electronics

Article

YG-SLAM: GPU-Accelerated RGBD-SLAM Using YOLOv5 in a Dynamic Environment

Yating Yu [1], Kai Zhu [2,*] and Wangshui Yu [1]

[1] School of Mechanical Engineering, Jiangsu University of Technology, Changzhou 213001, China; 2021655076@smail.jsut.edu.cn (Y.Y.); 2021655014@smail.jsut.edu.cn (W.Y.)
[2] School of Automobile and Traffic Engineering, Jiangsu University of Technology, Changzhou 213001, China
* Correspondence: fatkyo@jsut.edu.cn

Abstract: Traditional simultaneous localization and mapping (SLAM) performs well in a static environment; however, with the abrupt increase of dynamic points in dynamic environments, the algorithm is influenced by a lot of meaningless information, leading to low precision and poor robustness in pose estimation. To tackle this problem, a new visual SLAM algorithm of dynamic scenes named YG-SLAM is proposed, which creates an independent dynamic-object-detection thread and adds a dynamic-feature-point elimination step in the tracking thread. The YOLOv5 algorithm is introduced in the dynamic-object-detection thread for target recognition and deployed on the GPU to speed up image frame identification. The optic-flow approach employs an optic flow to monitor feature points and helps to remove the dynamic points in different dynamic objects based on the varying speeds of pixel movement. While combined with the antecedent information of object detection, the system can eliminate dynamic feature points under various conditions. Validation is conducted in both TUM and KITTI datasets, and the results illustrate that YG-SLAM can achieve a higher accuracy in dynamic indoor environments, with the maximum accuracy augmented from 0.277 m to 0.014 m. Meanwhile, YG-SLAM requires less processing time than other dynamic-scene SLAM algorithms, indicating its positioning priority in dynamic situations.

Keywords: visual SLAM; dynamic scenes; object detection; GPU acceleration; optic-flow method

check for updates

Citation: Yu, Y.; Zhu, K.; Yu, W. YG-SLAM: GPU-Accelerated RGBD-SLAM Using YOLOv5 in a Dynamic Environment. *Electronics* **2023**, *12*, 4377. https://doi.org/10.3390/electronics12204377

Academic Editors: Flavio Canavero and Beiwen Li

Received: 28 July 2023
Revised: 1 September 2023
Accepted: 15 October 2023
Published: 23 October 2023

1. Introduction

With its own sensors, simultaneous localization and mapping (SLAM) is capable of tackling both localization and map construction challenges. In the case where an environment map or the location of a robot is not available in advance, SLAM can be used to address a variety of applications [1]. Meanwhile, visual SLAM (VSLAM) that utilizes cameras as its major sensors has many advantages, such as cheaper hardware, direct object detection and tracking, and the provision of rich visual and semantic information [2], making it a hot topic in the field of robotics at present.

Over the years, VSLAM has attracted tremendous attention [3], and some extremely mature VSLAM algorithms have come into being, such as MonoSLAM [4], ORB-SLAM2 [5], ORB-SLAM3 [6], PTAM [7], RGB-D SLAM [8], and large-scale direct SLAM (LSD-SLAM) [9]. Among them, the ORB-SLAM3 [6] system proposed by Campos et al. operated robustly in real time in both large and small scenes and indoor and outdoor environments. Furthermore, it could support vision, vision-inertial guidance, and hybrid maps and can operate on monocular, binocular, and RGB-D cameras using pinhole or fisheye models. For all of these, the SLAM algorithms are predicated on a static environment or items moving at a very slow pace. However, in reality, the existence of dynamic items such as vehicles and animals is inevitable. Most of the SLAM algorithms employ feature points for localization, which are generally derived from locations with high textural information [5,7,10,11]. Therefore, when a moving object includes substantial textural information, the algorithms will collect a tremendous amount of wrong feature points, causing a low accuracy of the pose estimation, large trajectory errors, and even double shadows in the map. Thus, how to limit the interference of moving items and improve the positioning precision and robustness in dynamic situations is also a challenge in the field of VSLAM.

Proposed firstly by Smith, Self, and Cheeseman [12] in 1986, the SLAM technique has been applied for more than three decades with three main development stages. The first stage is the traditional phase, where the SLAM problem is proposed and converted into a state estimation problem. It mainly uses LiDAR as the sensor data source, which is usually tackled by an extended Kalman filter, particle filter, and maximum likelihood estimation. The drawback of this phase is that the convergence nature of the map is ignored, and the localization problem is treated separately from the map construction problem. The second stage is the algorithm analysis phase. A.J. Davison presented MonoSLAM [4] in 2007, which was treated as the first real-time monocular-vision SLAM system, but it could only perform localization and map building offline. Klein et al. [7] later suggested PTAM, which adopted nonlinear optimization as a backend to parallelize the tracking and map-building process. However, it is only applicable in small scenes and is prone to loss during tracking. MurArtal et al. [13] proposed ORB-SLAM in 2015 based on PTAM. It revolved around ORB features for pose estimation and overcame the cumulative error problem through a bag-of-words model, but there still existed frame loss during rotation. The third stage is the robustness and prediction phase. In the context of the comprehensive development of AI technology, deep learning has gradually been integrated into VSLAM to achieve robustness, high-level scene understanding, and computational resource optimization. Deep-learning-based VSLAM systems obtain semantic labels of features by applying sophisticated CNN structures such as YOLO [14], SSD [15], SegNet [16], and Mask R-CNN [17] when a new frame arrives. Bescos [18] et al. proposed a dynamic SLAM system designated Dyna-SLAM, which incorporated the Mask R-CNN instance segmentation algorithm and multiview geometry module into ORB-SLAM2 in order to eliminate dynamic feature points. In Detect-SLAM, Zhong et al. [19] employed a relatively fast SSD target detection model to identify image frames, and then propagated motion probability through feature matching to increase the influence area and eliminate the influence of interference factors in a dynamic environment. J. Vincent et al. [20] utilized case segmentation and multiview geometry to generate masks of dynamic objects in order to avoid the processing of these mask image regions during optimization, thereby minimizing the impact of these dynamic visual features on positioning and mapping. Yu et al. [21] enhanced ORB-SLAM2 and designed DS-SLAM, which adopted a SegNet neural network for semantic segmentation and filters out dynamic feature points based on point cloud motion analysis. Wu et al. [22] utilized an upgraded lightweight target detection network [23] to generate system semantic information and suggested a new geometric method to filter dynamic features in the detection region. Theodorou et al. [24] enhanced ORB-SLAM3 with the YOLOR object-detection network and optic-flow technology to detect and eliminate dynamic objects in key frames and validate them in customized train-station data sets, raising the accuracy by 89%. These algorithms combined with deep learning are able to filter the impact of dynamic factors and significantly improve localization accuracy compared to the ones in the first two stages, but they lack online performance due to great computing power consumption.

To address the dilemma of the poor real-time efficacy of VSLAM systems, graphics processing units (GPUs) are gradually entering the scope of research. Several research works have reported the performance results of the parallel execution of computer vision algorithms on CPUs and made some comparisons. Chaple et al. [25] compared the performance of image-convolution implementations on GPUs, FPGAs, and CPUs. Russo et al. [26] compared image-convolution processing on GPUs and FPGAs. References [27,28] addressed a K-means clustering method, a two-dimensional filter, and stereo-vision problems on quad-core CPUs and GPU processors, respectively. The investigation showed that all image pixels could be independently processed by GPUs, demonstrating their superiority.

To efficiently upgrade the positioning accuracy of the VSLAM algorithms in dynamic-indoor situations and to meet the real-time requirement, this paper refines the framework based on ORB-SLAM3 from the following aspects:

(1) An independent GPU-accelerated dynamic-object-detection thread is created. It could recognize and label different levels of dynamic objects in complex environments, thus enhancing the positioning precision and robustness of ORB-SLAM3 in dynamic situations. By deploying the YOLOv5 object-detection function on the GPU, the identification speed of image frames is significantly increased, thereby guaranteeing the system's online manifestation.

