applied sciences

MDPI

Article

Visual SLAM Mapping Based on YOLOv5 in Dynamic Scenes

Xinguang Zhang *, Ruidong Zhang and Xiankun Wang

School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China
* Correspondence: zxg416@126.com

Abstract: When building a map of a dynamic environment, simultaneous localization and mapping systems have problems such as poor robustness and inaccurate pose estimation. This paper proposes a new mapping method based on the ORB-SLAM2 algorithm combined with the YOLOv5 network. First, the YOLOv5 network of the tracing thread is used to detect dynamic objects of each frame, and to get keyframes with detection of dynamic information. Second, the dynamic objects of each image frame are detected using the YOLOv5 network, and the detected dynamic points are rejected. Finally, the global map is constructed using the keyframes after eliminating the highly dynamic objects. The test results using the TUM dataset show that when the map is constructed in a dynamic environment, compared with the ORB-SLAM2 algorithm, the absolute trajectory error of our algorithm is reduced by 97.8%, and the relative positional error is reduced by 59.7%. The average time consumed to track each image frame is improved by 94.7% compared to DynaSLAM. In terms of algorithmic real-time performance, this paper's algorithm is significantly better than the comparable dynamic SLAM map-building algorithm DynaSLAM.

Keywords: dynamic SLAM; YOLOv5 network; deep learning

check for updates

Citation: Zhang, X.; Zhang, R.; Wang, X. Visual SLAM Mapping Based on YOLOv5 in Dynamic Scenes. *Appl. Sci.* **2022**, *12*, 11548. https://doi.org/10.3390/app122211548

Academic Editor: Lidia Jackowska-Strumillo

Received: 29 September 2022
Accepted: 11 November 2022
Published: 14 November 2022

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

Copyright: © 2022 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

With the rapid development of robotics and computer application technology, Simultaneous Localization and Mapping System (SLAM) [1,2] technology has been widely used in agriculture, unmanned driving, aerospace, and other fields. As one of the most advanced technologies in the field of mobile robotics, SLAM utilizes a variety of sensors on the robot to conduct localization, navigation, and map construction in the environment. A robot must build a map of its environment to realize autonomous positioning and navigation in an unfamiliar environment. Therefore, for the robot, the construction of an environmental map is the foundation of all follow-up work, and an accurate map can enable the robot to conduct accurate autonomous positioning. With in-depth research, SLAM technology has also been greatly developed, and many SLAM mapping technologies have emerged, such as VSLAM [3], ORB-SLAM [4], ORB-SLAM2 [5], DynaSLAM [6], ORB-SLAM3 [7], RGBD SLAM [8–10], DS-SLAM [11], and so on. However, most of the mapping construction in the SLAM system is based on a static environment. When drawing in a static environment, the ORB-SLAM2 system is one of the classic drawing frames of the SLAM system. The ORB-SLAM2 system has the advantages of a simple structure and fast running speed. Therefore, this research is based on ORB-SLAM2. Most of the existing SLAM systems operate in a static environment, and most of them suffer from low accuracy of map building, large trajectory errors, and poor robustness of the system when building maps in a dynamic environment. As most of our environments are dynamic (i.e., they are static only in specific circumstances), map construction and positional trajectories in dynamic environments are even more important.

In the process of researching dynamic SLAM map building [12,13], many scholars have conducted in-depth research on SLAM systems in a dynamic environment, put forward many improvement methods, put forward new methods in the process of SLAM research,

and achieved certain results. However, in practical applications, problems such as poor real-time performance of the improved algorithms, low accuracy of map building, and large trajectory errors still exist and have not been fundamentally solved. Linhui Xiao et al. [14] constructed a Single Shot MultiBox Detector (SSD) object detector based on a convolutional neural network and prior knowledge when eliminating dynamic objects in a dynamic environment, and used this SSD object detector to detect moving objects in a dynamic environment. At the same time, a missing detection compensation algorithm was added to optimize object detection. Finally, map construction and pose estimation were carried out in the SLAM system. Linyan Cui et al. [15,16] used a semantic optical flow SLAM method to detect and reject dynamic objects in the environment, and at the same time used a segmentation Network (SegNet) to mask the semantic information, and then filter the object features, detect the dynamic object, and segment the dynamic object. The dynamic points were then eliminated by judging the feature points that have mobility without a priori dynamic markers. The obtained static features were then used to build the map. Yuxiang Sun et al. [17] proposed a method to delete RGB-D data when removing feature points in a dynamic environment, and applied this improved method before employing SLAM. In the process of drawing construction, the method proposed by the author was used to eliminate the dynamic data, to achieve the effect of removing the dynamic features. Feng Li et al. [18] used an enhanced semantic segmentation network to detect and segment dynamic features of dynamic SLAM environments as a way to achieve dynamic feature rejection. Miao Sheng et al. [19] adopted a static and dynamic image segmentation method to improve the accuracy of image construction to reduce the interference of dynamic objects in image construction. This method uses the semantic module to segment the image and then uses the geometric module to detect and eliminate the dynamic objects. Shiqiang Yang et al. [20] used a combined geometric and semantic approach for the detection of dynamic object features, which first uses a semantic segmentation network for the segmentation of dynamic objects, and then uses an approach based on the geometric constraint method to reject the segmented dynamic objects. Wenxin Wu et al. [21] accelerated and generated the semantic information necessary for the SLAM system by using a low-latency backbone while proposing a new geometrically constrained method to filter the detection of dynamic points in image frames, where dynamic features can be distinguished by random sampling consistency (RANSAC) using depth differences. Wanfang Xie et al. [22] proposed a motion detection and segmentation method to improve localization accuracy, while using a mask repair method to ensure the integrity of segmented objects, and using the Lucas–Kanade optical flow (LK optical flow) [23] method for dynamic point rejection in the rejection of dynamic objects. XUDONG LONG et al. [24] proposed a Pyramid Scene Parsing Network (PSPNet) detection thread based on the pyramid optical flow method combined with semantic information. In dynamic feature detection, PSPNet is used as the thread of semantic information detection to carry out semantic annotation on dynamic objects. An error-compensation matrix is designed to improve the real-time performance of the system, reducing the time consumed. Ozgur Hasturk et al. [25] proposed a dynamic dense construction method using 3D reconstruction errors to identify static and dynamic objects by studying the construction of dynamic environments. To improve the robustness and efficiency of dynamic SLAM mapping, Li Yan et al. [26] proposed a method of combining geometric information with semantic information. Firstly, a residual model combined with a semantic module was used to perform semantic segmentation for each frame of an image. At the same time, a feature point classification method was designed to detect potential dynamic objects. Weichen Dai et al. [27] proposed a map-building method that includes static maps and dynamic maps, in which dynamic maps consist of motion trajectories and points, and the correlation between maps is used to distinguish static points from dynamic points as a way to eliminate the influence of dynamic points in the map-building process. ImadE Bouazzaou et al. [28] eliminated the effect of errors on map-building localization in dynamic environments by improving the mode of sensor-to-image acquisition and by selecting the specifications of the sensor for a particular environment. Chenyang Zhang

et al. [29] proposed a PLD–SLAM combined with deep learning (for semantic information segmentation) and a K—Means clustering algorithm considering depth information (to detect the underlying dynamic features for the dynamic SLAM map building method) in order to solve the problem of incomplete or over-segmentation of images caused by deep learning while using point and line features to estimate the trajectory. Guihai Li et al. [30] combined the tracking algorithm and target detection, using the limit geometric constraint method to detect static points, while at the same time introducing depth constraints in the system to improve the robustness of the system. However, these methods have certain drawbacks when performing in dynamic environments, and there are various unavoidable disadvantages. When using geometric methods for dynamic point rejection, problems such as incomplete detection of dynamic points and non-detection of low dynamic points can easily occur, which in turn affects the accuracy of map building and positional accuracy. Additionally, when using deep learning networks [31,32] for map building, an unavoidable problem is the poor real-time performance of the system, and the processing time per image frame increases significantly.

When the above methods build a map in a dynamic environment, they have the disadvantages of inaccurate detection of dynamic feature points and low accuracy of pose estimation, as well as the large computational power of semantic networks leading to the slow running of algorithms and the phenomenon of lost frames. These problems will affect the mapping and pose estimation accuracy of the SLAM system, leading to the failure of mapping and serious deviations of positioning and navigation. To solve the interference of dynamic objects in the environment to the SLAM system and to improve the accuracy of SLAM mapping and pose estimation, a new SLAM mapping method in a dynamic environment is proposed in this paper. In this paper, a dynamic feature detection module is added to the ORB-SLAM2 algorithm. In the feature point recognition and rejection module, the LK optical flow method is used to detect and reject dynamic feature points. The YOLOv5 (You Only Look Once) target detection network can detect most objects with dynamic features. Afterward, the detected images are detected and rejected twice using the optical flow method. By detecting and rejecting the dynamic features twice, the proposed algorithm can reject most of the dynamic features well and ensure the accuracy of map building and pose estimation. Moreover, the YOLOv5 network has a small scale and small amount of calculation, which ensures the running speed of the algorithm, effectively solving the problem of lost frames in the algorithm. Compared with SLAM algorithms of the same type, the running speed of the algorithm in this paper is much faster than that of DynaSLAM and DS-SLAM algorithms in the literature.

The main contributions of this paper are as follows: 1. Applying the deep learning target detection network YOLOv5 to the SLAM mapping system, which enables the SLAM system to perform fast target detection and segmentation; 2. A secondary detection of dynamic targets based on the deep learning network YOLOv5 combined with the LK optical flow method is proposed, which can effectively remove dynamic features in the dynamic feature region of points; 3. Based on the ORB-SLAM2 system, a new SLAM mapping system is successfully constructed. The structure of this paper is as follows: In the Section 1, the principle of the ORB-SLAM2 algorithm and YOLOv5 network structure, as well as the function of each module, are introduced in detail. In the Section 2, an improvement to the ORB-SLAM2 algorithm is proposed, and the principle of eliminating dynamic feature points is described in detail. The Section 3 uses the TUM dataset to verify the proposed algorithm, and provides the experimental results based on the TUM dataset. Finally, the Section 4 summarizes and considers future developments that may arise from the work in this paper.

