applied sciences

MDPI

Article

Visual Simultaneous Localization and Mapping Optimization Method Based on Object Detection in Dynamic Scene

Yongping Zhu, Pei Cheng *, Jian Zhuang, Zhengjia Wang and Tao He

School of Mechanical Engineering, Hubei University of Technology, Wuhan 430068, China
* Correspondence: chengpei1004@outlook.com

Abstract: SLAM (Simultaneous Localization and Mapping), as one of the basic functions of mobile robots, has become a hot topic in the field of robotics this year. The majority of SLAM systems in use today, however, disregard the impact of dynamic objects on the system by defining the external environment as static. A SLAM system suitable for dynamic scenes is proposed, aiming at the issue that dynamic objects in real scenes can affect the localization accuracy and map effect of traditional visual SLAM systems. Initially, the enhanced lightweight YOLOv5s target detection algorithm is employed to detect dynamic objects in each frame of the image. Simultaneously, an assessment is conducted on the feature points present on dynamic objects to determine their potential impact on system accuracy, subsequently guiding the decision to retain or exclude these feature points. The preserved static feature points are then utilized for pose estimation and map construction. Experiments on the publicly available TUM dataset and the KITTI dataset are conducted to compare the system in this paper with ORB-SLAM 3, DS-SLAM, and DynaSLAM, and the algorithm is verified to have better performance.

Keywords: visual SLAM; dynamic scenes; object detection; YOLOv5s

Citation: Zhu, Y.; Cheng, P.; Zhuang, J.; Wang, Z.; He, T. Visual Simultaneous Localization and Mapping Optimization Method Based on Object Detection in Dynamic Scene. *Appl. Sci.* **2024**, *14*, 1787. https://doi.org/10.3390/app14051787

Academic Editor: João M. F. Rodrigues

Received: 21 December 2023
Revised: 12 February 2024
Accepted: 14 February 2024
Published: 22 February 2024

1. Introduction

SLAM (Simultaneous Localization and Mapping) is a robotic methodology employed when a robot operates in an unknown environment [1]. Utilizing onboard sensors such as cameras, inertial sensors, and laser sensors, the robot captures information about the external surroundings [2,3]. These acquired data are then used in real-time to estimate the robot's own pose and simultaneously construct a map of the surrounding environment. The camera possesses characteristics such as compactness, low-power consumption, and cost-effectiveness. Simultaneously, it has the capability to capture a more comprehensive set of environmental information. Consequently, the adoption of cameras as sensors for visual SLAM has emerged as a prominent and actively pursued research direction [4].

Research scholars have proposed several visual SLAM system frameworks [5], including the VINS-Mono [6] and ORB-SLAM series, which have shown promising results thus far [7]. Among them, the ORB-SLAM series has drawn a lot of interest from academics studying this area because of its benefits, which include strong stability and real-time performance. Mur-Artal et al., released the open-source ORB-SLAM2 in 2017 that offers three camera modes for greater versatility [8]. In 2021, they even unveiled ORB-SLAM3 with a multi-map system that incorporates a feature-based inertial visual odometry that greatly enhances the system's localization accuracy [9]. However, there are still certain issues with ORB-SLAM3 in practical situations [10]. While moving objects in actual real-world environments can cause errors in the correlation of visual odometry data, existing algorithms typically assume the external environment to be static [11]. This affects the accuracy of the SLAM algorithm [12]. As a result, studying SLAM algorithms in dynamic environments is very crucial [13,14].

To improve the robustness and accuracy of the SLAM system in dynamic environments, many researchers optimize the algorithm based on ORB-SLAM [15]. DynaSLAM [16], as

proposed by Bescos et al., is a hybrid semantic segmentation and multi-view geometry method that segments dynamic objects by instances using semantic segmentation networks. This allows for precise feature point culling on dynamic objects and background repair. On the basis of ORB-SLAM2, DS-SLAM is integrated with a semantic segmentation network that greatly enhances localization accuracy in dynamic scenes by minimizing the impact of dynamic targets on the system via the motion consistency detection method [17]. DM-SLAM leverages both the distributed and local RANSAC (Random Sample Consensus) to extract static features from dynamic scenes while incorporating them into the system's initialization [18], based on the characteristics of both statics and object dynamics. In dynamic scenes, the system performs better than conventional SLAM systems. Zhu et al. proposed a visual SLAM using a deep feature matcher to extract features in each image frame by a deep feature extractor in a visual odometer and proposed a camera position fusion estimation method [19], which is able to operate at 0.08 s per frame with low error. Wei et al. optimized the visual SLAM algorithm through semantic information, based on the framework of ORB-SLAM2, and added the dynamic region detection module into the visual odometer to improve the accuracy of the system's position estimation in dynamic environments. Then, the image is semantically segmented by the BiSeNetV2 network, dynamic objects are removed using semantic information and dynamic regions, and finally a map containing semantic information is constructed [20].

The existing SLAM system is susceptible to the influence of dynamic objects in dynamic scenes, which leads to a decrease in the system's localization accuracy. In this research, we propose a SLAM system for dynamic situations that determines whether a feature point influences the system accuracy and whether it should be rejected or retained [21]. The system detects a feature point on a dynamic object using an enhanced lightweight YOLOv5-based method [22]. After that, position is estimated using the feature points that were kept, and a dense point cloud map is produced to satisfy navigation and path planning requirements [23]. The experimental findings demonstrate the good real-time performance and ability of the algorithm presented in this research to increase the system accuracy in dynamic circumstances [24].

The rest of this paper is organized as follows. The general architecture of the SLAM system used in this work is described in Section 2, along with the guiding ideas and purposes of each module. The elements of the target detection network and the improvement techniques are covered in detail in Section 3. Section 4 provides a description and analysis of the comparative experiments that were conducted to evaluate the system's performance. In Section 5, the research of this paper is discussed. Finally, in Section 6, the work is summarized, along with an outlook for future work.

2. System Description

The system framework of this paper as shown in Figure 1 can be divided into five main components, i.e., tracking thread, localized mapping, loop closing, dynamic object detection thread, and dense map building.

The tracking thread is in charge of tracking the localization and orientation of the camera in real-time. This component matches features using ORB (Oriented FAST and Rotated BRIEF) feature points to estimate the camera pose. Pose estimation, descriptor matching, and feature extraction are steps in the feature matching process.

For the purpose of creating a local map, keyframe insertion and deletion are mainly handled by the local mapping thread. It also optimizes the data generated by the tracking thread to increase localization accuracy [25].