(2) In the tracking thread, a dynamic-feature-point-removal step is included. It incorporates the LK optic-flow algorithm and the dynamic prior information. Due to the differing instantaneous velocities of the pixels moving in static and dynamic objects, the optic-flow approach is employed to monitor and delete dynamic feature points. The object-detection prior information is utilized to categorize the discovered items, assist the system in eliminating potential dynamic factors, and retain feature points in static and some low-dynamic objects for positioning and environmental map construction.

(3) YG-SLAM is validated in the TUM RGB-D public dataset and further verified in KITTI dataset. Compared to ORB-SLAM3, it has obtained great enhancements in the accuracy and robustness over dynamic-environment sequences. It also provides a superior localization accuracy and greater speed in numerous sequences than other dynamic-scene SLAM algorithms.

The remaining portions are organized as follows: Section 2 introduces the system architecture, optic-flow method, object-detection module, and dynamic-point-elimination procedure. The results and numerical analysis of the dataset experiments are presented in Section 3. After discussing and concluding the paper, we detail its shortcomings as well as future studies.

2. Materials and Methods

2.1. System Architecture

Compared with the conventional ORB-SLAM3 framework, YG-SLAM is augmented with two modules: an object-detection module and a dynamic-feature-point-elimination one, as shown in Figure 1. At first, the RGB images and depth images acquired by the RGB-D camera are sent into the system, followed by the simultaneous extraction of feature points and GPU-based object detection. The identified object frame positions are transmitted to the tracking thread following object detection. Then, the optic-flow vectors of feature points are subsequently calculated via the optic-flow method. With the help of the prior information of the dynamic value, the extracted feature points are classified into diverse dynamic categories. Lastly, distinct strategies are implemented to classify feature points. High-dynamic feature points are eliminated directly, low-dynamic feature points are only retained in key frames, and static feature points are expressly retained. After the removal of dynamic points, the image frames are eventually output for positioning and mapping.

In YG-SLAM, object detection is implemented on the server's GPU to facilitate real-time operation. It detects and marks objects in the environment using the YOLOv5 algorithm. According to the prior information settings, the system divides the objects into three dynamic levels, as high-dynamic objects, low-dynamic objects, and static objects, and performs different operations on their feature points, retaining the useful ones for positioning and mapping. The LK optic-flow algorithm is adopted to monitor the extracted feature points in image frames. It can also be used to match the features of consecutive series of images, eliminating the need for descriptors in partial feature matching and lowering the likelihood of mismatching. When the quantity of tracked feature points falls below a certain threshold, they are extracted again. Meanwhile, optic flow can also be utilized to erase obvious dynamic feature points in distinct dynamic levels of moving objects with varying motion rates.

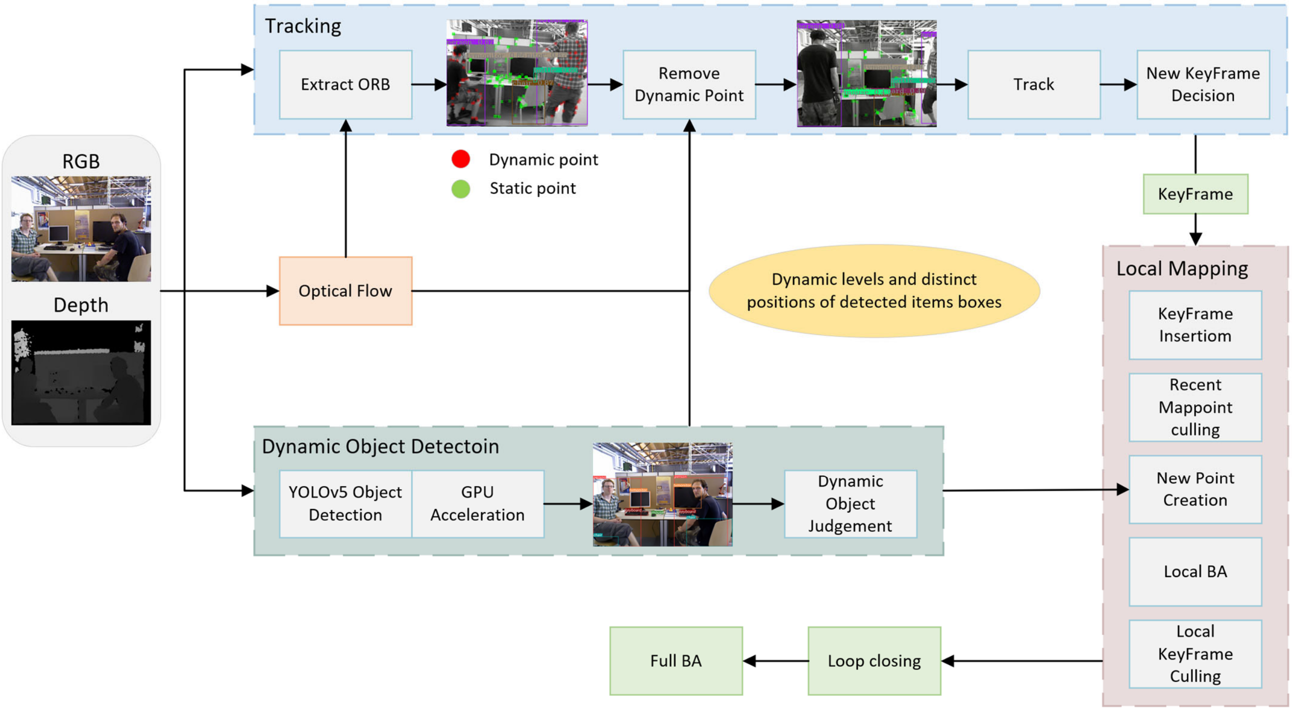


Figure 1. The system architecture of YG-SLAM.

2.2. The LK Optic Flow Approach

Optic flow is a concept in the motion detection of objects in the visual field, which was put forward by Stoffregen et al. [29] in 1987. Generally speaking, it is caused by the movement of the scene's foreground object, the movement of the camera, or the joint movement of both. It is often utilized to define the motion state of pixels in time-transformed image frames or in environments with instances of dynamic objects, so as to track pixels on them and determine their motion. According to the density of the number of tracking pixel motions, it could be classified into two types: dense optic-flow method and sparse optic-flow method, where the latter one is also known as LK optic flow [30]. ORB feature points have visible advantages in the extraction speed [31], and in ORB-SLAM3, an eight-layer image pyramid is built to maintain scale invariance after the extraction. In YG-SLAM, as illustrated in Figure 2, the extracted ORB feature points are used for LK optic-flow analysis. Three assumptions underpin the development of the LK optic flow method: gray-scale value invariance, motion constrains, and spatial consistency.

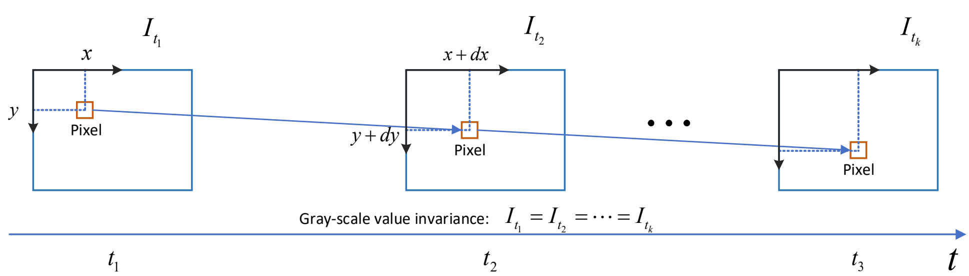


Figure 2. Optic flow simulation sketch.