2. Method Introduction

The dynamic environment-based YOLOv5 visual SLAM algorithm proposed in this paper uses three main modules: 1. the ORB-SLAM2 module; 2. the target-detection network YOLOv5 module; and 3. the dynamic feature point rejection module. The work of

building the algorithm for this paper can be broken into three main parts. For the first part, the underlying algorithms ORB-SLAM2 and the YOLOv5 network are publicly available. For the second part, the datasets required for the experiments are available on the TUM website. The third part is code improvement and model training. The research in this paper focuses on the third part. Firstly, the model is trained on the acquired dataset using YOLOv5. Secondly, the process of image processing using the LK optical flow method will be detailed in the paper. Finally, the modules are fused in the ORB-SLAM2 system for dynamic environment building. The specific computational process is as follows:

2.1. ORB-SLAM2 Algorithm Structure

ORB-SLAM2 is mainly divided into three threads: 1. the tracking thread; 2. the map-building thread; and 3. the loop closing detection thread. Each of these three threads has its function when building the map in conjunction with the other corresponding modules. In the process of map construction, the ORB-SLAM2 system first acquires image information through the sensors, then carries out ORB feature detection and feature extraction on the collected images, and finally carries out camera positioning through each image frame to determine whether to join keyframes. In the build thread, new keyframes are processed, while old ones are removed, and the map is globally optimized by Bundle Adjustment (BA). After the local mapping is established, loopback detection will optimize the global mapping. Figure 1 illustrates the principle of the ORB-SLAM2 algorithm.

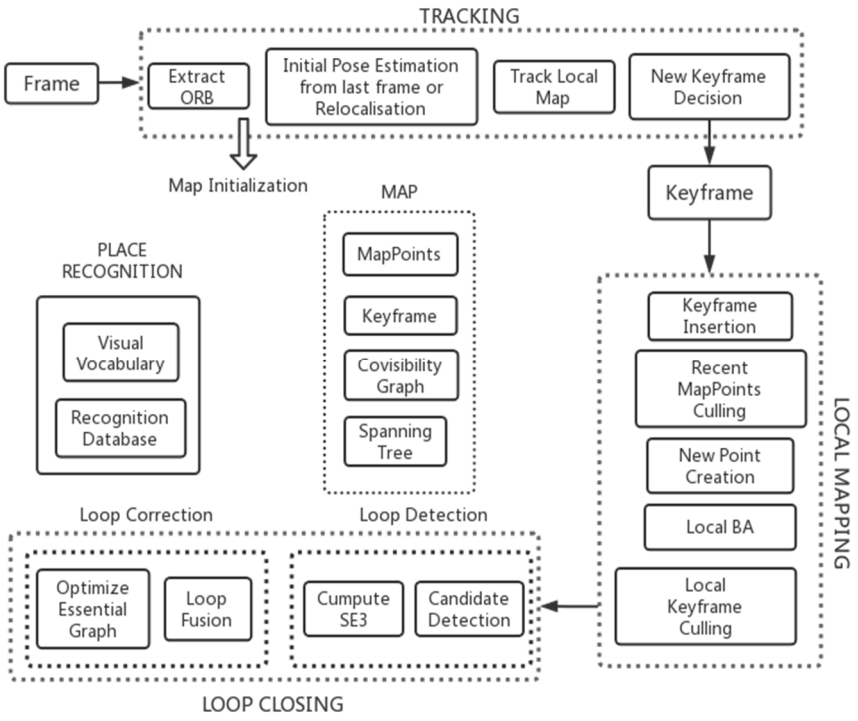


Figure 1. Block diagram of the ORB-SLAM2 algorithm.

As shown in Figure 1, when the camera collects the image, it inputs the image to the first thread of the ORB-SLAM2 system: the TRACKING thread, which consists of four modules. When inputting an image, the Extract ORB module first extracts ORB features from the image to initialize the map. After the map initialization is successful, the Initial Pose Estimation module initializes the camera pose according to the initialization map. After initialization, the system starts to track and construct the map, and the Keyframe Discrimination module starts to screen the new keyframe as the input of the next thread. The LOCAL MAPPING thread consists of five modules, and the key frame selected by the Tracking thread is used as the input of the LOCAL MAPPING thread. After receiving the keyframes, the Keyframe Insertion module inserts the newly entered keyframes into

the local map. The Recent Map Points Culling module inserts the old map points in the keyframes into the local map, while the New Point Creation module inserts the map points that do not exist in the local map into the local map as new map points. After the new map points are inserted, the Local BA module optimizes the local map locally, and then the Local Keyframes Culling module removes the current keyframe. The LOOP CLOSING phase is mainly divided into two modules: Loop Detection and Loop Correction. In pose estimation and optimization, the Candidates Detection module compares the current keyframe with the keyframe of the Recognition Database in PLACE RECOGNITION, calculates the Similarity Transformation (Sim3), and selects the keyframe with the largest result as the input of the Loop Correction module. The Loop Correction module uses the newly input keyframes for global map optimization and pose estimation, to reduce the error and make the mapping results more accurate.

The specific functions of the three threads are as follows:

- Tracking is mainly done by extracting ORB features from the image, performing pose estimation based on the previous frame, or performing initialization of the poses by global repositioning, then tracking the already-reconstructed local maps, optimizing the poses, and then determining new keyframes according to a particular set of rules.
- Map building is mainly used for local map construction. It includes insertion of keyframes; validation and filtering of recently generated map points; generation of new map points using local bundle set adjustment (Local BA); and finally filtering the inserted keyframes again to remove redundant keyframes.
- Closed-loop detection is divided into two main processes: closed-loop detection and closed-loop correction. First, the Bag of Words (BOW) is used to detect the closed-loop detection, and then a similar transformation is calculated by the Sim3 algorithm. Closed-loop correction mainly consists of closed-loop fusion and graph optimization of the Essential Graph.

2.2. YOLOv5 Network Target Detection

The YOLO network model is an algorithm that can be used for target detection, and the algorithm consists of a separate convolutional network model, as shown in Figure 2. The implementation process is roughly as follows: the size of the input image is modified and fixed, then fed into the convolutional neural network; the prediction results after the operation are shown on the far right of Figure 2, which clearly shows the detection content of the target object. Compared with the most basic Region Convolutional Neural Networks (R-CNN) algorithm, the YOLO algorithm is a more complete and advanced framework for image recognition and detection, and is faster and more efficient.

As shown in Figure 2, the YOLO series works on target recognition of images. Firstly, feature extraction is performed on the input image. The first convolutional layer performs a series of convolutional processes on the image to obtain three different feature maps of sizes 16×16, 64×64, and 256×256, respectively. Since the three feature images are not of the same size, they cannot fully reflect the target information when performing feature extraction, so it is necessary to fuse the features of the three differently sized feature images as a way to obtain stronger target features for better results. When the three feature images are input into the second convolutional layer, the convolutional network in the second layer performs a convolution operation on the feature images, making the three feature images into three feature images of the same size. The three feature maps, now of the same size, are fused in Classification and Location Regression to obtain the final detection image with the target detection result. One of the roles of the convolutional network is to divide the image into pixel levels and then perform feature extraction on the feature regions at the pixel level.

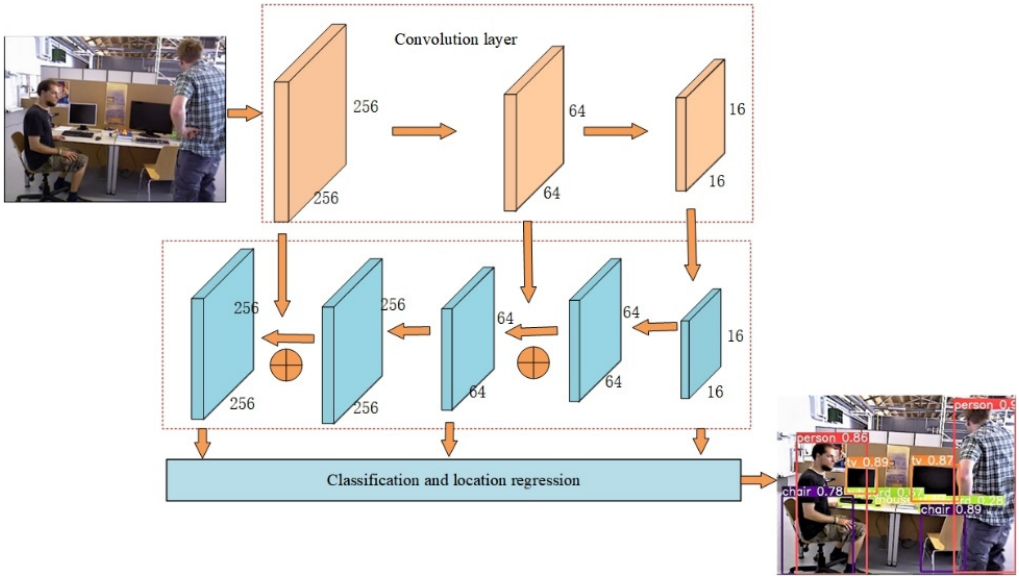


Figure 2. Structure principle of the YOLO algorithm.

