Journal of Real-Time Image Processing (2024) 21:169
https://doi.org/10.1007/s11554-024-01527-4

RESEARCH

# A real-time visual SLAM based on semantic information and geometric information in dynamic environment

Hongli Sun[1] · Qingwu Fan[1] · Huiqing Zhang[1] · Jiajing Liu[1]

Received: 15 May 2024 / Accepted: 24 July 2024

**Abstract**
Simultaneous Localization and Mapping (SLAM) is the core technology enabling mobile robots to autonomously explore and perceive the environment. However, dynamic objects in the scene significantly impact the accuracy and robustness of visual SLAM systems, limiting its applicability in real-world scenarios. Hence, we propose a real-time RGB-D visual SLAM algorithm designed for indoor dynamic scenes. Our approach includes a parallel lightweight object detection thread, which leverages the YOLOv7-tiny network to detect potential moving objects and generate 2D semantic information. Subsequently, a novel dynamic feature removal strategy is introduced in the tracking thread. This strategy integrates semantic information, geometric constraints, and feature point depth-based RANSAC to effectively mitigate the influence of dynamic features. To evaluate the effectiveness of the proposed algorithms, we conducted comparative experiments using other state-of-the-art algorithms on the TUM RGB-D dataset and Bonn RGB-D dataset, as well as in real-world dynamic scenes. The results demonstrate that the algorithm maintains excellent accuracy and robustness in dynamic environments, while also exhibiting impressive real-time performance.

**Keywords** Dynamic environment · Object detection · Geometric constraint · Depth information · Visual SLAM

## 1 Introduction

Simultaneous localization and mapping (SLAM) is a pivotal technology in robotics and serves as the foundation for robots to autonomously navigate, explore, and perceive their surroundings [1]. According to the sensor, SLAM is mainly divided into laser-based SLAM, visual-based SLAM, and multi-sensor fusion SLAM. Visual SLAM has attracted widespread attention from researchers due to its compact size, cost-effectiveness, and the rich environmental information it can acquire. With the relentless efforts of scientific researchers, visual SLAM has developed rapidly. There have been numerous exemplary systems that have proposed in recent years, such as ORB-SLAM2 [2], DS-SLAM [3], DynaSLAM [4], etc. Its applications span across autonomous driving, medical service robots, smart agriculture, augmented reality (AR) [5–7], and more.

Although visual SLAM has achieved promising results, it also encounters a significant challenge: dynamic environment. Many existing visual SLAM systems are built on the assumption of static scenes, which severely restricts their deployment and usability in dynamic environments [4]. For instance, the presence of dynamic objects such as moving people and animals can negatively affect pose estimation and map reconstruction, thereby undermining the accuracy and stability of the system.

To ensure the stability of the visual SLAM system in dynamic environment, the traditional visual SLAM utilizes geometric constraints to directly eliminate dynamic features but is susceptible to noise and requires manual threshold adjustments [5]. The emerging semantic visual SLAM combines geometric constraints with deep learning technologies, such as semantic segmentation and object detection, effectively improving the robustness of visual SLAM in dynamic environments [1]. For instance, semantic segmentation networks such as SegNet [8] and PSPNet [9], along with object detection networks like MobileNetV3 [10] and YOLOv7 [11], can identify moving objects. Integrating these technologies with geometric data ensures the accuracy of visual

✉ Qingwu Fan
fqw@bjut.edu.cn

[1] College of Information Science and Technology, Beijing University of Technology, 100124 Beijing, China

SLAM system, but it also inevitably introduces real-time performance challenges.

Addressing the impact of dynamic objects on visual SLAM is, therefore, crucial for enabling accurate positioning, mapping, and intelligent applications of robots in real-world environments. This paper proposes a real-time RGB-D visual SLAM system designed for dynamic indoor environments. The main contributions of this article include the following aspects:

1. First, we optimize and employ YOLOv7-tiny on a single CPU and compensate for missed detections using Extended Kalman Filtering and the Hungarian algorithm to ensure the accuracy of semantic information.
2. Second, we propose a dynamic feature point removal algorithm to enhance the tracking and locating accuracy of the visual SLAM in dynamic environments by combining semantic information, geometric constraints, and depth information.
3. Third, we conduct a detailed experimental evaluation on various public datasets and real-world dynamic scenes, and demonstrate superior accuracy and robustness through rigorous testing and analysis.

The rest of this paper is organized as follows: Sect. 2 reviews related work on dynamic visual SLAM, Sect. 3 presents the details of the proposed method, Sect. 4 provides extensive quantitative and qualitative evaluations of the proposed method, and Sect. 5 concludes the paper and discusses future prospects.

## 2 Related work

### 2.1 Geometric-based visual SLAM methods in dynamic scenes

In visual SLAM systems, only static features satisfy inter-frame and global geometric constraints. Therefore, some researchers utilize geometric constraints to distinguish between static and dynamic features. Zhang et al. [12] employed the dense optical flow residual method to detect dynamic regions in the scene and reconstructed the static background using a framework similar to the algorithm proposed in [13]. Ji et al. [14] proposed a motion detection geometry module that clusters depth images and utilizes reprojection errors to identify dynamic areas and detect moving objects. Dai et al. [15] established a correlation between mapping points and distinguished dynamic points from static map points based on the assumption that the relative positions of static points remain consistent over time. Du et al. [16] employed a graph-based approach that integrates long-term consistency via conditional random fields (CRFs) to robustly detect and track dynamic objects over extended observations across multiple frames, thereby tackling the challenges of SLAM in dynamic environments. Wang et al. [17] segmented objects to identify moving objects, thereby estimating the camera pose and performing dense point cloud reconstruction. Zhang et al. [18] used particle filtering to process depth images, combining optical flow and grid-based motion statistics to estimate inter-frame transformations during the tracking stage.

There are also many geometric methods based on line features and plane features. Zhang et al. [19] matched the point and line features in the images captured by stereo camera to estimate the pose, and utilized dynamic grids and motion models to remove dynamic features. Fan et al. [20] proposed a approach based on plane features, utilizing a network to extract these features and incorporating plane constraints into the reprojection error calculation during backend optimization to ensure accuracy. Long et al. [21] proposed a approach tailored for dynamic planar environments. This method effectively separates dynamic planar objects based on rigid motion and tracks them independently, showcasing superior performance in localization and mapping. Geometric methods are straightforward to implement, and do not require pretrained models, making them highly computationally efficient. However, they have limitations such as a strong dependence on geometric features, poor performance in highly dynamic scenes, and a lack of deep understanding of the surrounding environment, which prevents the establishment of globally consistent maps. Therefore, they are often used in combination with other methods.