In Figure 2, we let $I(x, y, t)$ be the gray-scale value of a pixel at (x, y) at time t_1, and relocate it to the position $(x + dx, y + dy)$ at time $t_2 = t_1 + dt$ when its gray-scale value is $I(x + dx, y + dy, t + dt)$. According to the assumption of gray-scale value invariance, we have:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) \tag{1}$$

By performing a Taylor expansion on the left side of the equation and using the assumption of gray-scale value invariance again, we can obtain:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} = -\frac{\partial I}{\partial t} \tag{2}$$

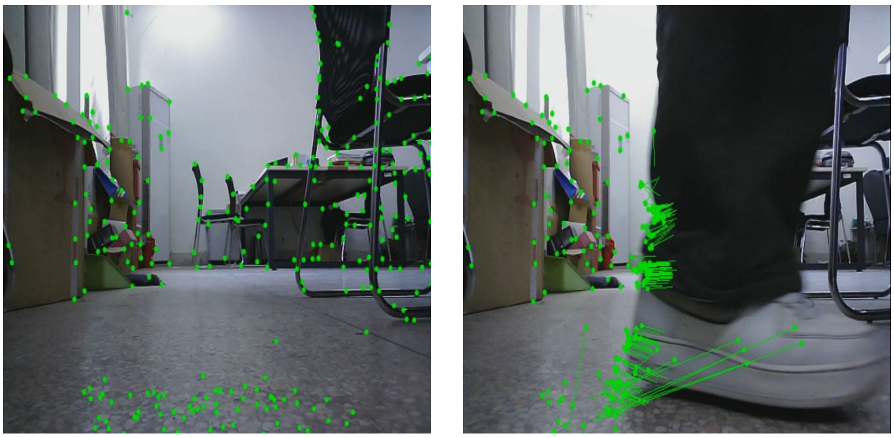
where dx/dt and dy/dt respectively denote the actual velocity decomposed to the x-axis and y-axis, marked as u, v. $\partial I/\partial x$ and $\partial I/\partial y$, set as I_x and I_y respectively, represent the gradient of the image frame in the direction of x and y. Then the partial derivative of the image with respect to time $\partial I/\partial t$ is set as I_t and performs matrix operations. Given that the binary first-order equation cannot be solved by a single pixel point, we set a $\omega \times \omega$ window in the LK optic-flow method and derive ω^2 equations based on the assumption that the motion mode of pixels in the agreement window is identical:

$$\begin{bmatrix} I_x & I_y \end{bmatrix}_k \begin{bmatrix} u \\ v \end{bmatrix} = -I_{tk}, k = 1, \ldots, w^2 \tag{3}$$

Through the preceding equation, we can procure:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_{x_i}^2 & \sum I_{x_i} I_{y_i} \\ \sum I_{x_i} I_{y_i} & \sum I^2 y_i \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_{x_i} I_{t_i} \\ -\sum I_{y_i} I_{t_i} \end{bmatrix} \tag{4}$$

To substantiate the reliability of the optic-flow method in real-world situations, the camera is placed in a real scene with a moving person. The optic-flow information in the scene is depicted in Figure 3.



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

Figure 3. Ambient optic flow. (**a**) Background optic flow; (**b**) Moving figures and background optic flow.

Figure 3a reveals that when the camera is positioned in a static environment, the static feature points within the scene exhibit a point-like distribution. And when the moving characters appear on the screen in Figure 3b, non-point optic-flow information is generated, which differs from the static environment. They are treated as dynamic feature points that can be cut out.

Therefore, the optic-flow method can provide additional motion data to assist the system to better cope with dynamic objects in the environment. However, when the person in Figure 3b remains stationary, the same optic-flow information as the background is displayed, which cannot be excluded, resulting in inaccurate positioning and map building. As a consequence, the optic-flow method is ineffective when people or other dynamic objects in real-world scenes fail to maintain motion. Hence, this paper combines the optic-flow method with object detection and the prior information for dynamic factor rejection, allowing the algorithm to accurately eradicate dynamic feature points and potential dynamic feature points under various conditions.

2.3. YOLOv5-Based Object Detection Module with Dynamic Feature Point Elimination

Since ORB-SLAM3 is incapable of handling dynamic objects and RANSAC [32] can only eliminate certain dynamic points from small objects, it is unable to manage scenes involving an excessive amount of moving targets. YG-SLAM is configured with an object-detection module based on YOLOv5, and the detection results are applied to the dynamic feature point removal stage alongside the optic-flow approach, making the system adaptive to the dynamic environment. The whole procedure of feature point processing for dynamic targets of different levels based on the object detection algorithm is depicted in Figure 4.

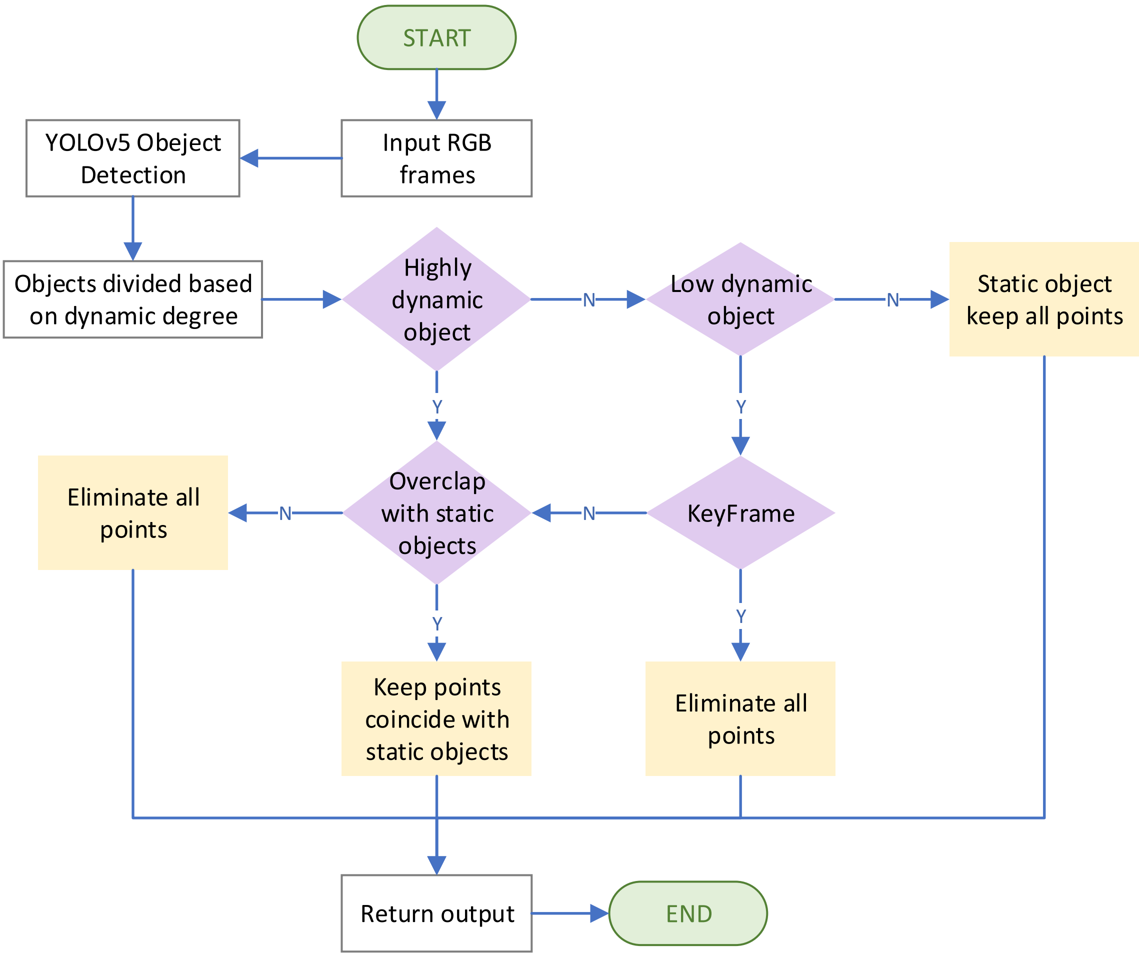
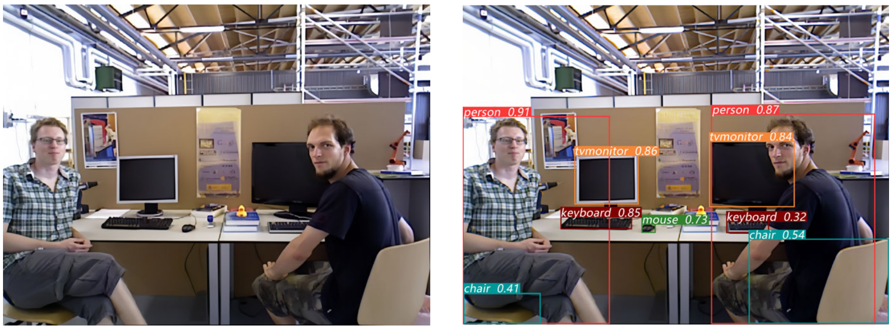


Figure 4. Flow chart of feature-point-retention strategies for objects of different dynamic levels.