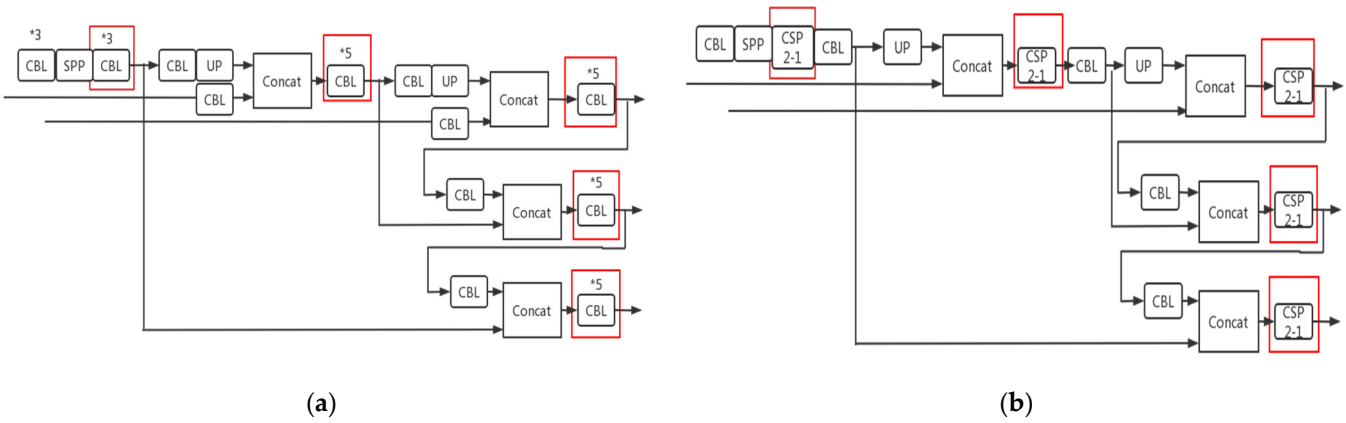
At present, the YOLOv5 [33,34] is mainly divided into four network models, which are YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. Among them, the YOLOv5s network model is the base model with shallow depth, and the narrowest width of the feature map compared to the other three models. The other three network models are upgraded versions of YOLOv5s, i.e., deepened and widened on its network model.

YOLOv5 consists of three main modes when performing data enhancement: 1. performing scaling; 2. performing color space adjustment; and 3. using mosaic enhancement. When using mosaic for image enhancement, the selected images need to be stitched in a manner with random scaling, random cropping, and random distribution, which makes the detection better when detecting small targets.

2.3. Adaptive Anchor Frame Calculation

The initial Anchor can be set in the dataset, and when the network is trained, the training network outputs the prediction frame based on the Anchor, and the loss can be calculated by comparing it with the ground truth frame; after the loss is obtained, the network is updated with feedback, and the network parameters are updated by iterating through each layer of the network. The initial anchor points in YOLOv3 [35] and YOLOv4 [36] can be set using a specific K-means algorithm setup procedure. However, the Anchor of the YOLOv5 network differs from the above two versions in that the Anchor is written directly into the overall code of the network in the YOLOv5 network, and the optimal anchor frame values of the YOLOv5 network are calculated adaptively each time the dataset is used for training.

In the Neck layer of YOLOv4, all the structures required for its operation need to be combined with the convolutional network. In contrast, the CSP2_X structure designed by the Content Security Policy network (CSPnet) is used in the Neck layer of YOLOv5, which makes the YOLOv5 network more capable of fusing features, as shown in Figure 3b.



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

Figure 3. (**a**) Neck structure of YOLOv4; (**b**) Neck structure of YOLOv5.

As shown in Figure 3, compared with the Neck of YOLOv4, the Neck of YOLOv5 uses a different convolution network on some modules. The Neck layer in YOLOv5 consists of five convolutional network modules combined. The five convolutional network modules are the Conv and BN and Leaky ReLU (CBL) module, the Spatial Pyramid Pooling (SPP) module, the CSP2-1 module, the Upsampling (UP) module, and the Concat module. The functions and roles of each module are as follows:

1. CBL module: The CBL module is mainly composed of the most basic convolution modules, i.e., Conv, Batch normalization (BN), and Leaky ReLU. Its main function is the convolution of the input image and extraction of image features, and it is often used in the convolution layer of image convolution processing in target detection.
2. SPP module: The SPP module mainly solves the problem of frame loss during image clipping and scaling, and it enables the convolutional neural network to extract features many times, which improves the speed of generating candidate boxes. In YOLOv5, the feature image is mainly pooled.
3. CSP2-1 module: the gradient value of backpropagation between layers can be increased by increasing the residual structure, which avoids the disappearance of the gradient caused by the deepening of the network, so that more fine-grained features can be extracted and so that we do not have to worry about network degradation. It is mainly used in the backbone of the YOLOv5 network.
4. UP module: the main function of the UP module is to pool the feature image from the CBL convolution layer so that the size of the input feature image is increased and the size of the feature image is consistent.
5. Concat module: the input feature images are fused to better highlight the features of target detection.

To make the calculation result more accurate, YOLOv5's Central Input/Output Unit (*CIOU*) not only adds the distance between the target object and the anchor frame to the calculation result, but also adds the coverage and penalty class to it, so it will not fail to converge as YOLOv4's *CIOU* did in the training process. The *CIOU* formula is as follows:

$$CIOU = IOU - \frac{\rho^2(b, b^{gt})}{c^2} - \alpha v \tag{1}$$

where $\rho^2(b, b^{gt})$ represents the Euclidean length of the prediction frame and the center of the real frame shape, respectively; c is the diagonal length of the prediction frame and the rear frame together forming the range; and α and v are given by the following equations:

$$\alpha = \frac{v}{1 - IOU + v} \tag{2}$$

$$\nu = \frac{4}{\pi^2}\left(\arctan\frac{\omega^{gt}}{h^{gt}} - \arctan\frac{\omega}{h}\right)^2 \tag{3}$$

This also results in the following corresponding formula:

$$LOSS_{CIOU} = 1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha\nu \tag{4}$$

In the data training of the YOLOv5 network, the classic datasets are used to train the model and obtain the training weights. In building the SLAM, the key frame image processed by the tracking thread is used for the detection of dynamic targets. (The detection results are shown in Figure 4, in which it can be seen that the network can accurately identify and frame objects in the image, although indoor detection is better.) Then, the recognized dynamic objects are rejected, and then the static points are used for loopback map building to construct the map with the dynamic environment rejected.



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

Figure 4. (**a**) YOLOv5 outdoor target recognition results; (**b**) YOLOv5 indoor target recognition results.

3. Dynamic Feature Point Rejection

3.1. Principle of Dynamic Feature Point Rejection

SLAM systems can be divided into two general categories, which are used for removing dynamic feature points when building maps in dynamic scenes. The first dynamic feature detection method uses geometric constraints and other traditional methods to detect whether dynamic points are detected, using the position change relationship between two adjacent frames. The disadvantages of this approach are inaccurate dynamic feature detection; unavoidable problems, such as failing to detect or incorrectly detecting objects with low dynamics; and large positional estimation errors. The second dynamic feature detection method that is now added to the existing SLAM system uses machine learning and deep learning target detection methods, such as deep neural networks, convolutional neural networks, etc. When using the second method for detecting dynamic targets, the network is first trained to recognize dynamic targets, which enables fast and accurate identification of the desired dynamic targets.

When the system builds a map, it first uses the YOLOv5 network to perform dynamic object detection on keyframes, labels the detected dynamic objects using dynamic object boxes, and then inputs each frame detected by the network as having dynamic objects into the Tracking thread to reject the dynamic points using the Dynamic Feature Rejection method. Figure 5 shows the overall algorithm framework of this paper, and the gray boxes indicate the three threads of ORB-SLAM2. In this paper, the dashed box is a new thread added to the dynamic object detection algorithm.

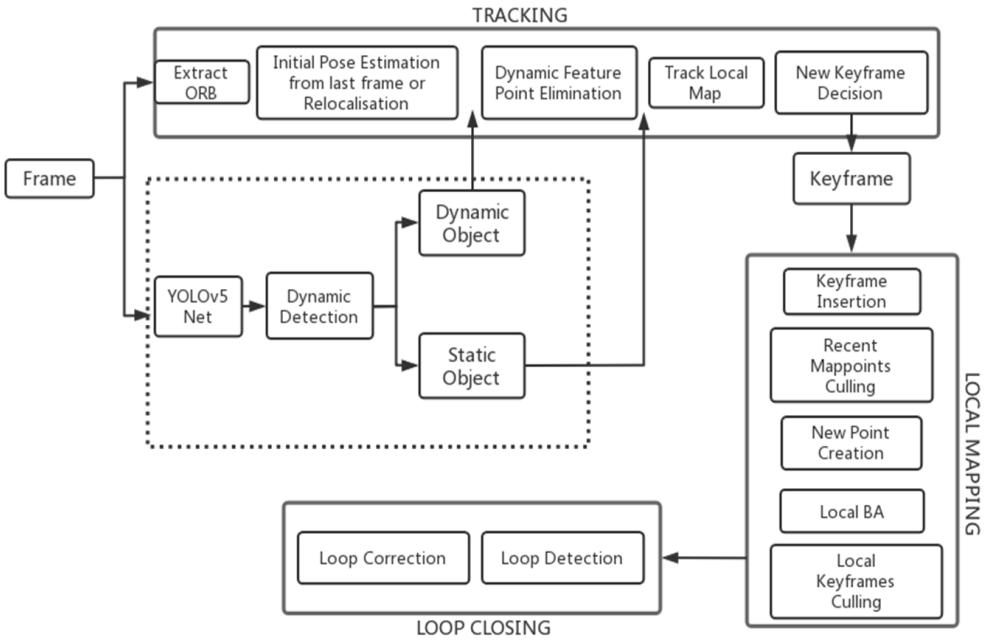


Figure 5. Dynamic object rejection algorithm framework.