The job of the loop closing thread is to check if the camera has returned to a previously visited area on a regular basis, which would indicate a loop closure event. The loop closure detection thread initiates a loop closure operation in the event that a loop closure is found.

The dynamic object detection thread uses the YOLOv5s network as its main body, replacing YOLOv5's Backbone with MobileNetV3-Small to lighten the overall network structure [26].

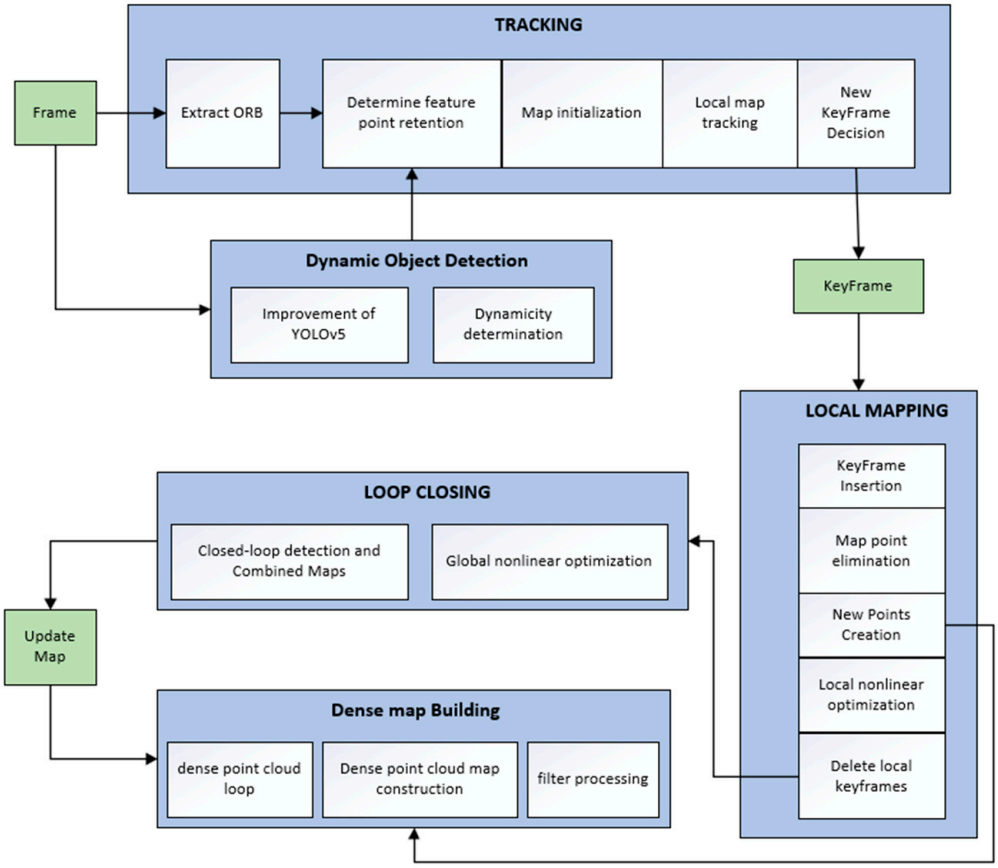


Figure 1. Framework overview of our proposed SLAM.

To create the dense point cloud map, the thread responsible for building the dense map must first convert the depth and image information into 3D point cloud data. Next, it must color the point cloud using the RGB image's color information and stitch the point cloud together using keyframes.

The tracking thread extracts ORB feature points from the input image. ORB is an algorithm that identifies and describes important feature points in an image. The two basic components of ORB feature extraction are BRIEF (Binary Robust Independent Elementary Features) feature description and FAST (Features from Accelerated Segment Test) feature point detection.

FAST is a high-speed algorithm for detecting corner points in an image. The FAST algorithm first selects one pixel in the image to act as the center pixel before determining corner points. To determine whether this center pixel is a corner point, its intensity values are compared to those of the surrounding pixels. As shown in Figure 2, the center pixel point p has a gray scale of I_P, and the FAST algorithm selects a fixed number of surrounding pixels, usually 16 pixels. It then compares the gray values of these surrounding pixels with the gray value of the center pixel. If there are 12 consecutive points whose gray values satisfy Equation (1), then the point is a corner point.

$$\sum_{x \in circle(p)} |I_X - I_P| > T \tag{1}$$

where T is the threshold value and I_x represents the gray level of the surrounding 16 pixel points.

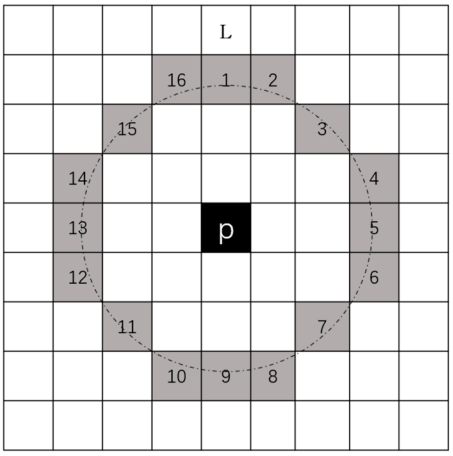


Figure 2. FAST feature point.

The algorithm known as BRIEF is chosen by ORB to describe the feature. BRIEF's primary objective is to produce robust and efficient binary feature descriptors that may be used to match and identify important or distinctive points in a picture. BRIEF selects a feature point and determines a $T \times T$ window and then randomly selects n pairs of pixel points defined as

$$\tau(p:x,y) = \begin{cases} 1 & p(x) < p(y) \\ 0 & p(x) \geq p(y) \end{cases} \tag{2}$$

where $p(x)$ and $p(y)$ are the gray values at point x and y, respectively. Then, the descriptor of feature point p is defined as

$$f_n(p) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(p:x_i,y_i) \tag{3}$$

One of the key components of the system is feature matching; the accuracy of the location estimation can only be ensured when the feature points are matched accurately and efficiently. The camera position problem must be solved when feature matching is finished, and the method in this work uses the ICP (Iterative Closest Point) algorithm to estimate the position of 3D feature points. The principle is as follows: assuming that the two known frames are F_1 and F_2, where $P = \{p_1, \cdots, p_2\} \in F_1$ and $Q = \{q_1, \cdots, q_2\} \in F_2$ are a set of well-matched three-dimensional points, and the camera's pose is calculated by SVD decomposition, the rotation matrix R and translation matrix t are made to satisfy Equation (4).

$$\forall i, p_i = Rq_i + t \tag{4}$$

The local mapping thread is responsible for optimizing keyframes through operations such as eliminating outliers, generating new map points, and removing redundant keyframes. Subsequently, the optimized keyframes are output to the loop closing thread. The loop closing thread, using the keyframe positioned at the front of the buffer queue, replaces it with the currently detected loop closure keyframe. The similarity between the current keyframe and the consensus keyframe is computed to determine the presence of a loop closure.