### 2.2 Semantic-based visual SLAM methods in dynamic scenes

Semantic-based visual SLAM primarily relies on deep learning techniques, such as object detection and semantic segmentation, to obtain semantic information. This is done to eliminate dynamic features during tracking and mapping. An et al. [22] were the first to use semantic segmentation methods for detecting and removing moving vehicles. Yu et al. [3] enhanced ORB-SLAM2 by incorporating a SegNet-based [8] semantic segmentation thread to manage dynamic objects. Runz et al. [23] utilized the semantic labels generated by Mask R-CNN [24] to segment moving objects, create semantic object masks, and establish object-level representations of dynamic objects. Jin et al. [25] proposed an unsupervised semantic segmentation model based on residual neural network structure, promoting the application of SLAM in various real-world scenarios. Chang et al. [26] utilized YOLACT for semantic segmentation and integrated geometric constraints to detect dynamic features beyond the segmentation mask. Wu et al. [27] improved the YOLOv3 network by incorporating depth difference and random sample consistency to distinguish dynamic features. Hu et al. [28] improved ORB-SLAM3 by adding geometric and semantic segmentation threads. They integrated multi-view geometry and semantic information to identify dynamic feature points. Cheng et al. [29] extended ORB-SLAM2 with object detection and semantic mapping, using a dynamic feature exclusion algorithm for 3-D octo-map and semantic object creation. Jin et al. [30] integrated ORB-SLAM3 with a saliency prediction model that comprehensively considers geometric, semantic, and depth information to focus on important areas and enhance the accuracy of visual SLAM. He et al. [31] combined YOLOv5, optical flow, and depth information to differentiate between foreground and background regions in images, treating the foreground areas as dynamic regions. Chen et al. [32] proposed a semantic SLAM method that integrates knowledge distillation and dynamic probability propagation. This approach effectively mitigates the impact of dynamic scenes on SLAM accuracy and processing speed through lightweight segmentation models and a static semantic keyframe selection strategy.

Some recent studies have proposed multimodal technologies, such as recovering missing depth images [33], and understanding and predicting human behavior and emotions from visual images [34]. These advancements will contribute to the development of visual SLAM systems. These studies have promoted the development of SLAM in dynamic environments, representing a substantial advancement in SLAM technology. While they enhance localization accuracy and map consistency, they also inevitably increase computational overhead and may result in the loss of some valuable information.

## 3 System overview

In this section, we will elaborate on the proposed method in detail from four different aspects. First, we will introduce the overall framework of the system. Second, we will discuss the deployment and implementation details of the object detecting thread. Subsequently, we will explain the inter-frame geometric constraint and depth-based RANSAC. Finally, we will focus on dynamic feature removal strategy.

### 3.1 System framework

The algorithm presented in this article is an improvement of the classic visual SLAM system, ORB-SLAM2. This algorithm operates through three primary parallel threads: tracking, local mapping, and loop closing. It has demonstrated outstanding performance in terms of both speed and accuracy. Therefore, this article introduces functional extensions and improvements based on the foundation provided by ORB-SLAM2.

As shown in Fig. 1, our system incorporates a parallel real-time object detecting thread. Normally, object detection processes are time-intensive, whereas ORB feature extraction and inter-frame tracking operate at faster speeds. To achieve real-time operation, we utilize parallel threads for object detection to minimize the blocking time of the SLAM system and optimize overall system efficiency. Next, the dynamic feature removal strategy proposed in this article is implemented to remove dynamic features. By removing these dynamic feature points, we then focus on robust static feature points for tracking purposes. Subsequently, keyframes are selected based on a specific keyframe decision strategy. These keyframes, devoid of dynamic features, are then forwarded to the local mapping and loop closing thread, where local and global pose optimization is conducted.

The detailed process of removing dynamic feature points is shown in Fig. 2. During system operation, RGB images captured by the RGB-D camera are sent to both the tracking and object detecting threads for feature point extraction and object detection. Next, a tightly coupled approach is used to integrate 2D semantic information, inter-frame geometric constraint, and depth threshold to effectively remove dynamic features. The semantic information includes the probability of semantic labels along with the 2D bounding box generated by the object detecting thread. The bounding box can identify the most probable location of dynamic feature points in the image. The geometric constraint utilize

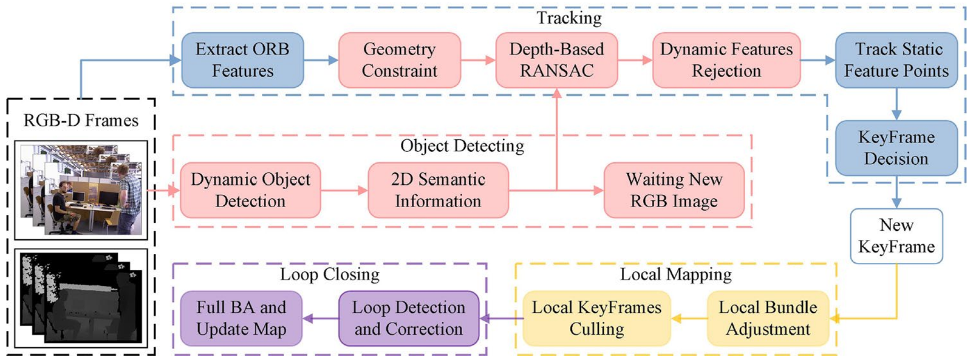

**Fig. 1** The overall framework of our system. Different colored boxes represent different threads

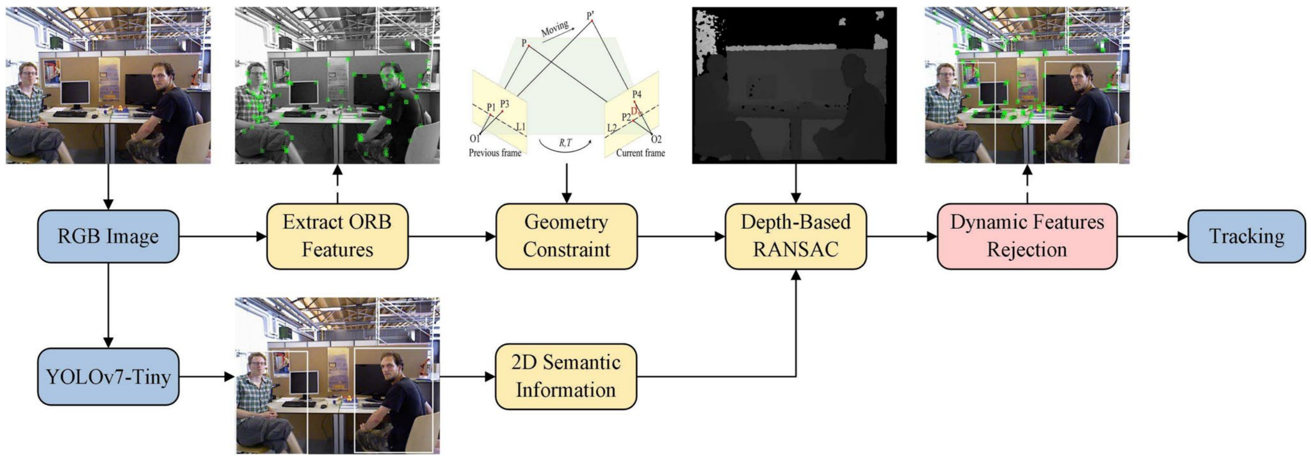

**Fig. 2** The process of removing dynamic features. First, ORB feature points are extracted, and dynamic objects are detected. Then, the proposed dynamic feature removal strategy is used to filter out dynamic features. Finally, only static features are retained for subsequent processing

the fundamental matrix to calculate the distance from each feature point to its corresponding epipolar line, followed by a comparison with an empirical threshold. The depth threshold within each 2D bounding box is iteratively calculated using an improved depth-based RANSAC algorithm. If a feature point's depth value exceeds the depth threshold, the feature point is removed. After processing the dynamic feature points, the remaining static feature points are accurately matched and subsequently used for tasks, such as tracking, local mapping, and loop closing.