Following the input of each RGB image frame, the detection frames of various objects are initially created using the YOLOv5 algorithm. Figure 5a depicts the incoming RGB image, whereas Figure 5b depicts the processed image; the differences demonstrate that YOLOv5 could draw detection frames for detecting different classes of objects and identify the object categories to which these objects belong.



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

Figure 5. Managed RGB images by YOLOv5 algorithm; (**a**) shows the initial RGB image, while (**b**) shows the processed image.

According to the classification results, YG-SLAM can obtain the object categories in the surrounding environment, but it fails to distinguish between static and dynamic objects. Therefore, this paper incorporates a prior judgment for dynamic objects to discriminate dynamic and static objects, and it introduces dynamic value m to describe the dynamic properties of objects. Based on the human prior judgments of things, the object is assigned an initial dynamic value m with a domain between (0, 1). All objects are classified into high-dynamic objects (m between 0.8 and 1), low-dynamic objects (m between 0.4 and 0.8), and static objects (m between 0 and 0.4). For example, tables and shelves are typically regarded as static objects that do not move frequently. Thus, the feature points can be considered static and saved. In most circumstances, chairs, mice, keyboards, and other low-dynamic objects require feature points to be preserved for tracking. People and vehicles are high-dynamic objects that should be removed at first. Table 1 gives the dynamic initial values of common objects in the room.

Table 1. Dynamic initial values of common objects.

Dynamic Levels	Detection Results	Dynamic Value
High-dynamic objects	Person	1
Low-dynamic objects	Cup	0.7
	TV monitor	0.5
	Mobile Phone	0.7
	Keyboard	0.7
	Mouse	0.7
	Chair	0.6
Static objects	Desk	0.2
	Air conditioner	0.1
	Bookshelf	0.1

After object detection and prior information processing, YG-SLAM can differentiate between static and dynamic points. Unquestionably, the feature points for extremely dynamic objects are disregarded. For low-dynamic objects, only the dynamic points in key frames are removed, whereas non-key frames retain this component. Static objects are equivalent to environment-valid objects, so the system would keep all of their feature points for positioning and mapping.

The objects detected using YOLOv5 are contained within regular rectangular regions. Removing all information directly from the detected object frame results in the improper removal of numerous static points, thereby drastically reducing the number of available feature points. Even when moving objects cover a large area, it can lead to more system mismatching problems in both positioning and mapping. Consequently, in accordance with the classification rules in Table 1, the overlapping portion of the detection box is first confirmed. It is judged whether the static object field is included within the dynamic object box. If positive, the characteristic points associated with the static object field are retained, while any remaining characteristic points in the dynamic target frame are deleted.

Figure 6a,b indicate the processed image after the extraction of feature points and subsequent removal of dynamic feature points, respectively. As shown in Figure 6b, the points of the covering part of the high-dynamic object "person" and the low-dynamic objects "tvmonitor", "chair", "keyboard", and "mouse" are retained. In this diagram, all of the red spots on the dynamic objects have been removed.

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

Figure 6. Using the YG-SLAM algorithm, targeted removal of dynamic feature points was performed: (**a**) shows the original image including all feature points; (**b**) shows the image without the dynamic feature points. The red points correspond to dynamic feature points, whereas the green points correspond to static feature points.

The combination of the YOLOv5 object-detection algorithm and the LK optic-flow approach enables the rejection of both absolutely and potentially moving objects under various dynamic circumstances. Matching with only using the LK optic-flow method for velocity calculation, object detection based on the prior information could lessen a great deal of optic-flow information calculations. At the same time, the algorithm using object-detection results instead of semantic segmentation results could effectively guarantee the real-time operation of the SLAM system.

3. Experiment and Analysis

3.1. Experiment Preparation

To evaluate the performance of YG-SLAM, the TUM dataset is primarily adopted, and quantitative tests for comparison with existing main-stream SLAM algorithms are performed. A Kinect RGB-D camera with a resolution of 640×480 is utilized to record information at a rate of 30 Hz in the TUM dataset created by the University of Munich. The dataset supplies real-time camera location and pose information that can be used to approximate the RGB-D camera's true position data. This research focuses on experiments with five sequences from the previously mentioned dataset: four walking sequences and one sitting sequence, respectively. Meanwhile, a sequence of KITTI dataset is selected for further verification. The comparison systems are ORB-SLAM3, DS-SLAM, DetectSLAM, RDS-SLAM [33], and DynaSLAM.

In this experiment, the software environment is Ubuntu 18.04, and the hardware is 8-core AMD Ryzen 7 5800X CPU, GeForce RTX 3060Ti graphics card, 8 GB video memory, and 32 GB RAM.

3.2. Ablation Study

To assess the contribution of new modules to the algorithm's efficiency, a comparative ablation experiment is conducted under different conditions. As shown in Table 2, "$\surd$" indicates the module is applied in ORB-SLAM3 while "$\times$" means it is not. The root-meansquare error (RMSE) and standard deviation (Std) of the absolute trajectory error (ATE) are selected to compare the algorithm accuracy.

Table 2. Ablation study based on ORB-SLAM3.

Module			APE/m		Improvement/%	
YOLOv5 (GPU)	**Optic Flow**	**Dynamic Value *m***	**RMSE**	**Std**	**RMSE**	**Std**
×	×	×	0.2769	0.1338	0	0
√	×	×	0.0266	0.0132	90.39	90.13
×	√	×	0.1452	0.0995	47.56	25.64
√	×	√	0.0186	0.0083	93.28	93.80
√	√	√	0.0139	0.0071	94.99	94.68

The results reveal that the original method is the least competitive. With the YOLOv5 object-detection algorithm, the accuracy is significantly improved in a dynamic environment. However, because all the feature points in the detected dynamic objects are inevitably eliminated, there are fewer available points for positioning and mapping, so the accuracy improvement effect does not reach the optimal level. The single optic-flow module improves the accuracy by 47.56%. It can assist the system in gathering more motion data, thereby aiding it in overcoming some challenges in a dynamic environment, but it fails to capture potential dynamic points. The addition of dynamic-value prior information exerted on the object-detection algorithm raises the accuracy by 93.28%, while the completed framework refines the accuracy by 94.99%, indicating that the experimental effect will be optimal when all these modules are available simultaneously.

3.3. Comparative Experimental Data and Analysis with ORB-SLAM3

Since YG-SLAM comes from ORB-SLAM3, they are compared at first. The ATE and relative pose error (RPE) are utilized in this paper to assess the accuracy of SLAM. ATE, which indicates the explicit distinction between the pose estimation and the ground truth, appears to reflect the algorithm's correctness and the uniformity of the global trajectory. RPE reveals the comparative translation error and relative rotation error measured by the odometer, which mainly calculates the error magnitude of the relative motion between each pair of trajectory points. For comparison, this study selects three indexes: RMSE, mean error (mean), and Std of ATE and RPE [34].

According to the data presented in Tables 3 and 4, YG-SLAM outperforms ORB-SLAM3 in dynamic scenarios. The accuracy of YG-SLAM is higher for sequences with movable objects and a greater extent of motion inside the image, and it achieves a significant reduction of up to 94% in the ATE. For relatively static scenes, YG-SLAM also provides updates.