According to the depth information and image information provided by the camera, the two-dimensional coordinate points are converted into three-dimensional point cloud data. $X_i = [x,y,z]$ represents the coordinates of the points, $X = \{x_1, x_2, \cdots, x_n\}$ represents the set of spatial points, and the corresponding pixel coordinate of the known three-dimensional points under the camera coordinate system is expressed as $X_i = [u, v, 1]$. According to the camera's pinhole model imaging principle, we can obtain the point cloud. The spatial

position information of the point cloud can be obtained according to the pinhole model imaging principle of the camera:

$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K(R|t)\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{5}$$

where K represents the camera internal reference; R represents the rotation matrix; t represents the translation vector; and z is the scale factor between the depth value and the actual spatial distance.

In this paper, the obtained point cloud data are processed by the PCL (Point Cloud Library). The p_{ix}, p_{iy}, and p_{iz} of the point cloud p_i can be calculated according to the following equation:

$$\begin{bmatrix} p_{iz} \\ p_{ix} \\ p_{iy} \end{bmatrix} = \begin{bmatrix} d/depthScale \\ (n - c_x) * p_{iz}/f_x \\ (n - c_y) * p_{iz}/f_x \end{bmatrix} \tag{6}$$

For every keyframe, a corresponding point cloud is created. All of the point clouds are then stitched together using the system's keyframe position data, and the RGB image's color information is utilized to give the point clouds their color, creating a globally dense point cloud map.

3. Dynamic Object Detection

3.1. YOLOv5

The YOLO (You Only Look Once) series is a superb example of a target detection algorithm that is easier to use, quicker, and more accurate than conventional algorithms [27,28]. Compared to YOLOv4, YOLOv5 optimizes the network model structure and data augmentation, which increases the accuracy and speed of detection [29].

The overall structure of YOLOv5 is mostly made up of the input end, the Backbone, the Neck, the prediction, and the output end, as seen in Figure 3. Mosaic data enhancement is used on the input side to enhance the background and small targets of the detected objects by randomly scaling, cropping, and arranging the images. Backbone is enhanced with the addition of Focus and CSP (Cross-Stage Partial) structures. The convolutional neural network layer of feature extraction uses the Focus structure, while the CSP structure efficiently lowers the parameter and computation to increase feature extraction efficiency. Prediction is utilized as the output side and Neck is used for the feature fusion network. Neck uses both PAN (Path Aggregation Network) and SPP (Spatial Pyramid Pooling) structures [30]. Based on the network's width and depth, YOLOv5 can be divided into five versions: YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x, and YOLOv5n. YOLOv5s is the base version utilized in this paper.

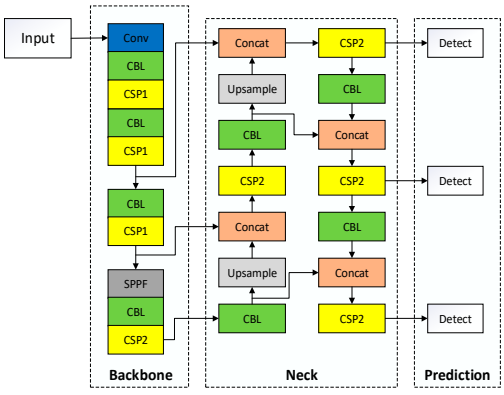


Figure 3. YOLOv5s basic framework.

3.2. Lightweight Networking MobileNet-V3

The real-time performance of traditional convolutional neural networks on embedded systems and mobile devices is limited by their high memory requirements. Specifically designed for image identification and computer vision tasks requiring little processing resources on mobile devices and embedded systems, MobileNet is a lightweight deep learning neural network architecture. MobileNet's design objective is to minimize the size of the model and the network's computational load while preserving high recognition performance to meet the resource requirements of mobile devices with limited resources. Conventional convolution, as illustrated in Figure 4, uses a convolution kernel to carry out a convolution operation on each input channel. Four feature maps with the same size as the input layer are produced as the final result of a convolutional layer with four filters, assuming the input layer is a three-channel color image of 64 by 64 pixels.

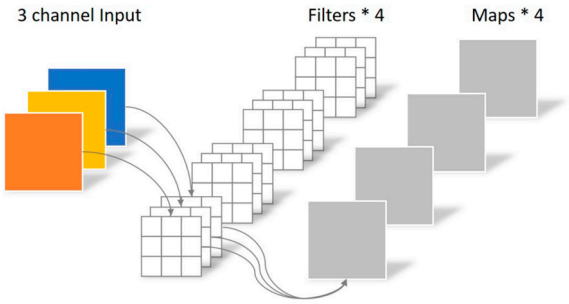


Figure 4. Conventional convolutional operations.

As Figure 5 illustrates, MobileNet reduces computational complexity and parameter count while capturing feature information effectively by using depth separable convolution, which splits the standard convolution into two steps: depth convolution and pointwise convolution.

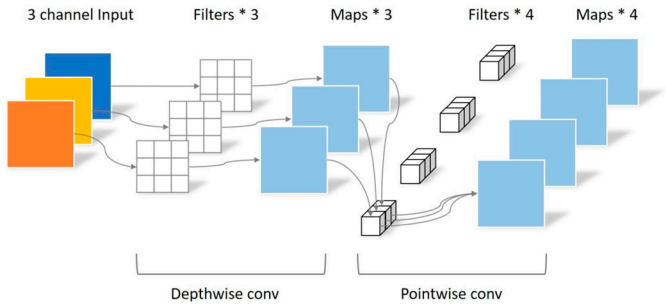


Figure 5. Deep separable convolution.

In 2019, Google released MobileNet-V3, which introduces a number of key improvements and optimizations to MobileNet-V2 for better performance and efficiency. As Figure 6 illustrates, NAS (Neural Architecture Search) technology is used to optimize the MobileNet-V3 network to perform better at particular tasks. Figure 6 illustrates this process by finding the optimal network structure to increase performance metrics. Second, MobileNet-V3 uses a lightweight activation function dubbed Hard Swish, which increases the model's nonlinearity while occasionally outperforming traditional ReLU. In the meantime, MobileNet-V3 presents the SE (Squeeze-and-Excitation) module, which uses an attention mechanism to execute adaptive weight adjustment of the feature map during training. The image input is first processed through a 1×1 convolution to increase the number of channels; next, the feature map data are optimized using the SE attention mechanism; finally, the number of channels is decreased through a 1×1 convolution under high-dimensional space. The residuals are used to connect the input and output when the step size is equal to 1 and the

input and output feature maps are of the same shape; when the step size is equal to 2, the output is the downscaled feature map directly.