## 3.2 Object detection

The visual SLAM system designed in this article is primarily designed for mobile robot applications. Under constraints such as limited space, cost, and power consumption, the deployment of object detection model faces a significant challenge, especially since many existing studies such as DS-SLAM [3], DynaSLAM [4] require GPU acceleration to operate effectively. This increases the flexibility and complexity of system deployment.

This paper selects YOLOv7-tiny, a lightweight object detection network, as the primary model for the object detecting thread, which is trained on the MS COCO dataset [35]. At the same time, considering the balance between speed and accuracy among various object detection networks, the deployment framework NCNN [36] is chosen. NCNN is a high-performance neural network inference framework designed for mobile devices, implemented in pure C++, and known for its ease of setup, rapid implementation, and seamless integration with the visual SLAM system.

When detecting dynamic objects, the absence of semantic information in certain frames can result in feature point mismatches, which can significantly impact the accuracy of the SLAM system. To address this issue, we introduce Extended Kalman Filtering (EKF) and the Hungarian algorithm to compensate for missed detections of dynamic objects. The EKF predicts the bounding box in the next frame, while the Hungarian algorithm associates the predicted box with its corresponding detection box.

## 3.3 Epipolar geometry constraint

The initial step in removing dynamic feature points using epipolar geometry involves matching feature points between two adjacent RGB image frames. To achieve this, the paper applies a pyramid-based Lucas–Kanade optical flow method, which establishes correlations between feature points across frames. Subsequently, the seven-point method based on RANSAC is used to compute the fundamental matrix $F$ between the two adjacent frames.

Figure 3 illustrates the principle of epipolar geometry. Consider two RGB image frames, denoted by $I1$ and $I2$, where $O1$ and $O2$ represent the optical center of the camera.

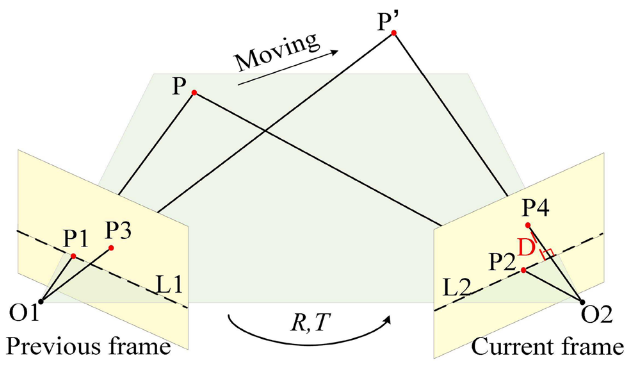

**Fig. 3** Epipolar geometry constraint

Let $P1$ and $P2$ represent a pair of correctly matched feature points in $I1$ and $I2$, respectively, with a common map point $P$. $L1$ and $L2$ represent polar lines. The fundamental matrix $F$ is solved using the following equation:

$$\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = P_1^T F P_2 = 0, \tag{1}$$

where $x_i$ and $y_i$ represent the pixel coordinates of the feature points $P1$ and $P2$ in their respective image coordinate systems, and $e_i$ represents each element of the fundamental matrix $F$. Then, the polar line $L_2$ of the current frame $I2$ can be calculated by the following equation:

$$L_2 = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = FP_1 = F \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}. \tag{2}$$

Here, $X$, $Y$, and $Z$ represent the vector form of the polar line $L_2$. The calculation formula of the distance $D$ from the feature point $P2$ to the corresponding polar line $L_2$ is

$$D = \frac{\left| P_2^T F P_1 \right|}{\sqrt{X^2 + Y^2}}. \tag{3}$$

In an ideal scenario, the point $P2$ in the current frame $I2$ should lie exactly on the corresponding polar line $L_2$. This alignment implies that, ideally, the distance $D$ between $P2$ and $L2$ equals zero. However, due to noise in real scenarios, the distance $D$ between $P2$ and $L2$ often exceeds zero. As shown in Fig. 3, when there is a dynamic point $P'$, the feature point $P2$ may shift to a new position like $P4$. The threshold $\epsilon$ helps in determining when the feature point is considered to be moving, which occurs when the distance $D$ exceeds this predefined threshold $\epsilon$.

## 3.4 Feature point depth-based RANSAC

Each frame of an RGB image undergoes processing by the object detecting thread to generate semantic labels and bounding boxes that describe the object's position. Some studies choose to directly remove all feature points located within the bounding box. which removes static feature points within the bounding box that are not associated with dynamic objects. However, when the bounding box in the RGB image occupies a substantial portion of the frame, it can result in the termination of the SLAM system due to insufficient remaining feature points.

Inspired by article [27], we utilize depth images to assist in identifying dynamic features. As shown in Fig. 4, it is evident that the depth value of the dynamic objects within the bounding box differs significantly from the static background. Therefore, the dynamic feature points can be removed utilizing depth information. Inspired by the approach described in [27], dynamic feature points are separated using depth images by improving the RANSAC iterative method based on depth values.

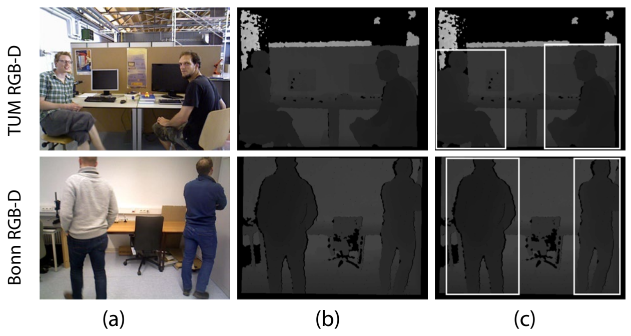

**Fig. 4** Some RGB images and depth images were acquired using RGB-D camera, along with bounding boxes

**Algorithm 1** Feature point depth-based RANSAC

**Input:**

- *Points*: The set of feature points in the bounding box.
- *Nums*: The number of iterations.
- *ImDepth*: The depth image corresponding to the current frame.

**Output:**

- *BestThres*: The depth threshold in the current bounding box.
- *MaxInliers*: The number of valid feature points in the current bounding box.

```
1:  $BestThres = 0, MaxInliers = 0, nIter = 0$;
2:  while $nIter < Nums$ do
3:      Select two points randomly and calculate the average depth value, $InitialVal = Average(d1, d2)$;
4:      $Inliers = 0$;
5:      $TotalDepth = 0$;
6:      for each point $p$ in $Points$ do
7:          Calculate the new value, $Thres = Average(d1, d2, d3)$;
8:          if $d3 > Thres$ then
9:              $++Inliers$;
10:             $TotalDepth = TotalDepth + d3$;
11:         end if
12:     end for
13:     if $Inliers > 0$ then
14:         $AveDepth = TotalDepth/Inliers$;
15:         if $Inliers > MaxInliers || AveDepth > BestThres$ then
16:             $MaxInliers = Inliers$;
17:             $BestThres = AveDepth$;
18:             $Nums = UpdateIterValue()$;
19:         end if
20:     end if
21:     $++nIter$;
22: end while
```

The description of depth-based RANSAC is presented in Algorithm 1. Utilizing the 2D semantic information generated by the object detecting thread, find the feature points located within the bounding box and store them in a set. Then, begin iterative processing by initializing a loop, randomly select two feature points from the set and calculate their average depth value $d_{\mathrm{ave}}$, which serves as the initial depth value. Subsequently, iterate through all the remaining feature points, recalculating the average depth value by incorporating the new feature points, and comparing it with the $d_{\mathrm{ave}}$. If the depth value exceeds the $d_{\mathrm{ave}}$, valid feature points should be included. After completing the traversal, calculate the average depth value of the effective feature points, and update the value from the previous cycle accordingly. Finally, return an optimal depth threshold. Feature points with depth values greater than this threshold are then classified as static feature points.

