion than the other SLAM systems because the threshold of DGFlow-SLAM is determined by the minimum grid scene flow and the corresponding grid depth difference. Finally, compared with the classical semantic SLAM systems, namely DS-SLAM [9] and DynaSLAM [10], DGFlow-SLAM is the fastest, and compared with the classic ORB-SLAM2 [5], DGFlow-SLAM improved the accuracy by 95.58% , 95.66%, and 26.25% on the moderate motion dataset, violent motion dataset, and slight motion dataset, respectively.

However, there is still some work to be conducted for DGFlow-SLAM. For example, our method cannot completely realize dense map construction without dynamic objects. In the future, we will use data association to remove temporarily stationary moving objects from the map. In addition, DGFlow-SLAM can only be applied to indoor scenes at present, and the conversion from indoor scenes to outdoor scenes requires more work. For the semantic SLAMs, their development direction should be to accelerate the speed of using semantic segmentation networks and increase the accuracy of segmentation edges. In addition, SLAMs with no prior semantic knowledge and SLAMs with prior knowledge should be organically combined to achieve the goal of recognizing both unknown objects and known dynamic objects with high accuracy.

Author Contributions: F.L.: experiment preparation, data processing, and publication writing; L.D.: experiment preparation and publication writing; J.L.: technology support, data acquisition, and publication review. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China under grants 61966014 and 61962023.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access* **2020**, *8*, 58443–58469. [CrossRef]
2. Qin, T.; Chen, T.; Chen, Y.; Su, Q. Avp-slam: Semantic visual mapping and localization for autonomous vehicles in the parking lot. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 5939–5945.
3. Höll, M.; Lepetit, V. Monocular LSD-SLAM Integration Within AR System. *arXiv* **2017**, arXiv:1702.02514.
4. Quan, L.; Yin, L.; Xu, C.; Gao, F. Distributed swarm trajectory optimization for formation flight in dense environments. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 4979–4985.
5. 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]
6. Engel, J.; Koltun, V.; Cremers, D. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 611–625. [CrossRef]
7. Sun, Y.; Liu, M.; Meng, M.Q.H. Improving RGB-D SLAM in dynamic environments: A motion removal approach. *Robot. Auton. Syst.* **2017**, *89*, 110–122. [CrossRef]
8. Li, S.; Lee, D. RGB-D SLAM in dynamic environments using static point weighting. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2263–2270. [CrossRef]
9. 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.
10. Bescos, B.; Fácil, J.M.; Civera, J.; Neira, J. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4076–4083. [CrossRef]
11. Zhang, Z.; Deriche, R.; Faugeras, O.; Luong, Q.T. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artif. Intell.* **1995**, *78*, 87–119. [CrossRef]
12. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]

13. Zhong, F.; Wang, S.; Zhang, Z.; Wang, Y. Detect-SLAM: Making object detection and SLAM mutually beneficial. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018; pp. 1001–1010.
14. Li, A.; Wang, J.; Xu, M.; Chen, Z. DP-SLAM: A visual SLAM with moving probability towards dynamic environments. *Inf. Sci.* **2021**, *556*, 128–142. [CrossRef]
15. Ji, T.; Wang, C.; Xie, L. Towards real-time semantic rgb-d slam in dynamic environments. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 11175–11181.
16. Fan, Y.; Zhang, Q.; Tang, Y.; Liu, S.; Han, H. Blitz-SLAM: A semantic SLAM in dynamic environments. *Pattern Recognit.* **2022**, *121*, 108225. [CrossRef]
17. Liu, Y.; Miura, J. KMOP-vSLAM: Dynamic visual SLAM for RGB-D cameras using K-means and OpenPose. In Proceedings of the 2021 IEEE/SICE International Symposium on System Integration (SII), Virtual, 11–14 January 2021; pp. 415–420.
18. Lu, X.; Wang, H.; Tang, S.; Huang, H.; Li, C. DM-SLAM: Monocular SLAM in dynamic environments. *Appl. Sci.* **2020**, *10*, 4252. [CrossRef]
19. Kushwaha, A.; Khare, A.; Prakash, O.; Khare, M. Dense optical flow based background subtraction technique for object segmentation in moving camera environment. *IET Image Process.* **2020**, *14*, 3393–3404 [CrossRef]
20. Sun, Y.; Liu, M.; Meng, M.Q.H. Motion removal for reliable RGB-D SLAM in dynamic environments. *Robot. Auton. Syst.* **2018**, *108*, 115–128. [CrossRef]
21. Jaimez, M.; Kerl, C.; Gonzalez-Jimenez, J.; Cremers, D. Fast odometry and scene flow from RGB-D cameras based on geometric clustering. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3992–3999.
22. Scona, R.; Jaimez, M.; Petillot, Y.R.; Fallon, M.; Cremers, D. Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 3849–3856.
23. Liu, Y.; Miura, J. RDS-SLAM: Real-time dynamic SLAM using semantic segmentation methods. *IEEE Access* **2021**, *9*, 23772–23785. [CrossRef]
24. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.; Tardós, J.D. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
25. Liu, Y.; Miura, J. RDMO-SLAM: Real-time visual SLAM for dynamic environments using semantic label prediction with optical flow. *IEEE Access* **2021**, *9*, 106981–106997. [CrossRef]
26. Sheng, C.; Pan, S.; Gao, W.; Tan, Y.; Zhao, T. Dynamic-DSO: Direct sparse odometry using objects semantic information for dynamic environments. *Appl. Sci.* **2020**, *10*, 1467. [CrossRef]
27. Dai, W.; Zhang, Y.; Li, P.; Fang, Z.; Scherer, S. Rgb-d slam in dynamic environments using point correlations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 373–389. [CrossRef]
28. Huang, J.; Yang, S.; Zhao, Z.; Lai, Y.K.; Hu, S.M. Clusterslam: A slam backend for simultaneous rigid body clustering and motion estimation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 5875–5884.
29. Rousseeuw, P.J. Least median of squares regression. *J. Am. Stat. Assoc.* **1984**, *79*, 871–880. [CrossRef]
30. Kohonen, T. Learning vector quantization. In *Self-Organizing Maps*; Springer: Berlin/Heidelberg, Germany, 1995; pp. 175–189.
31. 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.
32. 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]
33. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
34. Fang, W.; Wang, L.; Ren, P. Tinier-YOLO: A real-time object detection method for constrained environments. *IEEE Access* **2019**, *8*, 1935–1944. [CrossRef]
35. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *Lecture Notes in Computer Science: Proceedings of the European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
36. Barber, C.B.; Dobkin, D.P.; Huhdanpaa, H. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw. TOMS* **1996**, *22*, 469–483. [CrossRef]
37. Sokal, R.R. A statistical method for evaluating systematic relationships. *Univ. Kansas, Sci. Bull.* **1958**, *38*, 1409–1438.
38. Zhang, T.; Zhang, H.; Li, Y.; Nakamura, Y.; Zhang, L. Flowfusion: Dynamic dense rgb-d slam based on optical flow. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 7322–7328.
39. Sun, D.; Yang, X.; Liu, M.Y.; Kautz, J. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8934–8943.

40. Hempel, T.; Al-Hamadi, A. Pixel-wise motion segmentation for SLAM in dynamic environments. *IEEE Access* **2020**, *8*, 164521–164528. [CrossRef]
41. 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.