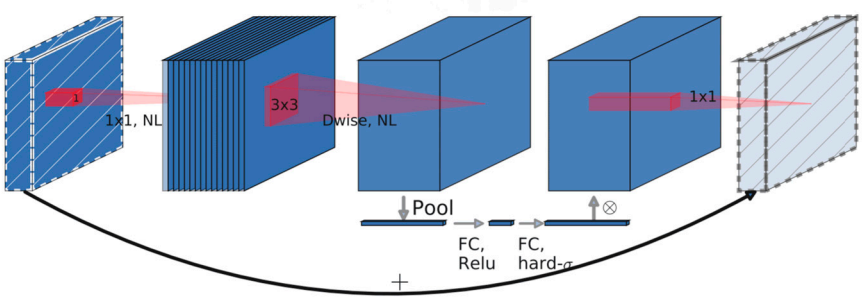


Figure 6. Structure of MobileNet-V3 network.

3.3. Improvement in YOLOv5s Target Detection Network

To achieve the lightweighting of the YOLOv5s network, the Backbone of YOLOv5s is replaced with MobileNetV3-Small, a specific configuration in the MobileNetV3 family designed to offer a smaller model size and a lighter computational burden for resource-constrained mobile devices and embedded systems. It is a variant of the MobileNetV3 family and is particularly suitable for applications that require higher model size and computational efficiency. The MobileNetV3-Small network structure is shown in Table 1.

Table 1. MobileNetV3-Small network structure.

Input	Operator	Exp Size	OUT	SE	NL	S
2242 × 3	conv2d, 3 × 3	_	16	_	HS	2
1122 × 16	bneck, 3 × 3	16	16	YES	RE	2
562 × 16	bneck, 3 × 3	72	24	_	RE	2
282 × 24	bneck, 3 × 3	88	24	_	RE	1
282 × 24	bneck, 5 × 5	96	40	YES	HS	2
142 × 40	bneck, 5 × 5	240	40	YES	HS	1
142 × 40	bneck, 5 × 5	240	40	YES	HS	1
142 × 40	bneck, 5 × 5	120	48	YES	HS	1
142 × 48	bneck, 5 × 5	144	48	YES	HS	1
142 × 48	bneck, 5 × 5	288	96	YES	HS	2
72 × 96	bneck, 5 × 5	576	96	YES	HS	1
72 × 96	bneck, 5 × 5	576	96	YES	HS	1
72 × 96	conv2d, 1 × 1	_	576	YES	HS	1
72 × 576	pool, 7 × 7	_	_	_	_	1
12 × 576	conv2d 1 × 1, NBN	_	1024	_	HS	1
12 × 1024	conv2d 1 × 1, NBN	_	K	_	_	1

As shown in the table, MobileNetV3-Small consists of a combination of several convolutional layers. Input denotes the shape of the feature matrix of the input current layer; Operator denotes the convolutional operation (conv2d is the normal convolutional layer; bneck is the BN layer, a combination of the convolutional layer and the activation function; NBN denotes that the BN layer is not used); Out denotes the output channel size; SE denotes the attention module; NL denotes the type of activation function (HS for the h-swish activation function and RE for the ReLU activation function); S denotes the step size; and "-" denotes that the layer is not used and has no parameters. As shown in Figure 7, the Backbone of YOLOv5s is replaced with MobileNetV3-Small and adjusted to generate feature maps at different scales, and the three different scales of feature maps required for the Neck of YOLOv5s are extracted from 3-P3, 8-P4, and 11-P5. The CBL (Convolutional Block Layer) mainly consists of a convolutional layer and a batch normalization layer, which is used for feature extraction and normalization. CSP splits the original input into two branches and performs convolutional operations separately to halve the number of

channels, allowing the model to learn more features. SPP can convert feature maps of arbitrary size into feature vectors of fixed size.

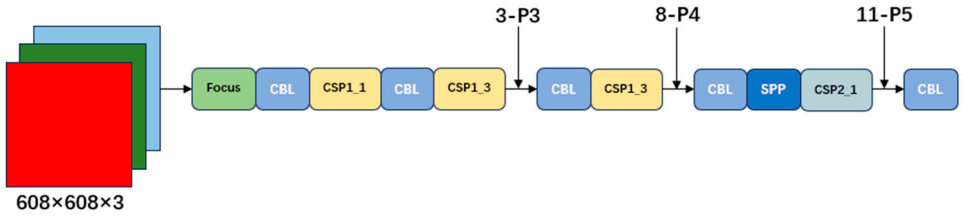


Figure 7. Backbone of improved YOLOv5s.

3.4. Methods of Determining Information

When the feature points in the tracking thread of a classical SLAM system move, it can cause a build-up of errors in the camera position computation. This can ultimately impact the system's accuracy and potentially result in localization failure. Generally speaking, objects that are immobile, like people and animals in indoor settings, are considered dynamic objects; on the other hand, stationary objects, like tables and computers, are considered static objects. Table 2 displays the information judgment method developed in this paper. This will prevent the effective feature points in the environment from being mistakenly eliminated. Specifically, only feature points located in the dynamic detection frame and not in the static detection frame are considered to be dynamic feature points, which will impact the accuracy of the system to be eliminated. This paper is simultaneously located in or not located in the dynamic detection frame and static detection frame feature points are considered as effective feature points to be retained.

Table 2. Feature point determination method.

Number	Within the Static Detection Box	Within the Dynamic Detection Box	Determination
1	NO	NO	Retain
2	YES	NO	Retain
3	YES	YES	Retain
4	NO	YES	eliminate

4. Experiment and Analysis

This paper uses the publicly available TUM dataset in addition to the KITTI dataset to validate the algorithm's performance. Image sequences of indoor scenes taken with a Kinect camera and referenced with actual camera poses are included in the TUM dataset. Experiments are also conducted using the highly dynamic sequences walking_xyz, walking_static, and walking_halfsphere on this dataset to confirm the algorithm's accuracy and robustness in actual indoor environments. Sequences 01, 02, and 03 on the KITTI dataset are also utilized for experiments to confirm the accuracy and resilience of the algorithms in actual outdoor scenes. The dataset comprises real image data that were gathered from urban, rural, and highway scenes. For this paper, a laptop running Ubuntu 20.04 with 16 GB of RAM serves as the experimental platform.