The depth-based RANSAC math model, which can be described as follows:

$$d_{\mathrm{ave}} = \frac{\sum_{i=1}^{N} d_i}{N}, \tag{4}$$

where $d_{\mathrm{ave}}$ represents the mean of depth, $N$ is the number of points, and $d_i$ represents the depth value of the selected points. We use $\omega$ to represent the percentage of inliers in the total number of points. In an iteration using $N$ points, the probability that at least one outlier in the selected points is $1 - \omega^N$, then the probability that at least one outlier continuously exists in $K$ iterations is $(1 - \omega^N)^K$. Using $P$ denote the probability that points randomly selected from the point set during $K$ iterations are all inliers. There is an equation as follows:

$$1 - P = (1 - \omega^N)^K. \tag{5}$$

Then, the calculation formula of $K$ is as follows:

$$K = \frac{\log(1 - P)}{\log(1 - \omega^N)}. \tag{6}$$

In our method, the number of $K$ is continuously adjusted based on the previous estimate $\omega$ and the desired probability $P$ to achieve a specific confidence level. Subsequently, we determine the optimal depth threshold for the feature points within the bounding box.

## 3.5 Dynamic feature rejection strategy

We will introduce the dynamic feature rejection strategy designed in this article, which integrates semantic information, geometric constraint, and depth threshold. Algorithm 2 describes the implementation process of the dynamic feature rejection strategy.

Let us discuss how to select the effective distance threshold $\epsilon$, which will directly affect the effectiveness of the dynamic feature rejection strategy. Therefore, we continuously track and record the distance from the feature points in adjacent frames to the corresponding epipolar line in a static scene. After analyzing the entire data, we found that the value of distance $D$ is mainly concentrated around 1.0. Hence, this article sets the distance threshold $\epsilon$ to 1.0. In an indoor environment, the probabilities and amplitudes of motion for different potential moving objects vary significantly. For example, pedestrians exhibit high dynamics, chairs exhibit low dynamics, and desks are typically static objects. Therefore, a value $W$ is assigned to represent the motion weight of the object. However, if the weight value is too small, too many moving dynamic points will be overlooked; conversely, if the weight value is too large, robust static points will be mistakenly classified and deleted. To accurately filter dynamic features while retaining static features, we utilize a weight scale ranging from 0 to 5. Pedestrians are assigned a weight of 5, chairs a weight of 3, and desks a weight of 1.

**Algorithm 2** Dynamic feature rejection strategy

```
Input:
 – P1: The set of feature points in the previous frame.
 – P2: The set of feature points in the current frame.
 – ε: The empirical threshold of distance.
 – F: The fundamental matrix relating two image frames.
 – BestThres: The depth threshold in the current bounding box.
Output:
 – S: The set of static feature points in the current frame.
 1: for each matched point pair $p1, p2$ in $P1, P2$ do
 2:     if $HaveDynamicObjects()\&\&IsInDynamicRegion(p2)$ then
 3:         if $(CalEpiLineDistance(p2, p1, F) \times WeightValue(p2) < \varepsilon)\&\&(Depth(p2) < BestThres)$ then
 4:             Append $p2$ to $S$;
 5:         end if
 6:     else
 7:         if $CalEpiLineDistance(p2, p1, F) < \varepsilon$ then
 8:             Append $p2$ to $S$;
 9:         end if
10:     end if
11: end for
```

For each feature point, we initially verify the presence of semantic information in the image frame after processing through the object detecting thread. If semantic information is present, we proceed to check if the feature point falls within the bounding box. Following this, calculations and assessments are conducted using the specified weights $W$ and thresholds $\epsilon$ as mentioned earlier.

## 4 Experiment and analysis

This section will comprehensively discuss the algorithm's performance in dynamic scenarios. We will systematically analyze and evaluate the proposed method using public datasets and real-world scenarios. First, we evaluate the tracking accuracy of the system using dynamic sequences from the TUM RGB-D dataset [37] and Bonn RGB-D dataset [38], and then compare it with current state-of-the-art dynamic visual SLAM solutions. Second, we compare the tracking performance of this algorithm with ORB-SLAM2 in real dynamic scenes. Next, we conduct an ablation experiment to verify the effectiveness of each module in the dynamic feature removal strategy. Finally, we analyze the real-time performance of the system. The experimental platform consists of a computer equipped with an Intel i5-8400 CPU, 16GB of memory, running the Ubuntu 18.04 operating system, and an NVIDIA Jetson Xavier NX 16GB development kit.

The absolute trajectory error (ATE) and relative pose error (RPE) are commonly used metrics for evaluating the performances of SLAM systems. ATE measures the direct difference between the estimated pose and the ground truth pose, providing an intuitive assessment of the algorithm's accuracy and the global trajectory consistency. RPE, on the other hand, quantifies the accuracy of pose differences between two frames separated by a fixed time interval, akin to measuring odometry error directly against ground truth. In each quantitative comparison table for the metrics, bold text is used to highlight the optimal results (e.g., error, processing time, etc.).

### 4.1 Performance evaluation on TUM RGB-D dataset

The TUM RGB-D dataset, provided by the Technical University of Munich, is a significant benchmark for visual SLAM systems. It comprises multiple sequences categorized into various usage scenarios, such as sitting, walking, and interactions involving objects like tables. In our experiments, we utilized four highly dynamic sequences from the fr3_walking for validation. These sequences depict two individuals walking in an office setting. Additionally, a low dynamic sequence is included as supplementary data. We selected ORB-SLAM2 [2], DS-SLAM [3], RDS-SLAM [8], Blitz-SLAM [20], and SG-SLAM [29] as the comparison algorithms. ORB-SLAM2 is the foundation of our algorithm, while DS-SLAM, RDS-SLAM, Blitz-SLAM, and SG-SLAM are outstanding dynamic visual SLAM algorithms from the past and recent years, providing strong reference significance.

Figure 5 shows the feature point extraction performance of our algorithm compared to ORB-SLAM2 using the TUM RGB-D dataset. Figures 6 and 7 show the ATE and RPE measurement diagrams, respectively, comparing the tracking trajectory with the ground truth of our algorithm and ORB-SLAM2 across different sequences. From a qualitative perspective, it is evident that the camera trajectories estimated by our algorithm are more consistent with the ground truth compared to ORB-SLAM2.