3.2. Dynamic Feature Point Detection Based on Optical Flow Method

The LK optical flow method is a two-frame differential optical flow estimation algorithm that computes a sparse optical flow field, based on the following features: 1. constant luminance: the luminance or color of adjacent frames is constant; 2. temporal continuity: continuous "small motion"; and 3. spatial consistency: the motion between adjacent pixels is consistent. The gray value of a point (x, y) in the image at time t is $I(x, y, t)$, and at time $(t + \Delta t)$ the point moves to a new position, $(x + \Delta x, y + \Delta y)$, and the gray value becomes $I(x + \Delta x, y + \Delta y, t + \Delta t)$. According to the luminance constancy assumption, $\partial I(x, y, t)/\partial t = 0$, the grayscale value of the point remains constant before and after the motion; therefore:

$$I(x, y, t) = I(x + \Delta t, y + \Delta y, t + \Delta t) \tag{5}$$

According to the time-continuous assumption that the image motion is tiny, using the Taylor expansion for Equation (5), and ignoring the second-order infinitesimal, we get:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{6}$$

Let $u = \Delta x/\Delta t, v = \Delta y/\Delta t$ be the velocity of the optical flow in the x and y directions, respectively. Then, Equation (6) can be rewritten as:

$$I_x u + I_y v + I_t = 0 \tag{7}$$

in which $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, and $I_t = \frac{\partial I}{\partial t}$ are the partial derivatives of the grayscale values concerning x, y, and t, respectively. Equation (8) is the optical flow constraint equation, which includes two unknown variables u, v. According to the assumption of spatial consistency, a small space w of $m \times m$ can be established, and the optical flow $U = (u, v)^T$ remains constant within the space w. Then, a system of overdetermined equations can be obtained, and the least squares method can be applied to the system of equations to estimate u and v. In the range of space w, the error function of the optical flow equation can be expressed as: $E = \sum_{x,y \in w} (I_x u + I_y u + I_t)^2$. The derivative of the function E is obtained for both u and v, and when the derivative value is 0, the estimated $E = (u, v)^T$ of u and v is obtained, which is the optical flow vector at that point.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum\limits_w I_x^2 & \sum\limits_w I_x I_y \\ \sum\limits_w I_x I_y & \sum\limits_w I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum\limits_w I_x I_t \\ \sum\limits_w I_y I_t \end{bmatrix} \tag{8}$$

When $\begin{bmatrix} \sum\limits_w I_x^2 & \sum\limits_w I_x I_y \\ \sum\limits_w I_x I_y & \sum\limits_w I_y^2 \end{bmatrix}$ is invertible, find $U = (u, v)^T$, at the location in the image where the two directional edges intersect along the line, where the point has two larger eigenvectors that can ensure matrix integrability.

The three assumptions of the LK optical flow algorithm are not easily satisfied, especially when the target speed is fast, and the optical flow vector cannot be calculated using the traditional LK optical flow algorithm. Therefore, in this paper, combining the LK optical flow method, the pyramid optical flow method is used to calculate the optical flow for different pyramid layers, which improves the accuracy of the optical flow calculation. The basic idea of the pyramid optical flow method is as follows: First, calculate the optical flow field at the top layer of the pyramid, and then at each step pass the optical flow information to the next lowest layer, until finally to the original image is reached at the bottom; then, set the bottom layer as level 0, and upgrade upwards through the layers in turn. The level 0 image is down-sampled to get the level 0 pyramid image, and then continues down-sampling to the set number of layers. At this point, the motion of the moving target between two adjacent frames is small enough to meet the assumption bar of the LK optical flow method, and the optical flow value of the layer can be calculated.

As shown in Figure 6, the calculation principle of the pyramid optical flow method is as follows: firstly, the image is divided into different pixel levels such as w^n, w^k, w^1, w^0, etc. Then calculate w^k,w^1,w^0 from the minimum pixel level in turn, and finally extract and reject the dynamic features of the image. The pyramid optical flow algorithm starts from the topmost level and iterates the optical flow calculation through the mapping relationship between each pyramid level to obtain the optical flow of the original image at the bottom level. Suppose the initial value of the top optical flow is $w^n = (u^n, v^n)^T = (0, 0)^T$. After that, we calculate the value of the top optical flow, and the obtained calculation result is expressed as $dw^n = (du^n, dv^n)^T$. Then, the initial value w^n is added to obtain the initial value $w^{n-1} = (u^{n-1}, v^{n-1})^T = w^n + dw^n$ for the next level of calculation. Next, substitute this value to a higher resolution layer for optical flow calculation, repeat the process until the highest resolution of the original image is reached, i.e.,:

$$w^k = w^{k+1} + dw^{k+1} = (u^{k+1} + du^{k+1}, v^{k+1} + dv^{k+1})^T \tag{9}$$

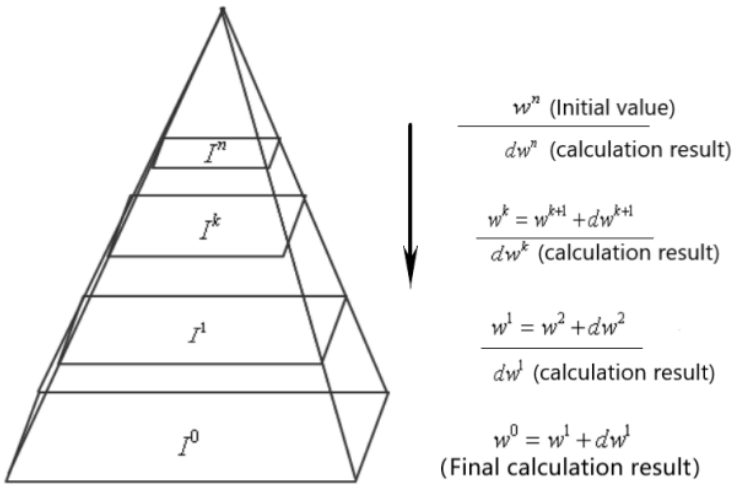


Figure 6. Flow chart of the pyramid optical flow method.

The algorithm in this paper uses pyramidal optical flow to track the feature points in the image when performing dynamic feature point detection, so that the correspondence between the feature points can be obtained. When using the feature point method for feature point detection, it is necessary to calculate the descriptors and perform the matching process of feature points, whereas the optical flow method avoids the time required to calculate and match the descriptors, and therefore has better real-time performance. By tracking the optical flow information between each frame in a dynamic environment, the relationship between matching feature points in neighboring frames is obtained, and thus the position relationship between each frame is recovered. The basis matrix F describes this relative transformation relationship, and its solution can be computed using eight pairs of matching points. Given a pair of matched points whose normalized coordinates are $p_1 = [u_1, v_1, 1]^T$ and $p_2 = [u_2, v_2, 1]^T$, they satisfy between:

$$s_1 p_1 = KP, s_2 p_2 = K(RP + t) \tag{10}$$

where: K—internal reference matrix; R—rotation matrix; and t—translation matrix. According to the pair of polar geometric constraints, we have:

$$(u_2, v_2, 1) \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & 1 \end{bmatrix} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = 0 \tag{11}$$

where F is the matrix in the middle, and Equation (11) can be further written as:

$$p_2^T F p_1 = [u_2, v_2, 1] F [u_1, v_1, 1]^T = 0 \tag{12}$$

Using the basis matrix F, the key point p_1 in the previous frame can be projected into the current frame to obtain the polar line l_2 in the current frame. The distance from the key point to the polar line obtained by optical flow tracking is calculated to determine whether it is a dynamic point or not.

$$l_2 = [X, Y, Z]^T = F p_1 = F [u_1, v_1, 1]^T \tag{13}$$

where X, Y, Z are the normal vectors of the polar lines and F is the basis matrix. The distance D from the matching feature p_2 of the optical flow tracking to the corresponding polar line l_2 can be calculated by Equation (14).

$$D = \frac{\left| p_2^T F p_1 \right|}{\sqrt{\|X\|^2 + \|Y\|^2}} > \sigma \tag{14}$$

where σ is the threshold value. If Equation (14) is satisfied, then p_1 and p_2 are dynamic feature points, and these dynamic feature points are then eliminated from the build map.

4. Method Validation

In this paper, the accuracy of this algorithm in a dynamic environment is tested with three datasets: fre3_walking_xyz, fre3_walking_halfsphere, and fre3_walking_static. The trajectory map results obtained by this algorithm are compared with those obtained by the ORB-SLAM2 algorithm and DynaSLAM algorithm, and the processing time for each frame is compared with that of DynaSLAM. The advantages and disadvantages of this algorithm in terms of accuracy and real-time performance are summarized and analyzed.

In this paper, we use the Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) evaluation metrics to evaluate the mapping accuracy of SLAM systems. Usually, when evaluating map construction accuracy, it is more intuitive to compare the results by observing the map construction, but such observation results can only describe in general terms whether a certain algorithm is relatively accurate with respect to map construc-

tion. Therefore, in order to calculate the accuracy of map construction more accurately, researchers generally use errors to estimate the accuracy of map construction. In this paper, the accuracy of map construction is expressed by calculating the magnitude of Root Mean Squared Error (*RMSE*) with the following formula:

$$RMSE(E_{1:n}, \Delta) = [\frac{1}{m}\sum_{i=1}^{m} \|trans(E_i)\|^2]^{1/2} \quad (15)$$

where $trans(E_i)$ denotes the error at moment i, $E_{1:n}$ denotes the camera pose from 1 to n, and Δ denotes the fixed time interval.

The calculation method of performance improvement in this paper is shown in Equation (16):

$$\eta_{RMSE} = \frac{\eta_{os2} - \eta_{our}}{\eta_{os2}} \times 100\% \quad (16)$$

where η_{os2} represents the *RMSE* of the ORB-SLAM2 algorithm and η_{our} represents the *RMSE* of the OurSLAM algorithm.

$$\eta_{SD} = \frac{\eta_{cs} - \eta_{co}}{\eta_{cs}} \times 100\% \quad (17)$$

where η_{cs} represents the *SD* of the ORB-SLAM2 algorithm and η_{co} represents the *SD* of the OurSLAM algorithm.

In the experiments for this paper, the values of ATE and RPE are used to reflect the results of the map construction accuracy of the algorithm in this paper. The values of RPE and ATE are used in the calculation of the evaluation parameters *RMSE* and Standard Deviation (*SD*). Since *RMSE* is more sensitive to outliers in the data and can be observed more intuitively by deviation, the robustness of the system is described by using *RMSE* in this paper. In response to the degree of dispersion of the camera trajectory, this paper uses the standard deviation for judgment, which can describe the stability of the system more clearly, as shown in Tables 1 and 2.