4.1. Experiments on Object Detection Models

The object detection network is trained using the VOC-2007 dataset, which consists of 20 different objects including people, chairs, and cars. The dataset for this training consists of 7653 images, 1000 epochs are trained, batch_size is set to 128, Learning_rate is 0.0032, Momentum is 0.843, and Weight_decay is 0.00036.

The images in the walking_halfsphere dataset are detected and the Parameter quantity, Calculation quantity, and Detection time of different target detection networks are shown in Table 3. All the tests are performed five times and the final results are averaged. It can

be observed that using MobileNetV3 instead of the YOLOv5s Backbone feature extraction network reduces the model Parameter quantity by 50.08%, Calculation quantity by 62.63%, and Detection time by 10.71%.

Table 3. Comparison of experimental results of different models.

Sequence	Parameter Quantity (10^6 M)	Calculation Quantity (GFLOPs)	Detection Time (ms)
YOLOv5l	45.65	110.6	29.65
YOLOv5m	22.15	53.6	27.51
YOLOv5s	7.12	18.2	24.83
YOLOv5s-MobileNetV3	3.50	6.8	22.17

4.2. Indoor Posture Estimation Experiment

This paper develops a set of information determination methods to retain the static feature points in the dynamic detection frame. A feature point is considered dynamic only if it is in the dynamic detection frame and not in the static detection frame. Rejecting a feature point based solely on its location will affect the accuracy of the system. Figure 8a illustrates the effect of culling, while Figure 8b displays the image prior to dynamic feature point culling and Figure 8b displays the image subsequent to culling. The man's plaid shirt in the figure has a lot of feature points that, if left unculled, will significantly lower the system's accuracy. This paper's method successfully identifies the dynamic objects in the image and retains the feature points on the static objects, like computers, while rejecting the feature points on the dynamic objects.

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

Figure 8. Dynamic feature point culling effect diagram. (**a**) Before culling, (**b**) after culling.

Robustness in SLAM refers to the system's capacity to withstand a range of errors and disturbances. The accuracy of the system will be affected by the presence of dynamic objects in the dataset, but the system's robustness will be demonstrated by its equally good performance in managing the dynamic dataset. The difference between the calculated and true values of the posture is represented by the Absolute Posture Error (APE), which provides a visual evaluation of the algorithm's correctness. The difference in the amount of position change at the same timestamp is represented by the Relative Posture Error (RPE), which is a useful tool for estimating system drift. After aligning the estimated and actual position values using the timestamps, the change in position is computed between the true and estimated values over time. The difference in the change in values represents the position error of the camera.

Tables 4 and 5 present comparative test results between ORB-SLAM3 and the algorithm proposed in this paper across three different datasets. The evaluation metrics include root-mean-square error (RMSE), mean error (MEAN), and standard deviation (SD). The improvement ratio is calculated using the formula given in Equation (7):

$$Improvement = \frac{m - n}{m} \times 100\% \tag{7}$$

where m is the running result of ORB-SLAM3 and n is the running result of the algorithm in this paper.

Table 4. Absolute Posture Error comparison.

Sequence	ORB-SLAM3 (m)			Our-SLAM (m)			Improvement (%)		
	RMSE	MEAN	SD	RMSE	MEAN	SD	RMSE	MEAN	SD
walking_xyz	0.6459	0.5122	0.4025	0.1290	0.0408	0.0486	80.02	92.09	87.92
walking_static	0.3371	0.2107	0.2489	0.0365	0.0259	0.0285	89.17	87.70	88.54
walking_halfsphere	0.4254	0.4022	0.3715	0.0801	0.0296	0.0653	81.17	92.64	82.42

Table 5. Relative Posture Error comparison.

Sequence	ORB-SLAM3 (m)			Our-SLAM (m)			Improvement (%)		
	RMSE	MEAN	SD	RMSE	MEAN	SD	RMSE	MEAN	SD
walking_xyz	0.4256	0.3245	0.2356	0.1204	0.0563	0.0905	71.71	82.65	61.58
walking_static	0.2245	0.0986	0.1856	0.0160	0.0120	0.0156	92.87	87.82	91.59
walking_halfsphere	0.3661	0.2265	0.3103	0.0436	0.0520	0.0284	88.09	77.04	91.45

From Table 5, it can be seen that the algorithm in this paper has improved the position estimation accuracy compared to ORB-SLAM3 after the dynamic feature points are removed. In terms of absolute trajectory error, the root-mean-square error in the walking_xyz sequence is improved by 80.02%, the mean error is improved by 92.09%, and the standard deviation is improved by 87.92%. In terms of the relative displacement trajectory error, the root-mean-square error in the walking_xyz sequence is improved by 71.71%, the mean error is improved by 82.65%, the standard deviation is improved by 61.58%, and the other two dynamic sequences are also significantly improved.

Figure 9a shows the camera trajectory of ORB-SLAM3 in the walking_xyz sequence, and Figure 9b shows the camera trajectory of our SLAM in the walking_xyz sequence, where the solid line is the trajectory value computed by the system and the dashed line is the real trajectory value of the camera. Figure 9c shows the comparison between ORB-SLAM3 and our SLAM in xyz mode with the real trajectory, and Figure 9d shows the comparison between ORB-SLAM3 and our SLAM in rpy mode with the real trajectory. From the figure, it can be seen that the trajectory calculated by ORB-SLAM3 has a large error, while the trajectory calculated by our SLAM is closer to the real value.

Figure 10a shows the results of the APE comparison between ORB-SLAM3 and our SLAM in the walking_xyz sequence, and Figure 10a shows the results of the RPE comparison between ORB-SLAM3 and our SLAM in the walking_xyz sequence. Our SLAM performs significantly better than ORB-SLAM3 in APE and better than ORB-SLAM3 in RPE. It can be seen that the algorithm in this paper has less error in dynamic environments compared to ORB-SLAM3 and has higher accuracy and robustness.

Table 6 displays the absolute trajectory errors of the algorithm used in this paper in comparison to the DS-SLAM and DynaSLAM algorithms. In the walking_xyz and walking_static sequences, Table 6 shows that the algorithm used in this study performs better than the other algorithms in terms of root-mean-square error, and that DS-SLAM and DynaSLAM are comparable in these two sequences. Meanwhile, in the walking_halfsphere

sequence, DynaSLAM performs better than DS-SLAM with respect to standard deviation and root-mean-square error, while the technique presented in this research comes in second. Comparing the error of this work to similar methods over different dataset sequences, it performs reasonably well overall.