Tables 1, 2, and 3 show the quantitative comparison results of ATE and RPE between our algorithm and other algorithms across five dynamic sequences. Our algorithm achieves superior results. In the quantitative comparison results of ATE in Table 1, our algorithm shows significant performance improvements across four highly dynamic sequences, with increases of at least 96.3% and 96.7% in RMSE and SD compared to ORB-SLAM2. This improvement can be attributed to the effective integration of 2D semantic information provided by the object detecting thread, along with the combined impact of inter-frame geometric constraint and depth threshold based on feature points. In five dynamic sequences, our algorithm consistently outperforms DS-SLAM and RDS-SLAM, achieving

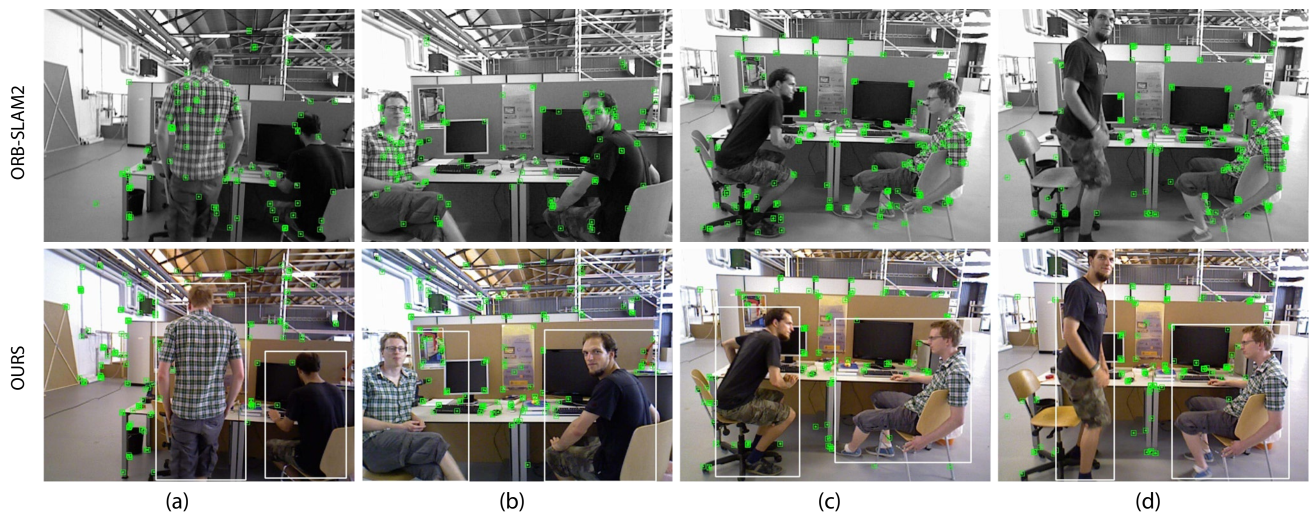

**Fig. 5** Performance comparison of feature point extraction between ORB-SLAM2 (top) and our algorithm (bottom) on the TUM RGB-D dataset

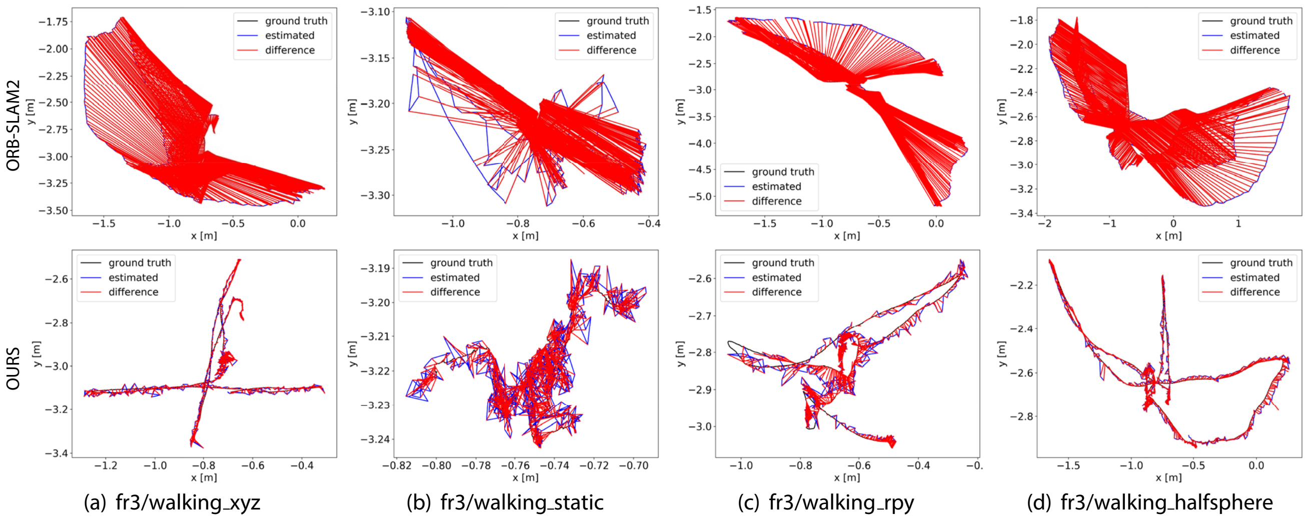

**Fig. 6** ATE results of the tracking trajectory for the algorithm proposed in this article and ORB-SLAM2 using the TUM RGB-D dataset

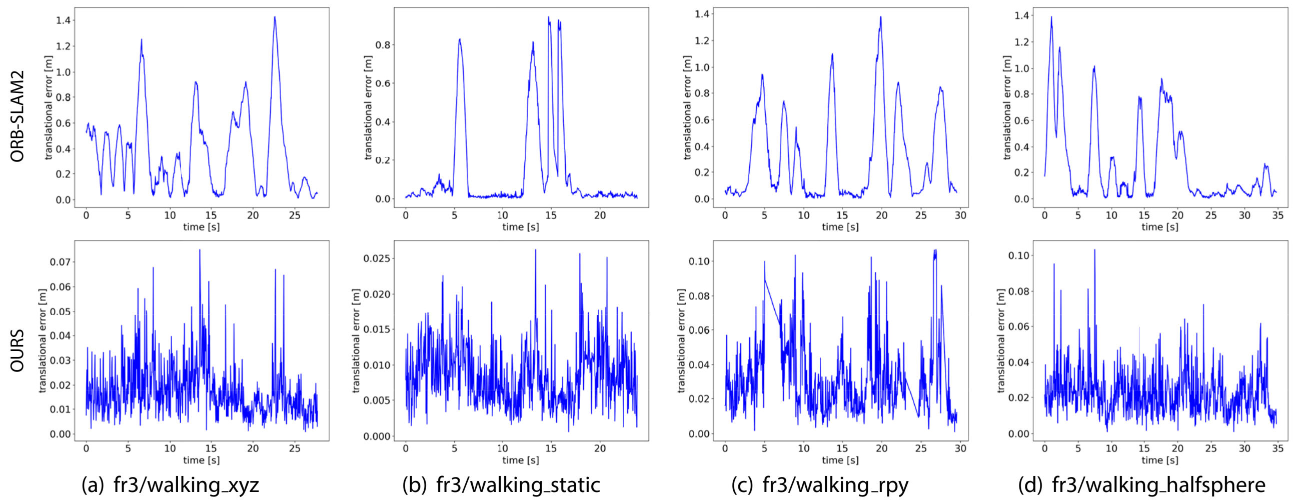

**Fig. 7** RPE results of the tracking trajectory for the algorithm proposed in this article and ORB-SLAM2 using the TUM RGB-D dataset