Table 1. Absolute trajectory error comparison results (ATE).

Datasets	ORB-SLAM2		OurSLAM		Performance Improvement/%	
	RMSE	***SD***	***RMSE***	***SD***	η_{RMSE}	η_{SD}
fre3_Warking_xyz	0.6395	0.3162	0.0139	0.0071	97.8	97.8
fre3_walking_halfsphere	0.6685	0.3385	0.0158	0.0153	94.6	95.5
Fre3_walking_static	0.4165	0.1742	0. 0171	0.0043	94.8	97.5

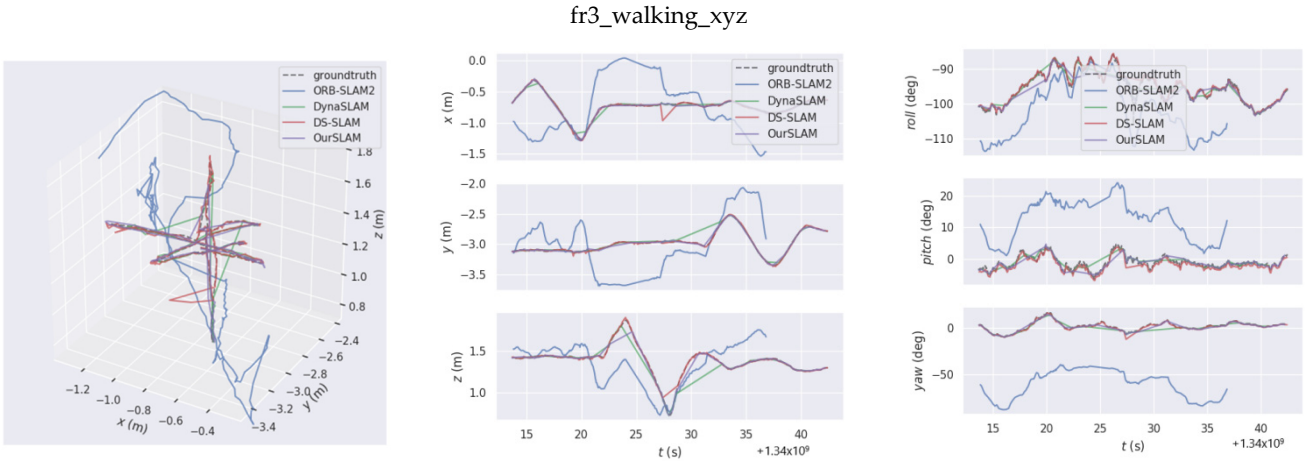
Table 2. Comparison results of relative positional error (RPE).

Datasets	ORB-SLAM2		OurSLAM		Performance Improvement/%	
	RMSE	***SD***	***RMSE***	***SD***	η_{RMSE}	η_{SD}
fre3_Warking_xyz	0.0489	0.5243	0.0198	0.0095	59.5	98.2
fre3_walking_halfsphere	0.0536	0.5894	0.0162	0.0238	69.8	96.0
fre3_walking_static	0.0498	0.4316	0.0186	0.0061	62.7	98.6

In Tables 1 and 2, the *RMSE* and *SD* results obtained by the proposed algorithm and ORB-SLAM2 algorithm are compared and analyzed, and the performance improvement of the proposed algorithm compared with the ORB-SLAM2 algorithm is obtained. As can be seen from Table 1, in terms of the comparison results of absolute trajectory errors, the *RMSE* of the proposed algorithm is 97.8% higher than that of the ORB-SLAM2 algorithm, and

the *SD* of the proposed algorithm is 97.8% higher than that of the ORB-SLAM2 algorithm. It can be seen from Table 2 that the *RMSE* of the proposed algorithm has a maximum performance improvement of 69.8% compared with that of the ORB-SLAM2 algorithm, and the *SD* of the proposed algorithm has a maximum performance improvement of 98.6% compared with that of the ORB-SLAM2 algorithm.

As shown in Figure 7, this paper uses the fr3_walking_xyz datasets to predict the trajectory of ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM as shown below.



(**a**) Track run comparison results (**b**) Fitting results on X, Y, and Z axes (**c**) yaw, pitch, roll trajectory fitting results

Figure 7. Comparison of ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM trajectory map operation results.

As shown in Figure 7, this paper uses different colored lines to represent the trajectory images of four different algorithms: ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM. As shown in Figure 7a, the dotted line represents the real trajectory. The blue line represents the track of the ORB-SLAM2 algorithm; the green line represents the track of the DynaSLAM algorithm; the red line represents the track of the DS-SLAM algorithm; and the purple line represents the trajectory of the OurSLAM algorithm. As can be seen from Figure 7a, the trajectory of the ORB-SLAM2 algorithm in the dynamic environment does not coincide with the real trajectory at all. It can also be seen from Figure 7b,c that the fitting effect of the ORB-SLAM2 algorithm in all directions is quite different from that of the real trajectory. Although DynaSLAM and DS-SLAM are much better than ORB-SLAM2 in dynamic environment mapping, we can see from Figure 7b,c that when there is a breakpoint in the real trajectory, that is, when the camera suddenly changes its angle, the frame loss of both DynaSLAM and DS-SLAM algorithm is serious, and the trajectory will have a large deviation. It can be seen from Figure 7 that compared with the three SLAM algorithms of ORB-SLAM2, DynaSLAM, and DS-SLAM, the OurSLAM algorithm proposed in this paper has obvious advantages in trajectory prediction. The trajectory of the OurSLAM algorithm fits well with the real trajectory in all directions, and compared with the other three SLAM algorithms, the frame loss phenomenon of the camera in this algorithm is significantly improved.

In this paper, two datasets (fr3_walking_xyz and fr3_walking_halfsphere) are used to verify the results of this algorithm, and to compare it with the advanced SLAM algorithms in recent years. Compared with the ORB-SLAM2, DynaSLAM, and DS-SLAM algorithms, the performance of this algorithm is better than the other three algorithms when building maps in a dynamic environment, and it can well meet the needs of dynamic mapping in the SLAM system. In pose estimation, Absolute Pose Error (APE) and RPE are usually used to prove the advantages and disadvantages of an algorithm.

In performing the error comparison, the results of the algorithm testing using the datasets in this paper are as follows. The results of the tests performed on the fre3_walking_xyz dataset are shown in Figure 8.

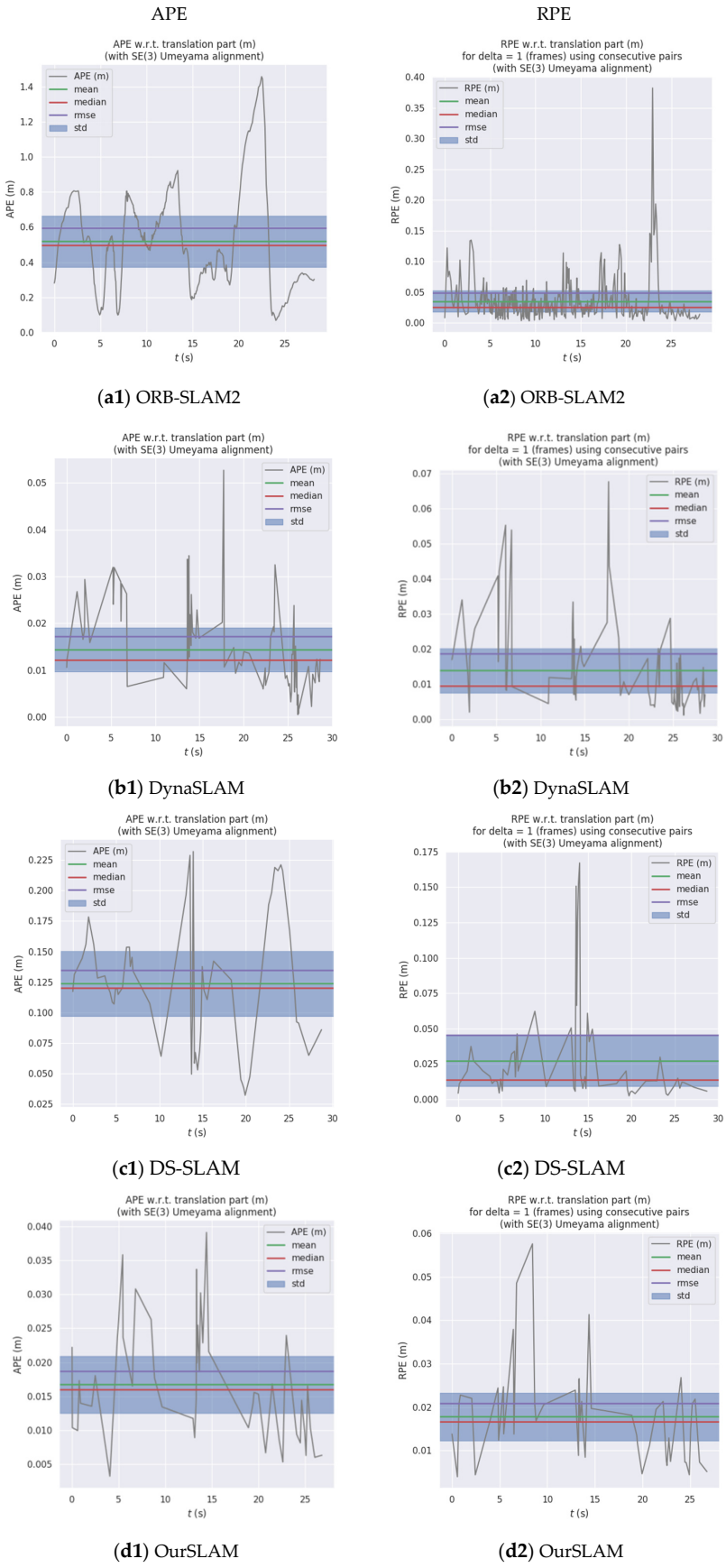
As can be seen from Figure 8, compared with the ORB-SLAM2 algorithm, DynaSLAM, DS-SLAM, and OurSLAM are superior to ORB-SLAM2 in absolute and relative trajectory errors. Among these, the absolute trajectory error and relative trajectory error of DS-SLAM are large, and it cannot meet the accuracy requirements of the SLAM system. Compared with DynaSLAM, OurSLAM is slightly better than DynaSLAM in absolute trajectory error and relative trajectory error.