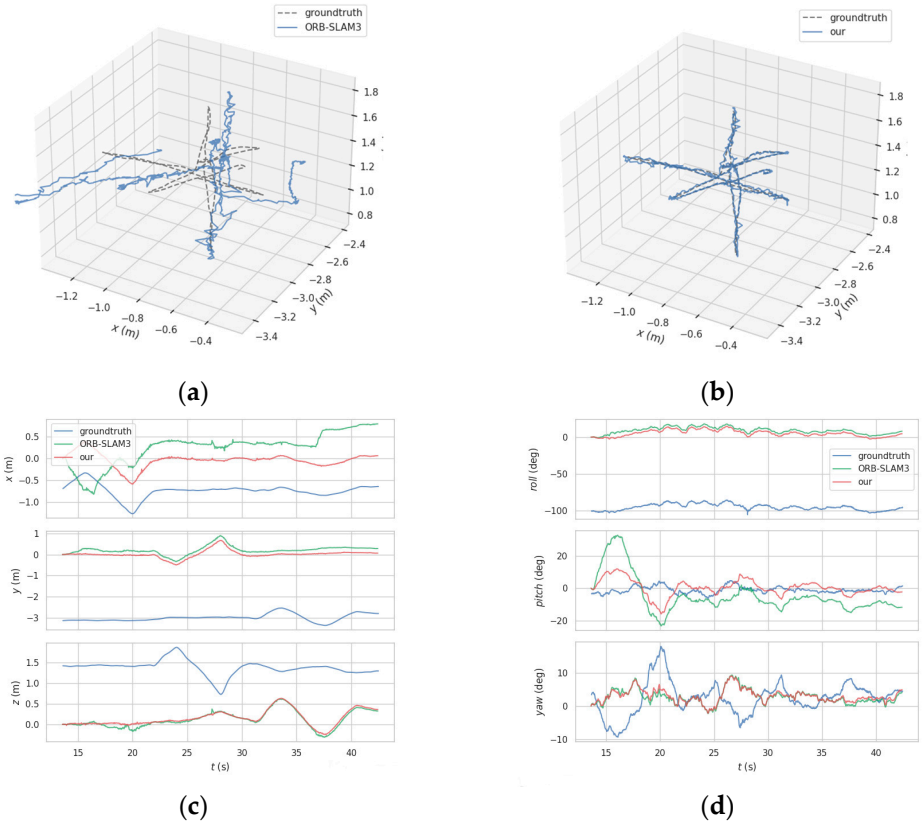


Figure 9. The results of APE in the walking_xyz sequence. (**a**) ORB-SLAM3 comparison results with true trajectories, (**b**) our SLAM comparison results with true trajectories, (**c**) comparison curves between the three trajectories in xyz mode, (**d**) comparison curves between the three trajectories in rpy mode.

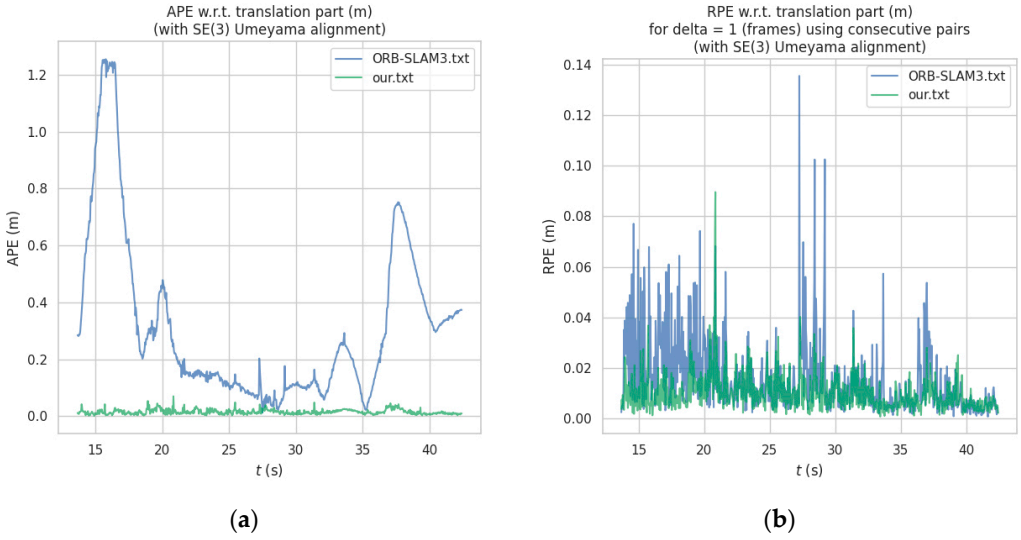


Figure 10. APE and RPE results in walking_xyz sequence. (**a**) APE results, (**b**) RPE results.

Table 6. Comparison of absolute trajectory errors of different algorithms.

Sequence	DS-SLAM (m)			DynaSLAM (m)			Our-SLAM (m)		
	RMSE	MEAN	SD	RMSE	MEAN	SD	RMSE	MEAN	SD
walking_xyz	0.1365	0.0681	0.0369	0.1087	0.0374	0.0426	0.1056	0.0324	0.0456
walking_static	0.0451	0.0652	0.0371	0.0315	0.0368	0.0366	0.0326	0.0231	0.0271
walking_halfsphere	0.1026	0.0892	0.0510	0.0786	0.0350	0.0291	0.0821	0.0298	0.0634

This paper compares the system with ORB-SLAM3, DS-SLAM, and DynaSLAM to validate the system's real-time performance. Table 7 shows that ORB-SLAM3 has the shortest tracking time per frame because it does not process the dynamic objects in the scene. However, the tracking time per frame outperforms DS-SLAM with DynaSLAM in this paper when comparing the systems in similar dynamic scenes.

Table 7. Tracking time comparison.

Sequence	ORB-SLAM3 (ms)		DS-SLAM (ms)		DynaSLAM (ms)		Our-SLAM (ms)	
	Median	MEAN	Median	MEAN	Median	MEAN	Median	MEAN
walking_xyz	24.6	26.8	28.8	30.6	27.5	30.1	28.9	29.5
walking_static	24.5	26.7	28.4	31.2	28.0	30.5	28.4	29.2
walking_halfsphere	24.5	26.4	28.8	31.5	27.1	31.5	29.0	30.2

The comparison of the frame rate of this paper system with ORB-SLAM3, DS-SLAM, and DynaSLAM is presented in Table 8. The table shows that ORB-SLAM3 achieves the highest frame rate of 37 frames per second because it does not detect dynamic objects. When comparing DS-SLAM and DynaSLAM, the system described in this paper has the highest frame rate. This paper's system has the highest frame rate, able to sustain over 33 frames per second while the algorithm runs, demonstrating the improved real-time performance of the algorithm.