**Table 1** Results of metric absolute trajectory error (ATE) on TUM RGB-D dataset

| Seq. | ORB-SLAM2 | | DS-SLAM | | RDS-SLAM | | Blitz-SLAM | | SG-SLAM | | Ours | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD |
| walking_xyz | 0.9528 | 0.5604 | 0.0247 | 0.0161 | 0.0571 | 0.0229 | 0.0153 | 0.0078 | 0.0152 | 0.0075 | **0.0151** | **0.0074** |
| walking_static | 0.4116 | 0.1876 | 0.0081 | 0.0036 | 0.0206 | 0.0120 | 0.0102 | 0.0052 | 0.0073 | 0.0034 | **0.0072** | **0.0032** |
| walking_rpy | 1.1808 | 0.5958 | 0.4442 | 0.2350 | 0.1604 | 0.0873 | 0.0356 | 0.0220 | 0.0324 | 0.0187 | **0.0264** | **0.0143** |
| walking_half | 0.6917 | 0.4120 | 0.0303 | 0.0159 | 0.0807 | 0.0454 | **0.0256** | **0.0126** | 0.0268 | 0.0134 | 0.0258 | 0.0134 |
| sitting_static | 0.0082 | 0.0041 | 0.0065 | 0.0033 | 0.0084 | 0.0043 | – | – | **0.0060** | **0.0029** | 0.0066 | 0.0031 |

The bold text represents the optimal results in the experimental data. These results arethe best values selected after comparing the data under different conditions

**Table 2** Results of metric translation drift (RPE) on TUM RGB-D dataset

| Seq. | ORB-SLAM2 | | DS-SLAM | | RDS-SLAM | | Blitz-SLAM | | SG-SLAM | | Ours | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD |
| walking_xyz | 0.4864 | 0.3131 | 0.0333 | 0.0229 | 0.0426 | 0.0317 | 0.0197 | **0.0096** | **0.0194** | 0.0100 | 0.0205 | 0.0108 |
| walking_static | 0.2731 | 0.2380 | 0.0102 | 0.0048 | 0.0221 | 0.0149 | 0.0129 | 0.0069 | 0.0100 | 0.0051 | **0.0095** | **0.0042** |
| walking_rpy | 0.4503 | 0.3288 | 0.1503 | 0.1168 | 0.1320 | 0.1067 | 0.0473 | 0.0283 | 0.0450 | 0.0262 | **0.0368** | **0.0197** |
| walking_half | 0.4082 | 0.3131 | 0.0297 | 0.0152 | 0.0482 | 0.0360 | **0.0253** | 0.0123 | 0.0279 | 0.0146 | 0.0262 | **0.0121** |
| sitting_static | 0.0092 | 0.0046 | 0.0078 | 0.0038 | 0.0123 | 0.0070 | – | – | **0.0075** | **0.0035** | 0.0085 | 0.0037 |

The bold text represents the optimal results in the experimental data. These results arethe best values selected after comparing the data under different conditions

**Table 3** Results of metric rotation drift (RPE) on TUM RGB-D dataset

| Seq. | ORB-SLAM2 | | DS-SLAM | | RDS-SLAM | | Blitz-SLAM | | SG-SLAM | | Ours | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD |
| walking_xyz | 9.5312 | 6.1544 | 0.8266 | 0.5826 | 0.9222 | 0.6509 | 0.6132 | 0.3348 | **0.5040** | **0.2469** | 0.6225 | 0.3877 |
| walking_static | 4.5227 | 3.9107 | 0.2690 | 0.1182 | 0.4944 | 0.3112 | 0.3038 | 0.1437 | 0.2676 | 0.1144 | **0.2633** | **0.1134** |
| walking_rpy | 8.8086 | 6.4082 | 3.0042 | 2.3065 | 13.1693 | 12.0103 | 1.0841 | 0.6668 | 0.9565 | 0.5487 | **0.8593** | **0.4856** |
| walking_half | 8.5589 | 6.5261 | 0.8142 | 0.4101 | 1.8828 | 1.5250 | 0.7879 | 0.3751 | 0.8119 | 0.3878 | **0.7357** | **0.3533** |
| sitting_static | 0.2777 | 0.1237 | 0.2735 | 0.1215 | 0.3338 | 0.1706 | – | – | **0.2657** | **0.1163** | 0.2723 | 0.1200 |

The bold text represents the optimal results in the experimental data. These results arethe best values selected after comparing the data under different conditions

optimal results in three highly dynamic sequences, demonstrating superior overall performance. Blitz-SLAM excels in the “fr3/walking_half" sequence due to its effective handling of dynamic object boundaries using semantic segmentation technology, demonstrating outstanding accuracy. Both our algorithm and other algorithms show improvements in the “fr3/sitting_static" low dynamic sequence, albeit relatively limited, as ORB-SLAM2 already effectively suppresses low dynamic features. From the quantitative comparison results of RPE in Tables 2 and 3, it is evident that the trend of error reduction in RPE for translation and rotation aligns closely with that observed in the ATE comparison in Table 1. According to these comparison results, it can be concluded that the algorithm presented in this paper achieves smaller ATE and RPE errors in dynamic scenarios of the TUM RGB-D dataset, demonstrating promising performance compared to state-of-the-art algorithms.

## 4.2 Performance evaluation on Bonn RGB-D dataset

To assess the generalization performance of our algorithm, we conducted additional experimental evaluations using the Bonn RGB-D dataset. The Bonn RGB-D Dynamic Dataset comprises 24 dynamic sequence datasets released by the Photogrammetry and Robotics Laboratory of the University of Bonn in 2019 for evaluating RGB-D SLAM systems. For our evaluation, we selected nine representative dynamic sequences where individuals engage in activities, such as crowd walking, moving boxes, synchronized movements, and more. Each scene in the dataset includes ground truth trajectories of the camera for reference. In Sect. 4.1, some of the selected comparison algorithms were not tested on the Bonn RGB-D dataset. Therefore, in Sect. 4.2, we selected ORB-SLAM2 [2], StaticFusion [13], ReFusion [38], YOLO-SLAM [27], and SG-SLAM [29] as the comparison algorithms. StaticFusion proposes a probability-based static/dynamic segmentation to handle moving objects in dynamic scenes. ReFusion introduces truncated signed distance functions, utilizing color information to estimate the camera pose in dynamic scenes.