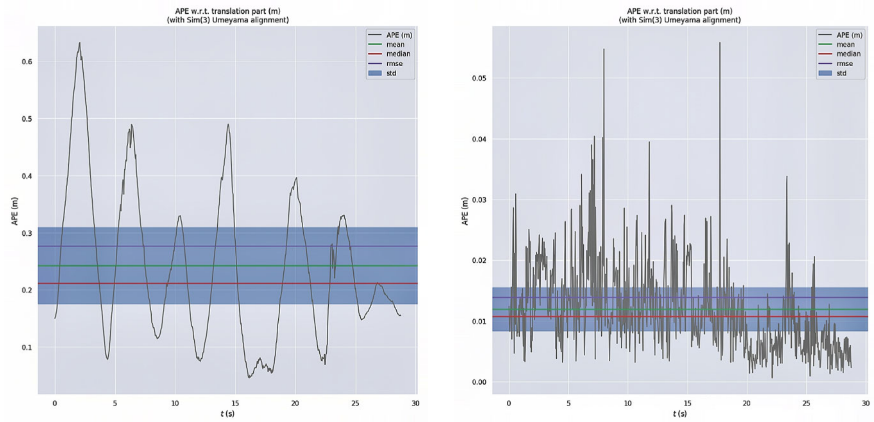
Table 3. ATE comparison in TUM sequences between YG-SLAM and ORB-SLAM3.

Sequences	ORB-SLAM3/m			YG-SLAM/m			Improvement		
	RMSE	**Mean**	**Std**	**RMSE**	**Mean**	**Std**	**RMSE**	**Mean**	**Std**
walking_xyz	0.2769	0.2424	0.1338	0.0139	0.0119	0.0071	94.99	95.09	94.68
walking_half	0.2225	0.2091	0.0760	0.0193	0.0169	0.0092	91.34	91.90	87.93
walking_static	0.0244	0.0212	0.0121	0.0081	0.0072	0.0036	66.84	65.84	70.13
walking_rpy	0.1506	0.1311	0.0742	0.0860	0.0756	0.0409	42.91	42.31	44.82
sitting_half	0.0708	0.0676	0.0210	0.0550	0.0340	0.0132	22.34	49.65	37.32

Table 4. RPE comparison in TUM sequences between YG-SLAM and ORB-SLAM3.

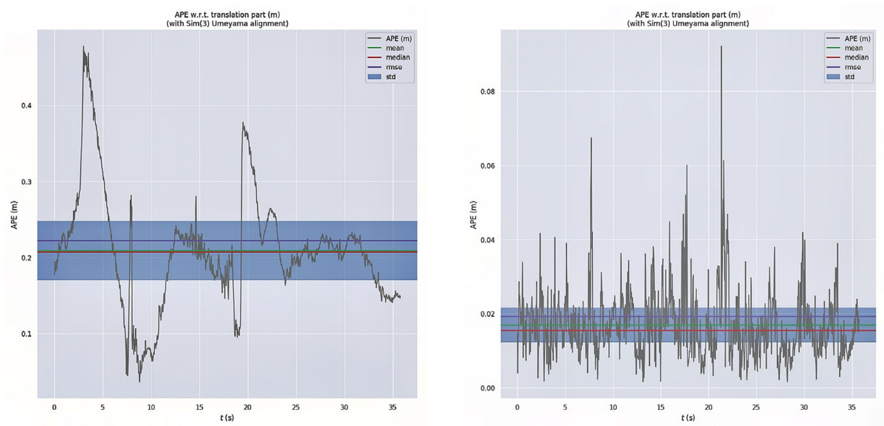
Sequences	ORB-SLAM3/m			YG-SLAM/m			Improvement/%		
	RMSE	Mean	Std	RMSE	Mean	Std	RMSE	Mean	Std
walking_xyz	0.0122	0.0101	0.0069	0.0114	0.0091	0.0067	6.81	10.51	2.57
walking_half	0.0162	0.0117	0.0112	0.0144	0.0114	0.0088	11.19	2.90	21.30
walking_static	0.0163	0.0148	0.0177	0.0009	0.0008	0.0005	96.16	94.87	97.41
walking_rpy	0.0061	0.0051	0.0035	0.0055	0.0042	0.0033	9.58	16.42	4.37
sitting_half	0.0137	0.0106	0.0087	0.0079	0.0065	0.0045	42.23	38.06	48.92

Figures 7 and 8 depict the distributions of ATE for ORB-SLAM3 and YG-SLAM in the walking_xyz and walking_half sequences. Specifically, Figures 7a and 8a illustrate the ATE of ORB-SLAM3 in the walking_xyz and walking_half sequences, while Figures 7b and 8b reflect the ATE of YG-SLAM.



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

Figure 7. Walking_xyz absolute trajectory error distribution graphs: (**a**) is the ATE of ORB-SLAM3 walking_xyz; (**b**) is the ATE of YG-SLAM walking_xyz.

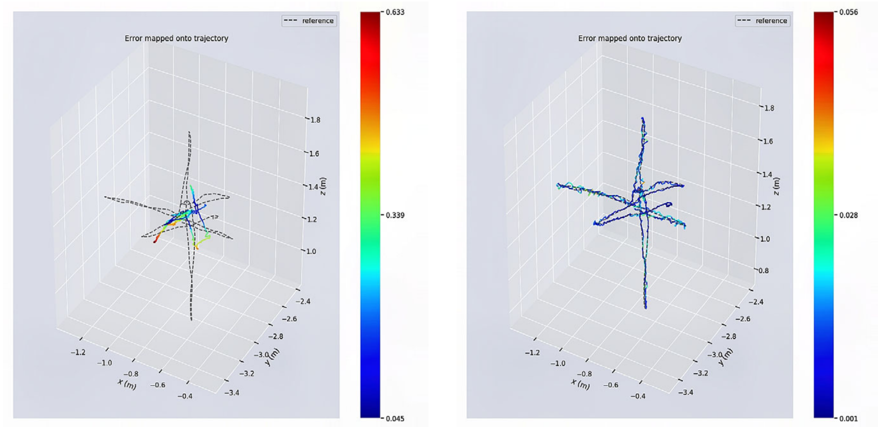


(**a**) (**b**)

Figure 8. Walking_half absolute trajectory error distribution graphs: (**a**) is the ATE of ORB-SLAM3 walking_half; and (**b**) is the ATE of YG-SLAM walking_half.

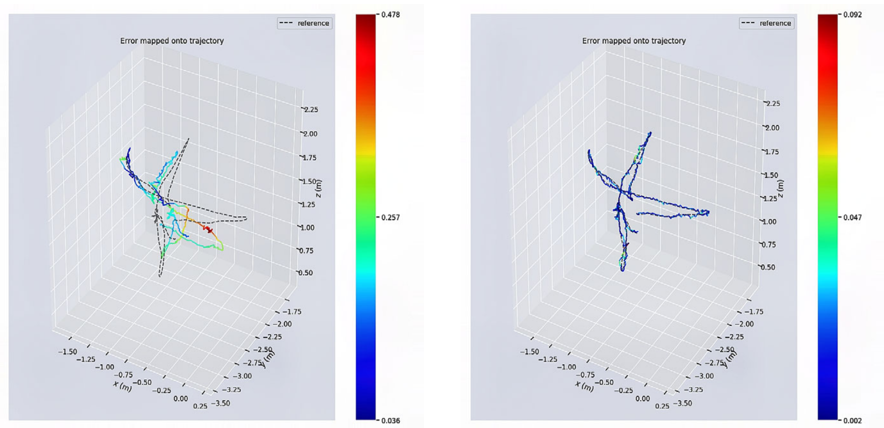
According to Figures 7 and 8, the dispersion coefficient of ORB-SLAM3 surpasses that of YG-SLAM and similarly, the maximum error of ORB-SLAM3 is also vaster. These findings suggest that YG-SLAM exhibits a significant reduction in ATE, leading to a notable improvement in positioning accuracy.

Figures 9 and 10 show the ATE plots of ORB-SLAM3 and YG-SLAM in 3D space for the walking_half and walking_xyz sequences, respectively. These figures provide a clear representation of the ATE distribution for both YG-SLAM and ORB-SLAM3. The presented image with a gray dashed line denotes the ground-truth data, while the colored line represents the error mapped onto the trajectory.