Table 8. Frame rate comparison.

Sequence	ORB-SLAM3 (fps)	DS-SLAM (fps)	DynaSLAM (fps)	Our-SLAM (fps)
walking_xyz	37.5	31.2	32.5	33.7
walking_static	37.7	32.5	31.6	34.5
walking_halfsphere	37.6	32.2	32.8	33.2

4.3. Outdoor Posture Estimation Experiment

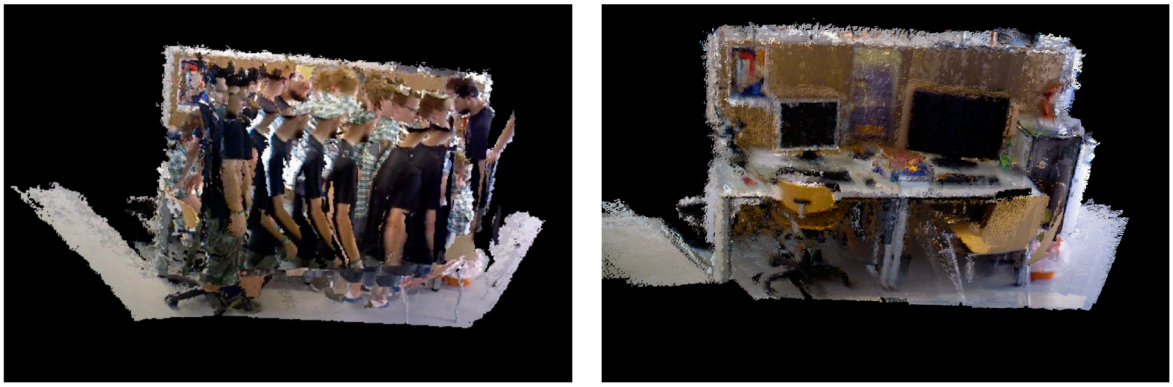
Table 9 presents the absolute trajectory errors for the KITTI dataset comparison between the algorithm presented in this paper and DS-SLAM, DynaSLAM. Table 9 shows that the algorithm used in this paper performs better in terms of root-mean-square error than the other algorithms in the 01 and 02 sequences, and that DS-SLAM and DynaSLAM perform similarly in these two sequences. In the 03 sequence, DynaSLAM performs poorly while DS-SLAM performs best in terms of standard deviation and root-mean-square error. Overall, this paper's errors in various outdoor dataset sequences outperform comparable algorithms.

Table 9. Comparison of absolute trajectory error.

Sequence	DS-SLAM (m)			DynaSLAM (m)			Our-SLAM (m)		
	RMSE	MEAN	SD	RMSE	MEAN	SD	RMSE	MEAN	SD
01	5.263	4.268	4.561	4.325	3.685	4.254	2.256	2.037	1.562
02	0.855	0.971	0.928	0.978	0.869	0.965	0.962	0.798	0.668
03	0.036	0.057	0.068	0.156	0.085	0.093	0.086	0.065	0.075

4.4. Dense Building Maps Experiment

In this paper, the walking_xyz sequence is selected as the image sequence for dense map construction, in which two men walk around a computer desk. If we do not eliminate the dynamic objects to build the map as shown in Figure 11a, we can observe that the dynamic objects in the scene have an impact on the map construction, resulting in a large number of characters in the map residual shadow, which seriously reduces the effect of dense map construction. The algorithm used in this study creates the map that is depicted in Figure 11b, removing the impact of dynamic objects and creating a good map that is suitable for use in robot navigation and path planning.



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

Figure 11. Comparison of maps before and after processing. (**a**) Before culling, (**b**) after culling.

5. Discussion

This work aims to maximize the performance of SLAM systems in dynamic environments by concentrating on the identification and rejection of dynamic feature points. The approach presented in this research performs well in terms of localization errors in dynamic settings, according to experimental results. This benefit is explained by the system's capacity to recognize dynamic objects and lessen their interference. In the meantime, the system suggested in this paper shows a higher frame rate in the frame rate comparison experiments with similar algorithms. This is explained by the fact that MobileNetV3-Small replaces the Backbone of the target detection network YOLOv5s, making the entire network model lighter. Varying application scenarios may require varying levels of accuracy from SLAM algorithms. For example, robots working in indoor environments often have slightly lower accuracy requirements, with absolute trajectory errors needing to be limited within a few centimeters to ten centimeters. The system in this paper can still fulfill this requirement in dynamic scenarios, so the system in this paper has good practicality. While the system presented in this paper has significantly improved the map effect and localization accuracy in dynamic scenes, in real-world applications, tracking failure may occur when features in dynamic environments are absent or the camera moves too quickly. The localization accuracy of the system may be adversely affected in situations with low light or abrupt changes in illumination because the system developed in this paper is a visual SLAM and its localization depends on visual information. Future research can look into adding more sensors, like inertial measurement units (IMUs), and introducing a variety of feature types to improve the system. These upgrades allow the system to adjust to increasingly complicated settings.

6. Conclusions

In this paper, a target-detection-based SLAM system is designed with the goal of minimizing the impact of dynamic objects in the environment on the accuracy of the SLAM system. The target detection network's lightweight MobileNetV3 network replaces the YOLOv5s Backbone network in an effort to reduce the model parameters and computation

of the target detection network. The improved YOLOv5s detects dynamic objects in the environment; in the meantime, this paper proposes a dynamic feature point determination method that can effectively identify the dynamic feature points on the dynamic objects that impact the system's accuracy while retaining the static feature points to avoid mistakenly excluding the effective feature points. Constructing a dense point cloud map concurrently requires the addition of a dense map building thread, and once the dynamic objects are removed, the map construction effect is satisfactory. The system in this paper improves the position estimation accuracy to over 90% when compared to the ORB-SLAM3 position estimation accuracy, and it also partially improves the position estimation accuracy when compared to dynamic SLAM systems like DS-SLAM and DynaSLAM, according to the experimental results on TUM and KITTI datasets. Consequently, there is a good chance that the SLAM system based on the enhanced YOLOv5s presented in this paper will find use. In the future, we will consider optimizing and improving the system by combining multiple sensors to further improve the system accuracy and at the same time adapt the system to more scenarios.