Figure 8 shows the feature point extraction performance of our algorithm compared to ORB-SLAM2 on the Bonn RGB-D dataset. It can be seen that the algorithm in this paper has a significant dynamic feature removal effect on the Bonn RGB-D dataset, showing a noticeable improvement compared to ORB-SLAM2. Table 4 presents the quantitative comparison results of ATE between the algorithm proposed in this paper and the selected comparison algorithms on nine dynamic sequences. Our algorithm demonstrated a 96.5% improvement in RMSE and a 95.3% improvement in SD compared to ORB-SLAM2 across the nine sequences. However, our algorithm did not achieve optimal performance on the “Moving_no_box1" sequence, possibly due to the presence of moving box that hindered the effective elimination

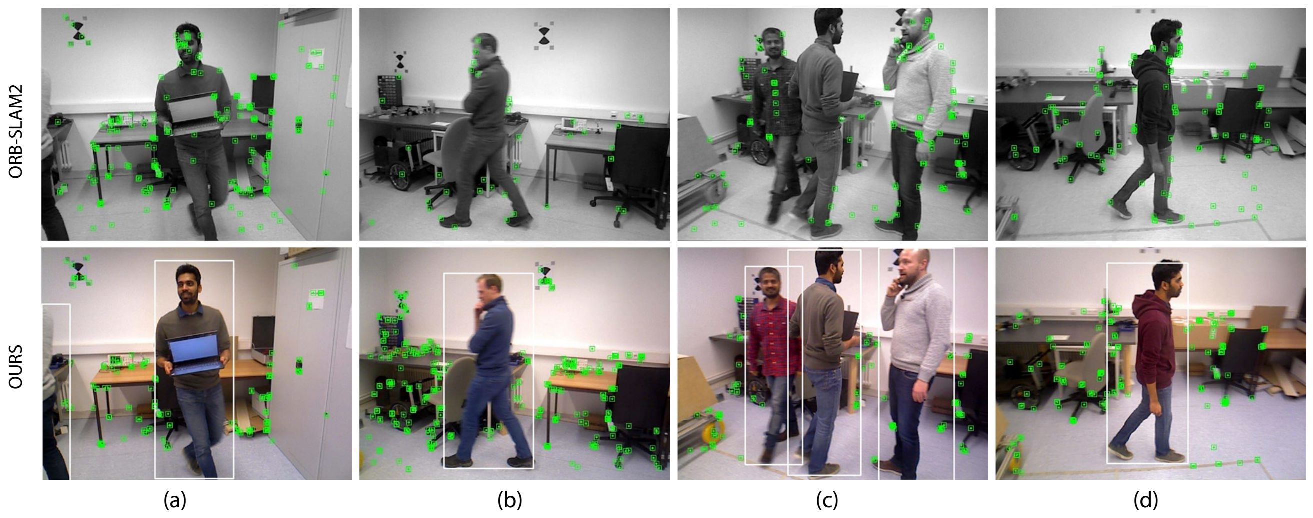

**Fig. 8** Performance comparison of feature point extraction between ORB-SLAM2 (top) and our algorithm (bottom) on the Bonn RGB-D dataset

**Table 4** Results of metric absolute trajectory error (ATE) on Bonn RGB-D dataset

| Seq. | ORB-SLAM2 | | StaticFusion | | ReFusion | | YOLO-SLAM | | SG-SLAM | | Ours | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD |
| crowd | 1.120 | 0.503 | 3.586 | – | 0.204 | – | 0.033 | – | 0.023 | 0.014 | **0.019** | **0.009** |
| crowd2 | 1.501 | 0.664 | 0.215 | – | 0.155 | – | 0.423 | – | 0.058 | 0.041 | **0.025** | **0.013** |
| crowd3 | 1.017 | 0.292 | 0.168 | – | 0.137 | – | 0.069 | – | 0.032 | 0.022 | **0.031** | **0.020** |
| moving_no_box | 0.317 | 0.113 | 0.141 | – | 0.071 | – | 0.027 | – | **0.019** | **0.008** | 0.023 | 0.009 |
| moving_no_box2 | 0.349 | 0.065 | 0.364 | – | 0.179 | – | 0.035 | – | **0.029** | 0.012 | 0.029 | **0.009** |
| person_tracking | 0.997 | 0.475 | 0.484 | – | 0.289 | – | 0.157 | – | 0.040 | 0.014 | **0.038** | **0.013** |
| person_tracking2 | 1.107 | 0.492 | 0.626 | – | 0.463 | – | **0.037** | – | 0.037 | 0.015 | 0.038 | **0.014** |
| synchronous | 1.068 | 0.521 | 0.446 | – | 0.441 | – | 0.014 | – | 0.323 | 0.182 | **0.013** | **0.008** |
| synchronous2 | 1.168 | 0.550 | 0.027 | – | 0.022 | – | 0.007 | – | 0.016 | 0.013 | **0.007** | **0.003** |

The bold text represents the optimal results in the experimental data. These results arethe best values selected after comparing the data under different conditions

of dynamic feature points. Nevertheless, the quantitative comparison results further emphasize the robustness and accuracy of the algorithm in highly dynamic environments.

### 4.3 Evaluation in the real-world environment

To showcase the robustness of this algorithm in real-world dynamic environments, we conducted experiments using the Intel RealSense D435i camera. We employed an Aprilgrid black and white checkerboard for calibrating the camera's internal parameters, ensuring accurate calibration for both the RGB and depth images, each sized at $640 \times 480$ pixels.

In our experiment, two individuals moved frequently within the camera's field of view. This movement had a substantial impact on the stability of the visual SLAM system, providing a valid test of the algorithm's performance. Figure 9 illustrates the effect of the proposed algorithm and

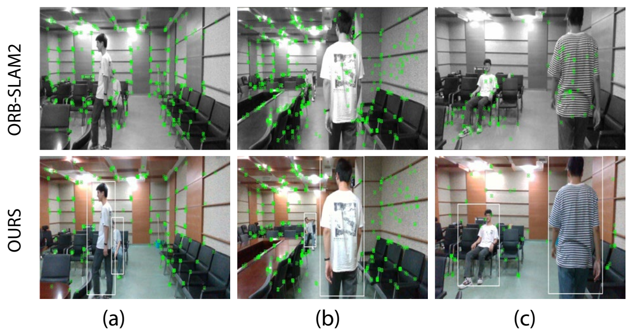

**Fig. 9** Performance comparison of the feature point extraction between ORB-SLAM2 (top) and our algorithm (bottom) in the real-world environment

ORB-SLAM2 on feature point extraction in real dynamic scenes. It is evident that the proposed algorithm effectively eliminates dynamic features, thus enhancing the accuracy of tracking and positioning within the SLAM system.

## 4.4 Ablation experiment

We conducted an ablation experiment to assess the impact of each proposed module (object detection, geometric constraint and depth-based RANSAC) on the overall system performance. As shown in Table 5, "O" represents object detection, "G" represents geometry constraint, and "D" represents the depth-based RANSAC. It is evident that among the five selected dynamic sequences, the "O" module demonstrates very high accuracy when operating independently. This is because it tends to remove a significant number of static points in the bounding box, which can easily result in tracking failure. The "G" module, when operating alone, can only eliminate feature points exhibiting large motion amplitudes, thereby resulting in limited overall improvement. Comparing the accuracy achieved when the "O" module operates alone, there is a significant improvement when combining the "O+G" modules in the "fr3_walking_rpy" sequence. Conversely, in the "fr3_walking_xyz" sequence, the accuracy decreases when the modules are combined. This is because the weight value of the dynamic object is too mechanical and cannot be adjusted dynamically. The algorithm presented in this paper introduces the depth threshold, represented by the "D" module, alongside the existing "O+G" modules. This new addition aids in effectively filtering dynamic feature points within the bounding box, thereby enhancing the accuracy of tracking and positioning.