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

Figure 9. ATE graphs for the walking_xyz sequence in 3D space: (**a**) is the ATE_map of ORB-SLAM3; (**b**) is the ATE_map of YG-SLAM.

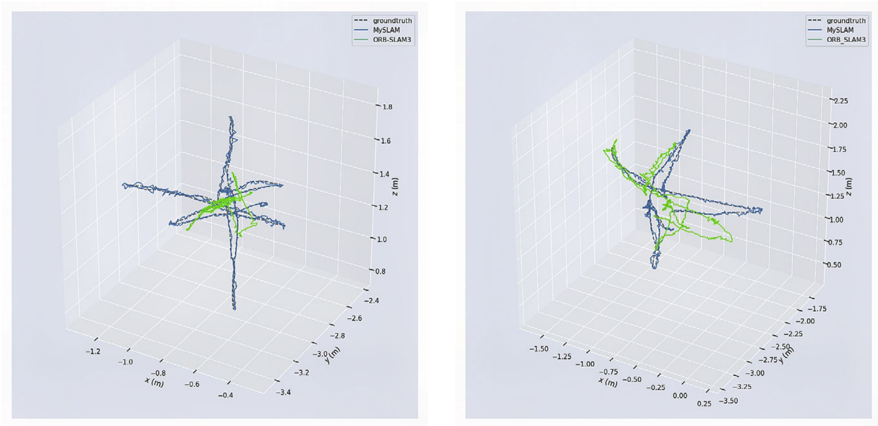


(**a**) (**b**)

Figure 10. ATE graphs for the walking_half sequence in 3D space: (**a**) is the ATE of ORB-SLAM3; (**b**) is the ATE of YG-SLAM.

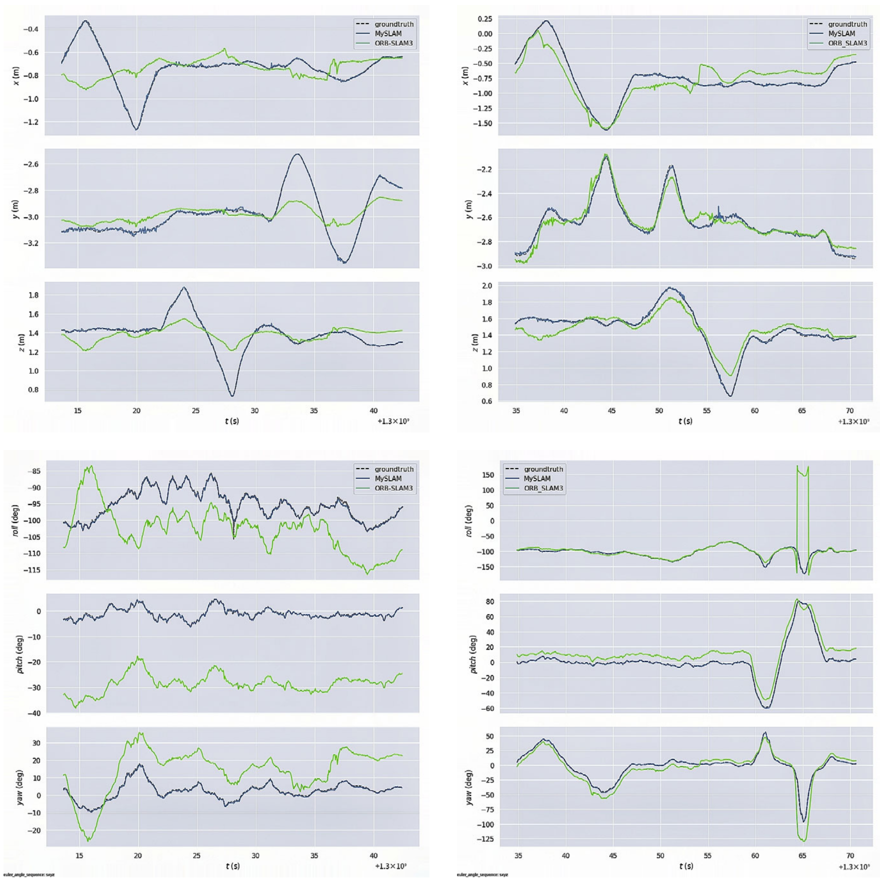
The graphical representation provides evident indication of a significant disparity between the color curve in ORB-SLAM3 and the ground-truth trajectory. This discrepancy is accompanied by a high level of inaccuracy, meaning that the positioning accuracy of ORB-SLAM3 is suboptimal when applied to dynamic sequences. The performance of YG-SLAM is satisfactory, as seen from the near alignment of its color curve with the ground-truth trajectory. Figures 9a and 10a depict the ATE map of ORB-SLAM3, whereas Figures 9b and 10b illustrate the ATE map of YG-SLAM.

For further analysis, we undertake comparisons between the ground-truth trajectory and the calculated trajectory of ORB-SLAM3 and YG-SLAM. Figure 11 displays a comparison graph illustrating three distinct 3D trajectories, and Figure 12 exhibits a comparison graph of three trajectories in terms of their xyz and rpy coordinates. It is notable that Figures 11a and 12a belong to the walking_xyz series, while Figures 11b and 12b correspond to the walking_half sequence. The gray dashed line in the figure represents the ground-truth trajectory, the green line represents the ORB-SLAM3 calculated trajectory, and the blue line represents the YG-SLAM estimated trajectory.



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

Figure 11. Comparative graphs showing three distinct 3D trajectories: (**a**) is the walking_xyz sequence and (**b**) is the walking_half sequence. The gray dashed lines in the figures indicate the ground truth, the green lines are the ORB-SLAM3 calculated trajectories, and the blue lines are the YG-SLAM calculated trajectories.

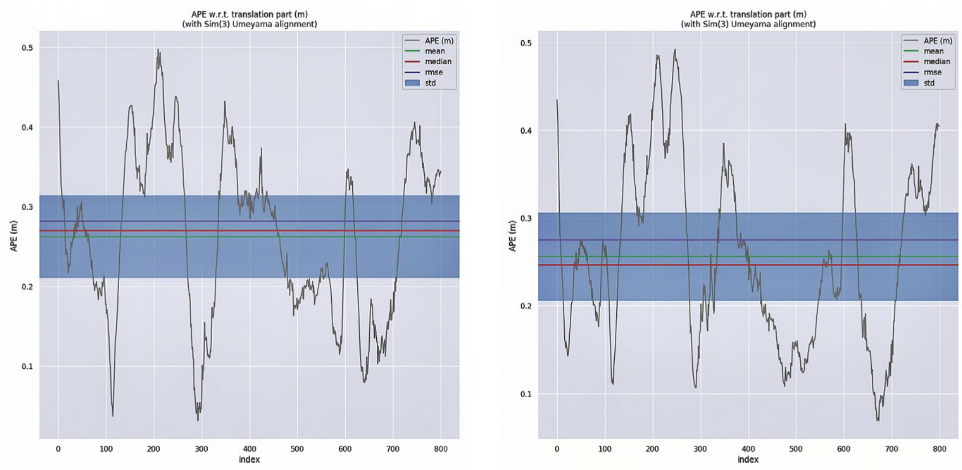


(**a**) (**b**)

Figure 12. Comparison graphs showing three trajectories in the xyz and rpy directions: (**a**) represents the walking_xyz sequence and (**b**) represents the walking_half sequence. The gray dashed lines in the figures indicate the ground truth, the green lines are the ORB-SLAM3 calculated trajectories, and the blue lines are the YG-SLAM estimated trajectories.

Based on the observations made in Figures 11 and 12, the blue line exhibits a closer proximity to the ground truth in comparison to the green line. It suggests that the inclusion of the dynamic feature point removal approach has resulted in enhanced accuracy in pose estimation and localization for YG-SLAM.