From Figure 9, it can be seen that the ORB-SLAM2 algorithm has almost no correct rate in trajectory error. The trajectory error of DynaSLAM is in the range of 0.003–0.039, and the trajectory error is relatively small. The trajectory error of DS-SLAM is relatively large due to drift, and the error range is 0.002–0.269. The trajectory error of OurSLAM is in the range of 0.002–0.043, and the accuracy of the algorithm is almost the same as that of DynaSLAM. OurSLAM has higher accuracy compared with DS-SLAM. The accuracy of OurSLAM in map building and pose estimation is very high, and the error is within the error tolerance. By comparing the trajectory error graphs, it can be seen that the algorithm in this paper outperforms ORB-SLAM2 and DynaSLAM in terms of mapping accuracy and pose estimation.

The experimental results on the fre3_walking_halfsphere dataset are as follows:

As shown in Figure 10, the trajectory predictions of ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM on the fr3_walking_halfsphere dataset are compared. This paper uses lines of different colors to represent the trajectory images of four different algorithms: ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM. As shown in Figure 10a, the dotted line represents the real trajectory; the blue line represents the track of the ORB-SLAM2 algorithm; the green line represents the track of the DynaSLAM algorithm; the red line represents the track of the DS-SLAM algorithm; and the purple line represents the trajectory of the OurSLAM algorithm. As can be seen from Figure 10a, the matching accuracy between the real trajectory and the trajectory of the ORB-SLAM2 algorithm in the dynamic environment is almost zero. Although the matching between the real trajectory and the trajectory of the DynaSlam algorithm and the DS-SLAM algorithm is greatly improved compared with the ORB-SLAM2 algorithm, there are still some shortcomings compared with the algorithm in this paper. It is obvious from Figure 10b that the trajectory of the ORB-SLAM2 algorithm differs too much from the real trajectory on each coordinate axis of x, y, and z. The accuracy of the matching result is very low. The matching results of the trajectories of DynaSLAM and DS-SLAM are roughly the same as those of the real trajectories on each coordinate axis of x, y, and z, and have a certain accuracy, but there is still a certain error in the peak and valley of the image. The matching results of the OurSLAM trajectory on the x, y, and z, axes are consistent with the real trajectory, and have a good matching accuracy. It can be seen from Figure 10c that when trajectory prediction is carried out on the axes of yaw, pitch, and roll, the matching effect of OurSLAM is the best, while that of ORB-SLAM2 is the worst.

The results of the experiments using the fr3_walking_halfsphere dataset are shown in Figure 11. In terms of APE, the APE range of OurSLAM fluctuates between 0.01–0.06. However, the fluctuation range of ORB-SLAM2 reaches a maximum of over 1.2, and the fluctuation range of DynaSLAM reaches a maximum of over 0.12. The fluctuation range of the APE of DS-SLAM is not much different from OurSLAM, but its APE range is 0.00–0.07, reaching maximum of over 0.07. In terms of RPE, this algorithm is better than the other three algorithms. The RPE of OurSLAM fluctuates in the range of 0.01–0.04, while the RPE of DS-SLAM fluctuates in the range of 0.00–0.07, the RPE of DynaSLAM fluctuates in the range of 0.00–0.14, and the RPE of ORB-SLAM2 fluctuates in the range of 0.00–0.12. Therefore, by comparing the APE and RPE of each algorithm, it is obvious that the OurSLAM algorithm has obvious advantages, which proves the effectiveness of the algorithm in this paper.



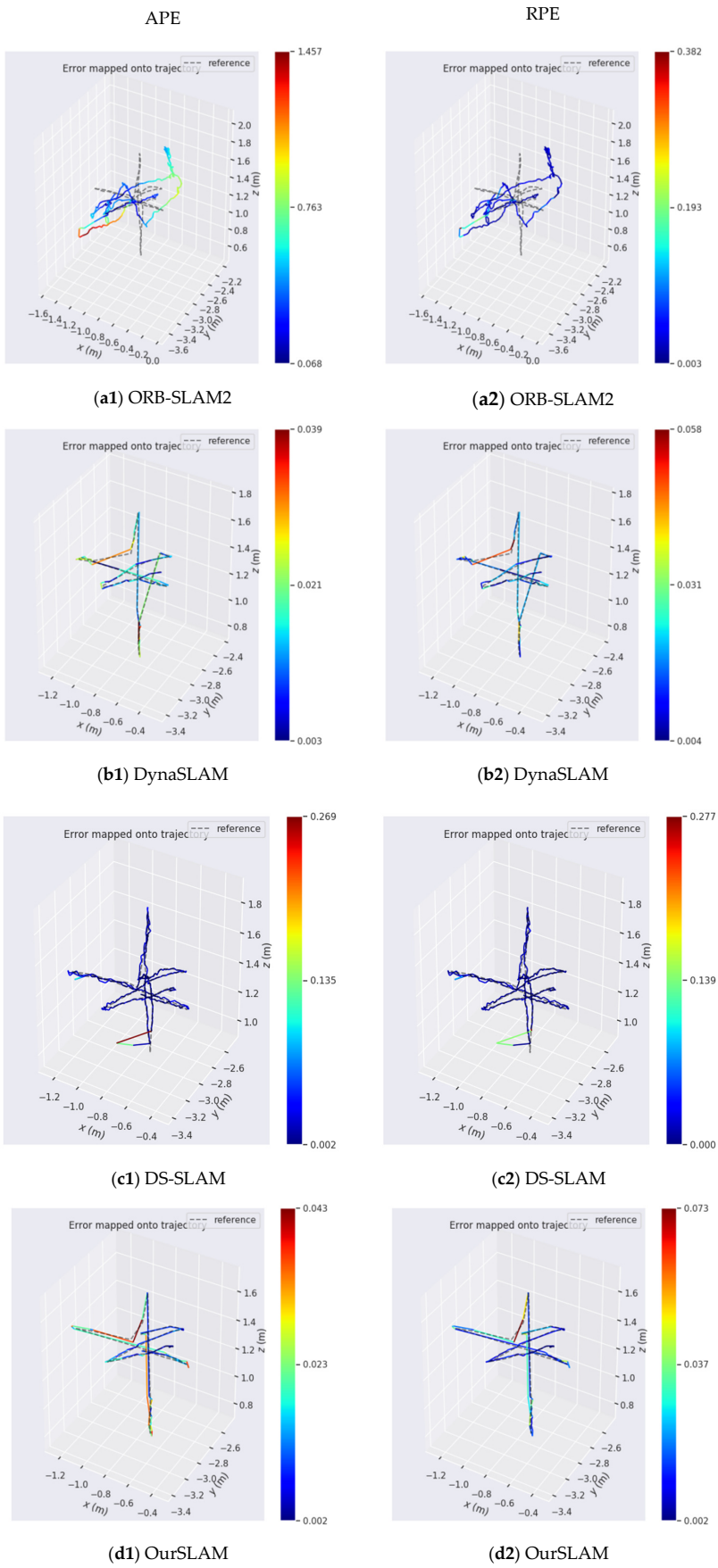
(**a1**) ORB-SLAM2 (**a2**) ORB-SLAM2

(**b1**) DynaSLAM (**b2**) DynaSLAM

(**c1**) DS-SLAM (**c2**) DS-SLAM

(**d1**) OurSLAM (**d2**) OurSLAM

Figure 8. APE, RPE results for ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM on dataset fre3_walking_xyz.



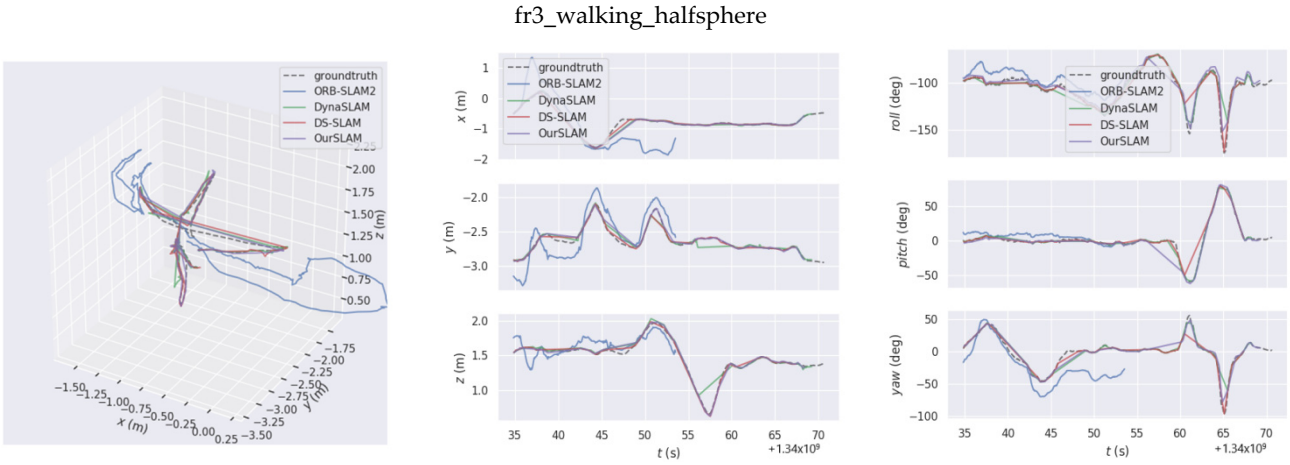
(**a1**) ORB-SLAM2 (**a2**) ORB-SLAM2

(**b1**) DynaSLAM (**b2**) DynaSLAM

(**c1**) DS-SLAM (**c2**) DS-SLAM

(**d1**) OurSLAM (**d2**) OurSLAM

Figure 9. ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM absolute trajectory error mapping to trajectory map results on dataset fre3_walking_xyz.