Author Contributions: Conceptualization, Y.Z. and P.C.; methodology, Y.Z., P.C. and T.H.; software, P.C.; validation, Z.W., P.C. and J.Z.; formal analysis, P.C.; investigation, P.C.; resources, P.C.; data curation, P.C.; writing—original draft preparation, P.C. and J.Z.; writing—review and editing, Z.W., P.C. and J.Z.; visualization, J.Z.; supervision, P.C.; project administration, P.C. and J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 51275158.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

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

References

1. Zou, Q.; Sun, Q.; Chen, L.; Nie, B.; Li, Q. A Comparative Analysis of LiDAR SLAM-Based Indoor Navigation for Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 6907–6921. [CrossRef]
2. Junaedy, A.; Masuta, H.; Sawai, K.; Motoyoshi, T.; Takagi, N. Real-Time 3D Map Building in a Mobile Robot System with Low-Bandwidth Communication. *Robotics* **2023**, *12*, 157. [CrossRef]
3. Macario Barros, A.; Michel, M.; Moline, Y.; Corre, G.; Carrel, F. A Comprehensive Survey of Visual SLAM Algorithms. *Robotics* **2022**, *11*, 24. [CrossRef]
4. Ni, J.; Wang, L.; Wang, X.; Tang, G. An Improved Visual SLAM Based on Map Point Reliability under Dynamic Environments. *Appl. Sci.* **2023**, *13*, 2712. [CrossRef]
5. Wu, Z.; Li, D.; Li, C.; Chen, Y.; Li, S. Feature Point Tracking Method for Visual SLAM Based on Multi-Condition Constraints in Light Changing Environment. *Appl. Sci.* **2023**, *13*, 7027. [CrossRef]
6. Qin, T.; Li, P.; Shen, S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
7. Wei, S.; Wang, S.; Li, H.; Liu, G.; Yang, T.; Liu, C. A Semantic Information-Based Optimized vSLAM in Indoor Dynamic Environments. *Appl. Sci.* **2023**, *13*, 8790. [CrossRef]
8. 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]
9. 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]
10. Dias, N.J.B.; Laureano, G.T.; Da Costa, R.M. Keyframe Selection for Visual Localization and Mapping Tasks: A Systematic Literature Review. *Robotics* **2023**, *12*, 88. [CrossRef]
11. Li, Y.; Brasch, N.; Wang, Y.; Navab, N.; Tombari, F. Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6583–6590. [CrossRef]
12. Lajoie, P.; Ramtoula, B.; Chang, Y.; Carlone, L.; Beltrame, G. DOOR-SLAM: Distributed, Online, and Outlier Resilient SLAM for Robotic Teams. *IEEE Robot. Autom. Lett.* **2020**, *5*, 1656–1663. [CrossRef]

13. Shamseldin, M.A.; Khaled, E.; Youssef, A.; Mohamed, D.; Ahmed, S.; Hesham, A.; Elkodama, A.; Badran, M. A New Design Identification and Control Based on GA Optimization for An Autonomous Wheelchair. *Robotics* **2022**, *11*, 101. [CrossRef]
14. Sahoo, B.; Biglarbegian, M.; Melek, W. Monocular Visual Inertial Direct SLAM with Robust Scale Estimation for Ground Robots/Vehicles. *Robotics* **2021**, *10*, 23. [CrossRef]
15. 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]
16. 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]
17. Yu, C.; Liu, Z.; Liu, X.; 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.
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. Zhu, B.; Yu, A.; Hou, B.; Li, G.; Zhang, Y. A Novel Visual SLAM Based on Multiple Deep Neural Networks. *Appl. Sci.* **2023**, *13*, 9630. [CrossRef]
20. Liu, Y.; Miura, J. RDS-SLAM: Real-Time Dynamic SLAM Using Semantic Segmentation Methods. *IEEE Access* **2021**, *9*, 23772–23785. [CrossRef]
21. Islam, R.; Habibullah, H. Place Recognition with Memorable and Stable Cues for Loop Closure of Visual SLAM Systems. *Robotics* **2022**, *11*, 142. [CrossRef]
22. Wang, Y.; Hussain, B.; Yue, C.P. VLP Landmark and SLAM-Assisted Automatic Map Calibration for Robot Navigation with Semantic Information. *Robotics* **2022**, *11*, 84. [CrossRef]
23. Xu, G.; Li, X.; Zhang, X.; Xing, G.; Pan, F. Loop Closure Detection in RGB-D SLAM by Utilizing Siamese ConvNet Features. *Appl. Sci.* **2022**, *12*, 62. [CrossRef]
24. Dwijotomo, A.; Abdul Rahman, M.A.; Mohammed Ariff, M.H.; Zamzuri, H.; Wan Azree, W.M.H. Cartographer SLAM Method for Optimization with an Adaptive Multi-Distance Scan Scheduler. *Appl. Sci.* **2020**, *10*, 347. [CrossRef]
25. Wang, X.; Zhou, Y.; Yu, G.; Cui, Y. A Lightweight Visual Odometry Based on LK Optical Flow Tracking. *Appl. Sci.* **2023**, *13*, 11322. [CrossRef]
26. Guo, Z.; Wang, C.; Yang, G.; Huang, Z.; Li, G. MSFT-YOLO: Improved YOLOv5 Based on Transformer for Detecting Defects of Steel Surface. *Sensors* **2022**, *22*, 3467. [CrossRef] [PubMed]
27. Wu, W.; Liu, H.; Li, L.; Long, Y.; Wang, X.; Wang, Z.; Li, J.; Chang, Y. Application of local fully Convolutional Neural Network combined with YOLO v5 algorithm in small target detection of remote sensing image. *PLoS ONE* **2021**, *16*, e259283. [CrossRef] [PubMed]
28. Mathew, M.P.; Mahesh, T.Y. Leaf-based disease detection in bell pepper plant using YOLO v5. *Signal Image Video Process.* **2022**, *16*, 841–847. [CrossRef]
29. Li, S.; Li, Y.; Li, Y.; Li, M.; Xu, X. YOLO-FIRI: Improved YOLOv5 for Infrared Image Object Detection. *IEEE Access* **2021**, *9*, 141861–141875. [CrossRef]
30. Yao, J.; Qi, J.; Zhang, J.; Shao, H.; Yang, J.; Li, X. A Real-Time Detection Algorithm for Kiwifruit Defects Based on YOLOv5. *Electronics* **2021**, *10*, 1711. [CrossRef]