To enhance the credibility of our findings, the KITTI outdoor dataset, which is primarily used in the field of autonomous driving research and comprises authentic image data collected from many environments, including urban areas, rural settings, and highways, is selected for supplementary verification. This paper chooses the rural road scene and runs it on ORB-SLAM3 and YG-SLAM, respectively. The ATE of YG-SLAM is 0.2772 m and the ATE of ORB-SLAM3 is 0.2837 m; both errors are on the decimeter scale, while the algorithms exhibit errors in the centimeter range when applied to indoor datasets. Therefore, it is evident that the ORB-SLAM3 algorithm and YG-SLAM algorithm are well-suited for inside environments when only a vision sensor is utilized. Figure 13 depicts the absolute pose differences between the two algorithms.



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

Figure 13. KITTI-03 absolute trajectory error distribution graphs: (**a**) is the ATE of ORB-SLAM3; (**b**) is the ATE of YG-SLAM.

As shown in Figure 13, there are disparities between the two algorithms at the x-coordinate index of 100. Here, YG-SLAM fluctuates less, and the discrepancy between the representation and the actual trajectory is smaller. This is because a moving vehicle is present in the dataset here, and YG-SLAM can analyze these image frames to obtain a more precise positioning. However, there exist no dynamic objects in other sections of the dataset, the performance is comparable to the original algorithm, and the positioning accuracy is improved by 3% overall. Hence, the proposed algorithm is also superior to ORB-SLAM3 when dynamic factors occur in other datasets.

In summary, YG-SLAM is more competitive than ORB-SLAM3 in terms of positioning estimation accuracy in dynamic scenarios. The calculated trajectory of ORB-SLAM3 differs substantially in the face of dynamic sceneries from the ground truth, whereas YG-SLAM can estimate the trajectory quite effectively.

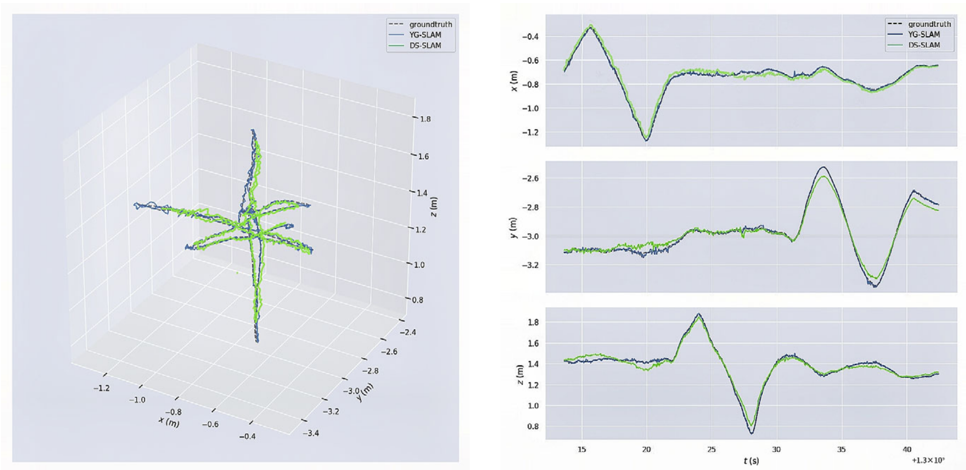
3.4. Comparative Experimental Data and Analysis with Other Dynamic SLAM Algorithms

To further validate YG-SLAM's superiority, comparative tests are conducted with DS-SLAM and Detect-SLAM, which are two other SLAM methods commonly employed in dynamic situations. The RMSE and Std values of ATE are picked as comparative metrics for verification in these tests, and the results are reported in Table 5.

Table 5. ATE evaluation of YG-SLAM and other dynamic-scene SLAM algorithms. The bold data represents better outcomes.

Sequences	DS-SLAM/m		Detect-SLAM/m		YG-SLAM/m	
	RMSE	**Std**	**RMSE**	**Std**	**RMSE**	**Std**
walking_xyz	0.0218	0.0127	0.0198	0.0077	**0.0139**	**0.0071**
walking_half	0.0271	0.0113	0.0473	0.0096	**0.0193**	**0.0092**
walking_static	0.0092	0.0046	0.0162	0.0036	**0.0081**	**0.0036**
walking_rpy	0.4169	0.2564	0.2903	0.2277	**0.0860**	**0.0409**
sitting_half	**0.0252**	**0.0029**	0.1140	0.1076	0.0550	0.0132

Figure 14 displays a comparative analysis between the walking_xyz trajectories of DS-SLAM and YG-SLAM, in relation to the ground-truth trajectory both in 3D space and along the xyz direction. The graphic illustrates the trajectory of DS-SLAM with the green line, and the trajectory of YG-SLAM with the blue line. The closer proximity of the blue line to the ground truth indicates that YG-SLAM is more accurate in walking_xyz.

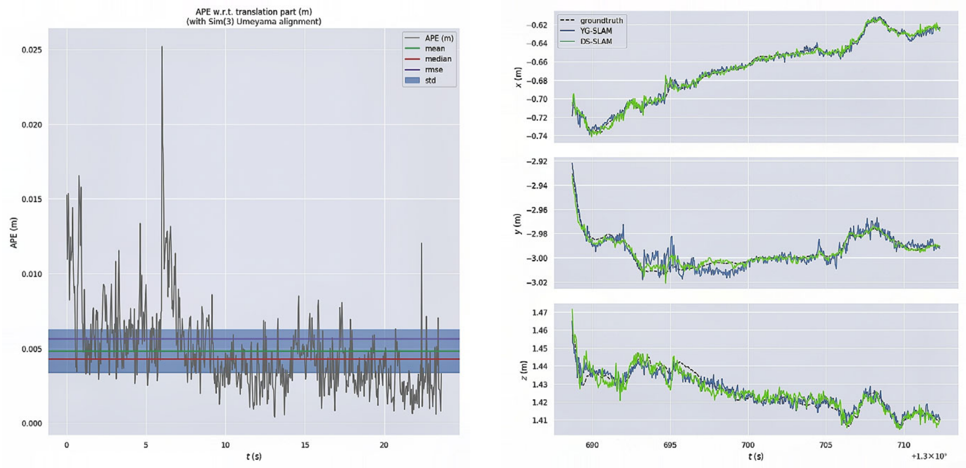


(**a**) (**b**)

Figure 14. Comparative graphs showing three trajectories in walking_xyz: (**a**) is in the 3D space and (**b**) is in the xyz direction.

Figure 15a illustrates the ATE error distribution of DS-SLAM in the sitting_half sequence, whereas Figure 15b reflects a comparison between DS-SLAM and YG-SLAM tracks in the xyz direction. In this particular instance, the amplitude of fluctuation in the trajectory of the green line is comparatively smaller and more closely aligned with the ground-truth trajectory. This observation serves as proof that DS-SLAM exhibits superior performance in the sitting_half sequence.

In conclusion, YG-SLAM provides a superior performance than DS-SLAM and Detect-SLAM in sequences characterized by significant dynamic object motion amplitudes. However, the power of DS-SLAM grows when a smaller number of dynamic target movements is detected. The reason is that YG-SLAM promptly eliminates all dynamic feature points without considering the possibility that feature points on dynamic targets may remain static, causing a few static points to be retained, and it lowers the accuracy of tracking and placement.



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

Figure 15. The experimental graphs of the sitting_half sequence: (**a**) is the absolute trajectory error distribution graph; (**b**) is the comparison graph showing three trajectories in the xyz direction.

3.5. Real-Time Performance Analysis

Real-time performance is another significant SLAM evaluation metric. Once SLAM prioritizes accuracy over real-time performance, it will be difficult to implement in the actual world. The time cost to manage each frame under assorted sequences is compared to validate the real-time performance of DS-SLAM, RDS-SLAM, Detect-SLAM, DynaSLAM, ORB-SLAM3, and YG-SLAM. Table 6 displays the experimental outcomes.

Table 6. Real-time performance comparison of YG-SLAM and other dynamic-scene SLAM algorithms.