(**a**) Track run comparison results (**b**) Fitting results on X, Y, and Z axes (**c**) yaw, pitch, roll trajectory fitting results

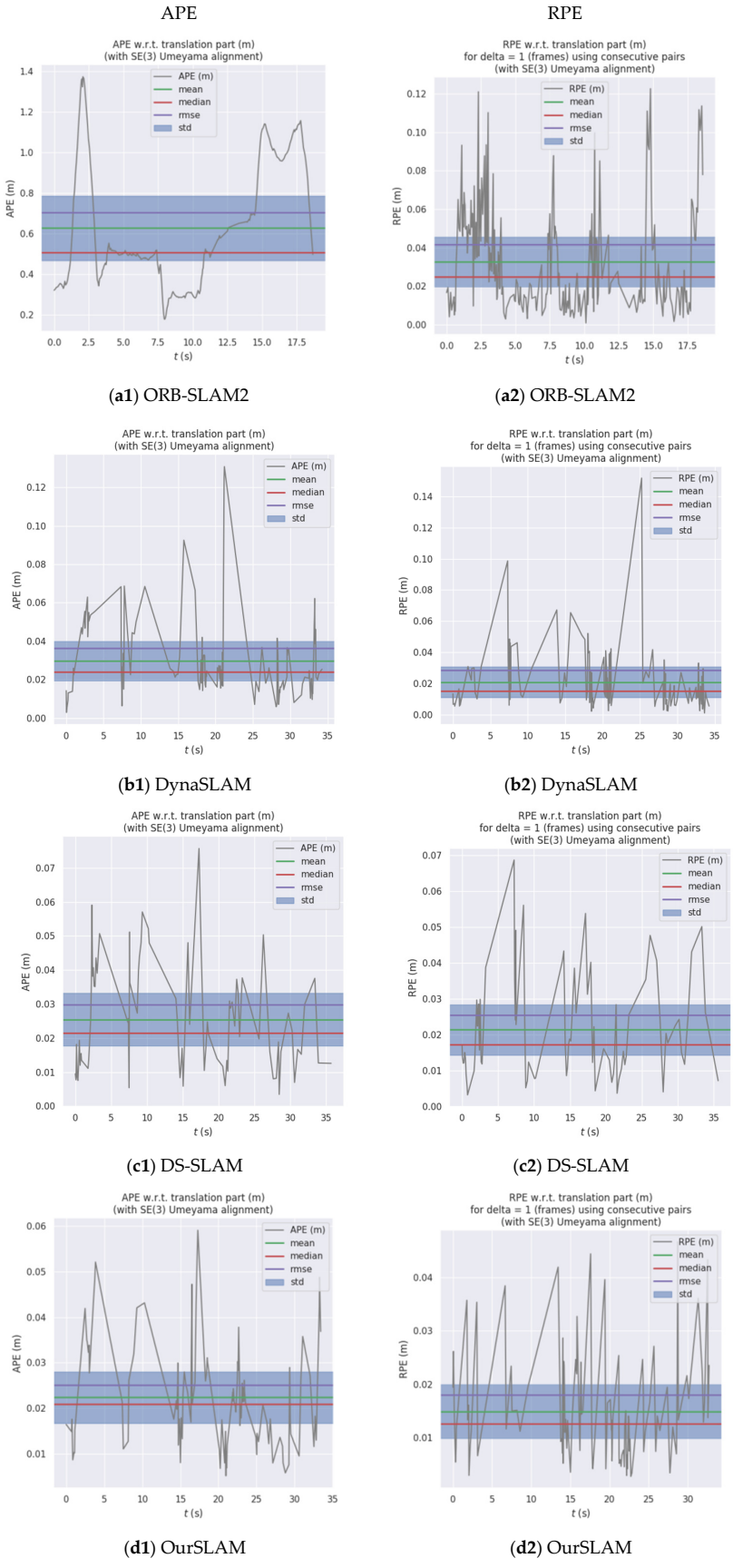
Figure 10. Comparison of ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM trajectory map operation results.

The results of the experiments performed on the fr3_walking_halfsphere dataset are shown in Figure 12. On both APE and RPE, the error trajectories of ORB-SLAM2 and DynaSLAM have obvious red trajectories, so these two algorithms have some errors in performing trajectory prediction. Compared with ORB-SLAM2 and DynaSLAM, a small amount of red traces exist on the error trajectories of the DS-SLAM algorithm, which indicates that DS-SLAM has less error in performing trajectory prediction, and thus that its prediction results are better than those of ORB-SLAM2 and DynaSLAM. The trajectory error results of OurSLAM show nearly all blue lines with almost no red lines. Therefore, it can be shown that the error of this paper's algorithm in trajectory prediction is very small, which proves the high quality of the map building and trajectory prediction results of this paper's algorithm.

When the SLAM system is used to build maps, the real-time performance is often used as an important evaluation index to reflect its merits. So, we compared the time spent per frame by this algorithm and the DynaSLAM algorithm on detecting dynamic points, and the results of all three algorithms are shown in Table 3. It is obvious that the time difference between this algorithm and the ORB-SLAM2 algorithm in processing each frame is not significant. However, the time consumed by this algorithm in processing each frame is significantly lower than the time used by the DynaSLAM algorithm in tracking each frame, so this algorithm has a greater advantage in real-time compared with other SLAM map-building algorithms in dynamic environments.

Table 3. Real-time comparison of SLAM algorithms in dynamic environments.

Datasets	Average Tracking Time per Frame/s			
	OurSLAM	**DynaSLAM**	**DS-SLAM**	**ORB-SLAM2**
fre3_walking_xyz	0.0601	1.127	0.759	0.0372
fre3_walking_halfsphere	0.0645	1.140	0.693	0.0379
fre3_walking_static	0.0632	1.152	0.732	0.0337



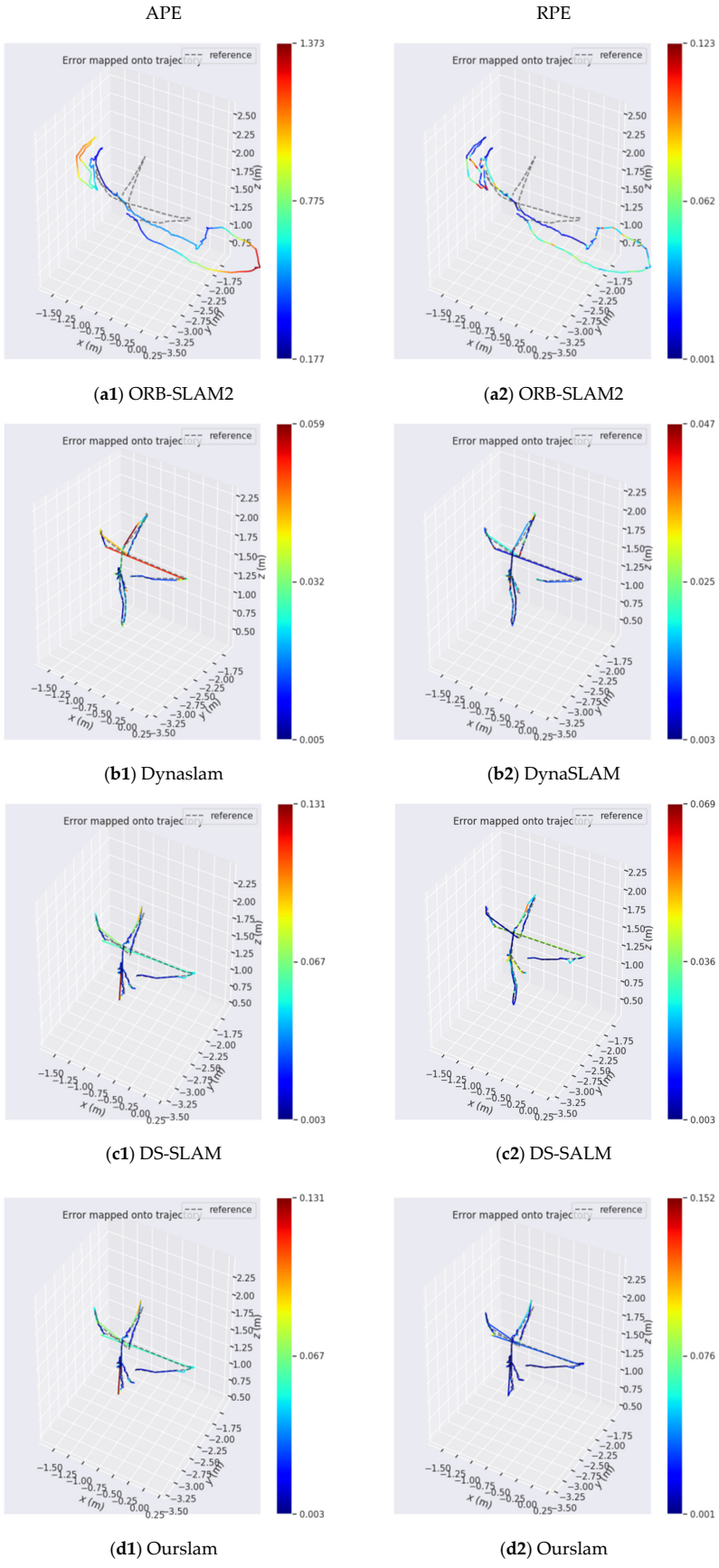
(**a1**) ORB-SLAM2 (**a2**) ORB-SLAM2

(**b1**) DynaSLAM (**b2**) DynaSLAM

(**c1**) DS-SLAM (**c2**) DS-SLAM

(**d1**) OurSLAM (**d2**) OurSLAM

Figure 11. APE, RPE results for ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM on dataset fre3_walking_halfsphere.



(**a1**) ORB-SLAM2 (**a2**) ORB-SLAM2

(**b1**) Dynaslam (**b2**) DynaSLAM

(**c1**) DS-SLAM (**c2**) DS-SALM

(**d1**) Ourslam (**d2**) Ourslam

Figure 12. Error mapping to trajectory results of ORB-SLAM2, DynaSLAM, DS-SLAM, and OurSLAM on dataset fre3_walking_halfsphere.

5. Conclusions

When using the SLAM system to build a map of a dynamic environment, the dynamic features in the environment will affect both the accuracy of building a map and the accuracy of trajectory prediction, resulting in poor accuracies of both the map and the trajectory prediction, or even the failure to build a map at all, which will greatly reduce the accuracy of subsequent positioning and navigation. Therefore, to address the interference of dynamic targets with the SLAM map-building system, this paper improves the system based on the ORB-SLAM2 system. In this paper, a target detection and recognition module based on the YOLOv5 network was added to the ORB-SLAM2 system, which allowed the improved system to not only detect and recognize targets, but also to better detect the dynamic feature points in images. Meanwhile, this paper used the LK optical flow method in the dynamic feature detection module of ORB-SLAM2 for the secondary detection of dynamic objects in images, which increased the accuracy of the system for dynamic feature detection. At present, the mainstream semantic SLAM algorithms, such as DynaSLAM and DS-SLAM, are effective in building maps in dynamic environments, but the semantic segmentation network requires a large amount of computation, which leads to long running times and low real-time performance when building maps, as well as making the algorithms prone to frame loss, resulting in low accuracy of map construction. Compared with semantic SLAM algorithms such as DynaSLAM and DS-SLAM, the dynamic SLAM algorithm based on the YOLOv5 network proposed in this paper is able to detect dynamic features in real-time, can quickly complete the detection of dynamic features, and rarely lose frames due to its fast detection speed, which effectively improves the accuracy of map building and trajectory prediction of the SLAM system.

Although the dynamic SLAM mapping system based on the YOLOv5 network proposed in this paper can eliminate most dynamic objects in the environment and can improve the accuracy of mapping and trajectory prediction, it also has some limitations. When the dynamic objects in the environment are stationary for a long time, they will obscure the static background leading to inaccurate map building, and the algorithm in this paper detects and rejects them. The obscured static background area will have a large error. Therefore, in our future work, our research will mainly focus on recovering the occluded region. The occluded region is first segmented out together with the occluding object using a segmentation network, and then the segmented region is masked into three parts: the occluded background, the occluding object, and the unobscured background. Then the occluding object is removed and the occluded background is predicted according to the unobscured background, in order to recover the occluded background.

Author Contributions: Methodology, X.Z.; Writing—original draft, R.Z.; Writing—review & editing, X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The research data used in this paper are from https://vision.in.tum.de/data/datasets/rgbd-dataset/download (accessed on 1 September 2022).

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

References

1. Abaspur Kazerouni, I.; Fitzgerald, L.; Dooly, G.; Toal, D. A survey of state-of-the-art on visual SLAM. *Expert Syst. Appl.* **2022**, *205*, 117734–117751. [CrossRef]
2. Zhang, S.; Zhi, Y.; Lu, S. Moncular Vision SLAM Research for Parking Environment with low Light. *Int. J. Automot. Technol.* **2022**, *23*, 693–703. [CrossRef]
3. Wen, S.; Liu, X.; Wang, Z.; Zhang, H.; Zhang, Z.; Tian, W. An improved multi-object classification algorithm for visual SLAM under dynamic environment. *Intell. Serv. Robot.* **2021**, *15*, 39–55. [CrossRef]

4. Mur-Artal, R.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
5. Zhang, L.; Zhang, Y. Improved feature point extraction method of ORB-SLAM2 dense map. *Assem. Autom.* **2022**, *42*, 552–566. [CrossRef]
6. Bescos, B.; Facil, J.M.; Civera, J.; Neira, J. DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4076–4083. [CrossRef]
7. Campos, C.; Elvira, R.; Rodriguez, J.J.G.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
8. Zhang, S.; Zheng, L.; Tao, W. Survey and Evaluation of RGB-D SLAM. *IEEE Access* **2021**, *9*, 21367–21387. [CrossRef]
9. Li, S.; Lee, D. RGB-D SLAM in dynamic environments using static point weighting. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2263–2270. [CrossRef]
10. Naudet-Collette, S.; Melbouci, K.; Gay-Bellile, V.; Ait-Aider, O.; Dhome, M. Constrained RGBD-SLAM. *Robotica* **2020**, *39*, 277–290. [CrossRef]
11. Yu, C.; Liu, Z.; Liu, X. DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments. In Proceedings of the 25th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1168–1174.
12. Arshad, S.; Kim, G.W. Role of Deep Learning in Loop Closure Detection for Visual and Lidar SLAM: A Survey. *Sensors* **2021**, *21*, 1243. [CrossRef] [PubMed]
13. Jia, G.; Li, X.; Zhang, D.; Xu, W.; Lv, H.; Shi, Y.; Cai, M. Visual-SLAM Classical Framework and Key Techniques: A Review. *Sensors* **2022**, *22*, 4582. [CrossRef] [PubMed]
14. Xiao, L.; Wang, J.; Qiu, X.; Rong, Z.; Zou, X. Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment. *Robot. Auton. Syst.* **2019**, *117*, 1–16. [CrossRef]
15. Cui, L.; Ma, C. SOF-SLAM: A Semantic Visual SLAM for Dynamic Environments. *IEEE Access* **2019**, *7*, 166528–166539. [CrossRef]
16. Yang, S.; Fan, G.; Bai, L.; Zhao, C.; Li, D. SGC-VSLAM: A Semantic and Geometric Constraints VSLAM for Dynamic Indoor Environments. *Sensors* **2020**, *20*, 2432. [CrossRef]
17. 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]
18. Li, F.; Chen, W.; Xu, W.; Huang, L.; Li, D.; Cai, S.; Yang, M.; Xiong, X.; Liu, Y.; Li, W. A Mobile Robot Visual SLAM System With Enhanced Semantics Segmentation. *IEEE Access* **2020**, *8*, 25442–25458. [CrossRef]
19. Miao, S.; Liu, X.; Wei, D.; Li, C. A Visual SLAM Robust against Dynamic Objects Based on Hybrid Semantic-Geometry Information. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 673. [CrossRef]
20. Yang, S.; Zhao, C.; Wu, Z.; Wang, Y.; Wang, G.; Li, D. Visual SLAM Based on Semantic Segmentation and Geometric Constraints for Dynamic Indoor Environments. *IEEE Access* **2022**, *10*, 69636–69649. [CrossRef]
21. Wu, W.; Guo, L.; Gao, H.; You, Z.; Liu, Y.; Chen, Z. YOLO-SLAM: A semantic SLAM system towards dynamic environment with geometric constraint. *Neural Comput. Appl.* **2022**, *34*, 6011–6026. [CrossRef]
22. Xie, W.; Liu, P.X.; Zheng, M. Moving Object Segmentation and Detection for Robust RGBD-SLAM in Dynamic Environments. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 1–8. [CrossRef]
23. Chen, Q.; Yao, L.; Xu, L.; Yang, Y.; Xu, T.; Yang, Y.; Liu, Y. Horticultural Image Feature Matching Algorithm Based on Improved ORB and LK Optical Flow. *Remote Sens.* **2022**, *14*, 4465. [CrossRef]
24. Long, X.; Zhang, W.; Zhao, B. PSPNet-SLAM: A Semantic SLAM Detect Dynamic Object by Pyramid Scene Parsing Network. *IEEE Access* **2020**, *8*, 214685–214695. [CrossRef]
25. Hastürk, Ö.; Erkmen, A.M. DUDMap: 3D RGB-D mapping for dense, unstructured, and dynamic environment. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 17298814211016178. [CrossRef]
26. Yan, L.; Hu, X.; Zhao, L.; Chen, Y.; Wei, P.; Xie, H. DGS-SLAM: A Fast and Robust RGBD SLAM in Dynamic Environments Combined by Geometric and Semantic Information. *Remote Sens.* **2022**, *14*, 795. [CrossRef]
27. Dai, W.; Zhang, Y.; Zheng, Y.; Sun, D.; Li, P. RGB-D SLAM with moving object tracking in dynamic environments. *IET Cyber-Syst. Robot.* **2021**, *3*, 281–291. [CrossRef]
28. El Bouazzaoui, I.; Florez, S.A.R.; El Ouardi, A. Enhancing RGB-D SLAM Performances Considering Sensor Specifications for Indoor Localization. *IEEE Sens. J.* **2022**, *22*, 4970–4977. [CrossRef]
29. 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. [CrossRef]
30. Li, G.-H.; Chen, S.-L. Visual Slam in Dynamic Scenes Based on Object Tracking and Static Points Detection. *J. Intell. Robot. Syst.* **2022**, *104*, 33–43. [CrossRef]
31. Bruno, H.; Colombini, E. LIFT-SLAM: A deep-learning feature-based monocular visual SLAM method. *Neurocomputing* **2021**, *455*, 97–110. [CrossRef]
32. Mu, R.; Zeng, X. A Review of Deep Learning Research. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 1738–1765. [CrossRef]
33. Jung, H.-K.; Choi, G.-S. Improved YOLOv5: Efficient Object Detection Using Drone Images under Various Conditions. *Appl. Sci.* **2022**, *12*, 7255. [CrossRef]
34. Song, Q.; Li, S.; Bai, Q.; Yang, J.; Zhang, X.; Li, Z.; Duan, Z. Object Detection Method for Grasping Robot Based on Improved YOLOv5. *Micromachines* **2021**, *12*, 1273. [CrossRef] [PubMed]

35. Lee, T.-Y.; Jeong, M.-H.; Peter, A. Object Detection of Road Facilities Using YOLOv3 for High-definition Map Updates. *Sens. Mater.* **2022**, *34*, 251–260. [CrossRef]
36. Hu, Y.; Liu, G.; Chen, Z.; Guo, J. Object Detection Algorithm for Wheeled Mobile Robot Based on an Improved YOLOv4. *Appl. Sci.* **2022**, *12*, 4769. [CrossRef]