Sequences	DS-SLAM	RDS-SLAM	Detect-SLAM	DynaSLAM	ORB-SLAM3	YG-SLAM
walking_xyz	0.2804	0.0752	0.1230	0.4133	0.0137	0.0251
walking_half	0.2530	0.0785	0.1278	0.4274	0.0143	0.0249
walking_static	0.2637	0.0774	0.1215	0.4189	0.0162	0.0254
walking_rpy	0.2528	0.075	0.1234	0.4131	0.0137	0.0240

Based on the findings presented in Table 6, YG-SLAM exhibits better real-time results than other dynamic-scenario SLAM algorithms. Additionally, the utilization of GPU technology considerably boosts the runtime of the object detection thread. Hence, despite the increased object detection process, it takes more time than ORB-SLAM3 but still performs at a relatively satisfactory level in terms of speed.

4. Discussion

The findings of our study demonstrate that the utilization of GPU-based object detection and the optic-flow approach effectively minimizes errors in SLAM positioning under dynamic scenarios. The LK optic-flow method is effective for tracking and recognizing the obvious dynamic points. Additionally, the GPU-based object detection coupled with prior information is proven to be efficient in reducing numerous degrees of dynamic feature points, while also meeting real-time processing requirements. Under the TUM sequences of tests, the cost time of each frame could remain within 3 ms, and the absolute trajectory errors could be reduced by up to 94%. However, we can facilitate the works from the following aspects:

(1) Given that SLAM technology is predominantly utilized in indoor robots, it is imperative for the algorithm to streamline the network structure to enhance its versatility and facilitate its implementation in diverse circumstances.

(2) The concept of generating 3D semantic maps to enable route planning in dynamic situations might be considered based on the existing semantic map construction approach.
(3) In previous research, YG-SLAM achieved favorable positioning outcomes in indoor environments. However, given the considerable dropout in the accuracy of positioning systems that rely on depth cameras in outdoor or dim conditions, it may be beneficial to add other sensors. This would enhance the capability of these systems to address the mapping and positioning challenges encountered in various circumstances.

5. Conclusions

To refine the localization accuracy of VSLAM, this paper creates the YG-SLAM method, a real-time and highly accurate system that uses an optimized deep learning algorithm to detect objects in dynamic scenes. This new method is augmented with an independent object-detection module and a dynamic-feature-point-elimination step in the tracking thread with the employment of YOLOv5 algorithm, optic-flow method, and dynamic-value prior information based on the traditional ORB-SLAM3. Numerous experiments are carried out in order to evaluate and compare the system's performance. We could draw the following conclusion:

(1) YG-SLAM is effective for reducing the positioning errors in highly dynamic environments;
(2) Compared with the original ORB-SLAM3 method, YG-SLAM improves the localization accuracy in multiple sequences by more than 40% and up to 94%. Comparison with other SLAM algorithms in dynamic environments demonstrates that YG-SLAM performs better under circumstances with more dynamic objects and a wider range of object motion;
(3) This system utilizes GPU to accelerate YOLOv5 to meet the real-time performance demand. Compared to other dynamic systems, YG-SLAM is significantly faster than DS-SLAM, RDS-SLAM, and DynaSLAM, and even faster than DetectSLAM that uses a similar object detection method, indicating its real-time performance.

Author Contributions: Methodology, Y.Y.; writing—original draft, Y.Y.; writing—review and editing, K.Z.; resources, W.Y.; funding acquisition, K.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (19KJB620001).

Data Availability Statement: The research data used in this paper are from https://vision.in.tum.de/data/datasets/rgbd-dataset/download (accessed on 21 May 2023) and https://www.cvlibs.net/datasets/kitti/ (accessed on 26 August 2023).

Acknowledgments: The authors are grateful to the corresponding author Kai Zhu, the editors, and the anonymous reviewers for their insightful comments and suggestions.

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

References

1. Ali, T.; Hriday, B.; Luis, S.J.; Holger, V. Visual SLAM: What Are the Current Trends and What to Expect? *Sensors* **2022**, *22*, 9297.
2. Filipenko, M.; Afanasyev, I. Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment. In Proceedings of the 2018 International Conference on Intelligent Systems (IS), Funchal, Portugal, 25–27 September 2018; pp. 400–407.
3. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
4. Davison, A.J.; Reid, I.D.; Molton, N.D.; Stasse, O. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1052–1067. [CrossRef]
5. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
6. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.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]

7. Klein, G.; Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 225–234.
8. Kerl, C.; Sturm, J.; Cremers, D. Dense visual SLAM for RGB-D cameras. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 2100–2106.
9. Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-Scale Direct Monocular SLAM. In Proceedings of the Computer Vision—ECCV 2014, Zurich, Switzerland, 6–12 September 2014; pp. 834–849.
10. 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]
11. Li, S.; Lee, D. RGB-D SLAM in Dynamic Environments Using Static Point Weighting. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2263–2270. [CrossRef]
12. Smith, R.; Self, M.; Cheeseman, P. Estimating uncertain spatial relationships in robotics. In Proceedings of the 1987 IEEE International Conference on Robotics and Automation, Raleigh, NC, USA, 31 March–3 April 1987; p. 850.
13. 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]
14. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 Computer Vision & Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
15. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
16. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef]
17. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
18. Bescós, B.; Fácil, J.M.; Civera, J.; Neira, J. DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4076–4083. [CrossRef]
19. Zhong, F.; Wang, S.; Zhang, Z.; Chen, C.; Wang, Y. Detect-SLAM: Making Object Detection and SLAM Mutually Beneficial. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018.
20. Vincent, J.; Labbé, M.; Lauzon, J.S.; Grondin, F.; Comtois-Rivet, P.M.; Michaud, F. Dynamic Object Tracking and Masking for Visual SLAM. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 4974–4979.
21. 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.
22. 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]
23. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
24. Theodorou, C.; Velisavljevic, V.; Dyo, V. Visual SLAM for Dynamic Environments Based on Object Detection and Optical Flow for Dynamic Object Removal. *Sensors* **2022**, *22*, 7553. [CrossRef]
25. Chaple, G.; Daruwala, R.D. Design of Sobel operator based image edge detection algorithm on FPGA. In Proceedings of the 2014 International Conference on Communication and Signal Processing, Melmaruvathur, India, 3–5 April 2014; pp. 788–792.
26. Russo, L.M.; Pedrino, E.C.; Kato, E.; Roda, V.O. Image convolution processing: A GPU versus FPGA comparison. In Proceedings of the 2012 VIII Southern Conference on Programmable Logic, Bento Gonçalves, Brazil, 20–23 March 2012.
27. Saegusa, T.; Maruyama, T.; Yamaguchi, Y. How fast is an FPGA in image processing? In Proceedings of the 2008 International Conference on Field Programmable Logic and Applications, Heidelberg, Germany, 8–10 September 2008; Volume 108, pp. 83–88.
28. Asano, S.; Maruyama, T.; Yamaguchi, Y. Performance comparison of FPGA, GPU and CPU in image processing. In Proceedings of the 2009 International Conference on Field Programmable Logic & Applications, Prague, Czech Republic, 31 August–2 September 2009.
29. Stoffregen, T.A.; Schmuckler, M.A.; Gibson, E.J. Use of central and peripheral optical flow in stance and locomotion in young walkers. *Perception* **1987**, *16*, 113–119. [CrossRef]
30. Lucas, B.D.; Kanade, T. An Iterative Image Registration Technique with an Application toStereo Vision. In Proceedings of the 1997 International Joint Conference on ArtificialIntelligence, Nagoya, Japan, 23–29 August 1997.
31. Yang, G.; Chang, X.; Jiang, Z. A Fast Aerial Images Mosaic Method Based on ORB Feature and Homography Matrix. In Proceedings of the 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), Beijing, China, 28–31 August 2019.
32. Fischler, M.A.; Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In *Readings in Computer Vision*; Fischler, M.A., Firschein, O., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 1987; pp. 726–740.

33. Liu, Y.; Miura, J. RDS-SLAM: Real-Time Dynamic SLAM Using Semantic Segmentation Methods. *IEEE Access* **2021**, *9*, 23772–23785. [CrossRef]
34. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012.