## 4.5 Time analysis

The time cost is a critical indicator for assessing the quality of an SLAM system. The algorithm discussed in this article is specifically designed for real-time application scenarios, to evaluate the real-time performance of this algorithm, we measured the processing time for each frame of an RGB image and compared it with other algorithms. The experiment was conducted on two platforms: a computer with an i5-8400 CPU, 16GB of memory (without GPU acceleration), and an embedded platform using the NVIDIA Jetson Xavier Nx with 16GB of memory, running Ubuntu 18.04. During the experiment, we extracted 1000 ORB feature points per frame and executed the algorithm ten times to obtain the average processing time. This methodology allowed us to assess and compare the algorithm's efficiency and real-time capabilities across different hardware platforms.

The processing time of each frame is shown in Table 6. From the observed results, it is evident that visual SLAM systems like DS-SLAM and RDS-SLAM, which employ semantic segmentation networks, often incur significant time overheads, compromising real-time performance. In contrast, our method utilizes parallel thread technology to implement a lightweight target detection network. Compared to the ORB-SLAM2 system, the algorithm presented in this paper only marginally increases the frame processing time by a few milliseconds. This modest increase is sufficient to meet the real-time operational demands of the SLAM system.

**Table 6** The time-consuming cost

| Systems | Processing time per frame (ms) | Hardware platform |
|---|---|---|
| DS-SLAM | 76.46 | Intel i7 CPU, P4000 GPU |
| RDS-SLAM | 206.13 | Nvidia RTX 2080Ti GPU |
| Blitz-SLAM | 81 | Intel Core i5-6500 CPU |
| YOLO-SLAM | 696.09 | Intel Core i5-4288U CPU |
| ORB-SLAM2 | **32.50** | Intel Core i5-8400 CPU |
| Ours | **37.25** | Intel Core i5-8400 CPU |
| ORB-SLAM2 | 94.76 | Nvidia Jetson Xavier Nx |
| Ours | 103.5 | Nvidia Jetson Xavier Nx |

The bold text represents the optimal results in the experimental data. These results arethe best values selected after comparing the data under different conditions

**Table 5** The ATE (m) results of ablation experiments of Ours using different methods

| Methods | | | fr3/walking_xyz | | fr3/walking_static | | fr3/walking_rpy | | fr3/walkingw_half | | fr3/sitting_static | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *O* | *G* | *D* | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD | RMSE | SD |
| – | – | – | 0.9528 | 0.5604 | 0.4116 | 0.1876 | 1.1808 | 0.5958 | 0.6917 | 0.4120 | 0.0082 | 0.0041 |
| ✓ | – | – | 0.0162 | 0.0086 | 0.0079 | 0.0037 | 0.0434 | 0.0294 | 0.0307 | 0.0177 | 0.0086 | 0.0046 |
| – | ✓ | – | 0.3211 | 0.1655 | 0.0279 | 0.0231 | 0.7563 | 0.4315 | 0.1160 | 0.0940 | 0.0078 | 0.0039 |
| ✓ | ✓ | – | 0.0178 | 0.0092 | 0.0074 | 0.0037 | 0.0293 | 0.0192 | 0.0344 | 0.0215 | 0.0081 | 0.0039 |
| ✓ | ✓ | ✓ | **0.0151** | **0.0074** | **0.0072** | **0.0032** | **0.0264** | **0.0143** | **0.0258** | **0.0134** | **0.0066** | **0.0031** |

The bold text represents the optimal results in the experimental data. These results arethe best values selected after comparing the data under different conditions

## 5 Conclusion

In this paper, we introduce a real-time dynamic visual SLAM algorithm that integrates a parallel object detecting thread with ORB-SLAM2 to identify dynamic objects and generate 2D semantic information. Additionally, we propose an efficient strategy for removing dynamic feature points within the tracking thread. This strategy leverages semantic information, inter-frame geometry constraint, and a depth-based RANSAC approach to enhance the accuracy and robustness of the visual SLAM system in dynamic environments. We conducted qualitative and quantitative comparisons with state-of-the-art algorithms using datasets, such as TUM RGB-D, Bonn RGB-D, and real-world scenarios. Our algorithm demonstrates high tracking and positioning accuracy while maintaining excellent real-time performance.

Despite its strengths, this algorithm exhibits some limitations. For instance, when dynamic objects appear in irregular forms, such as individuals walking at a 45° angle due to camera rotation, the detection accuracy may decrease. In future research, we aim to address these challenges by exploring methods to estimate camera pose through sensor fusion, integrating information from multiple sensors to enhance accuracy and robustness. Furthermore, we are dedicated to developing a globally consistent multi-level map to support advanced tasks such as navigation and path planning. These efforts will contribute to further enhancing the algorithm's capabilities and applicability in diverse real-world scenarios.

**Data availability** The data that support the findings of this study are available from the corresponding author, upon reasonable request.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Wang, Y., Tian, Y., Chen, J., et al.: A survey of visual SLAM in dynamic environment: the evolution from geometric to semantic approaches. IEEE Trans. Instrum. Meas. **73**, 1–21 (2024)
2. Mur-Artal, R., Tardós, J.D.: Orb-slam2: an open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Trans. Robot. **33**(5), 1255–1262 (2017)
3. Yu, C., Liu, Z., Liu, X.J., et al.: DS-SLAM: a semantic visual SLAM towards dynamic environments. In: 2018 IEEE/RSJ international conference on intelligent robots and systems, pp. 1168–1174. IEEE (2018)
4. Bescos, B., Fácil, J.M., Civera, J., et al.: DynaSLAM: tracking, mapping, and inpainting in dynamic scenes. IEEE Robot. Autom. Lett. **3**(4), 4076–4083 (2018)
5. Bresson, G., Alsayed, Z., Yu, L., et al.: Simultaneous localization and mapping: a survey of current trends in autonomous driving. IEEE Trans. Intell. Veh. **2**(3), 194–220 (2017)
6. Hussain, K., Wang, X., Omar, Z., et al.: Robotics and artificial intelligence applications in manage and control of COVID-19 pandemic. In: 2021 International Conference on Computer, Control and Robotics, pp. 66–69. IEEE (2021)
7. Liu, J., Liu, R., Chen, K., et al.: Collaborative visual inertial slam for multiple smart phones. In: 2021 IEEE International Conference on Robotics and Automation, pp. 11553–11559. IEEE (2021)
8. Liu, Y., Miura, J.: RDS-SLAM: real-time dynamic SLAM using semantic segmentation methods. IEEE Access. **9**, 23772–23785 (2021)
9. Zhao, H., Shi, J., Qi, X., et al.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2881–2890. IEEE (2017)
10. Howard, A., Sandler, M., Chu, G., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 1314–1324. IEEE (2019)
11. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 7464–7475. IEEE (2023)
12. Zhang, T., Zhang, H., Li, Y., et al.: Flowfusion: dynamic dense rgb-d slam based on optical flow. In: 2020 IEEE International Conference on Robotics and Automation, pp. 7322–7328. IEEE (2020)
13. Scona, R., Jaimez, M., Petillot, Y.R., et al.: Staticfusion: background reconstruction for dense rgb-d slam in dynamic environments. In: 2018 IEEE international conference on robotics and automation, pp. 3849–3856. IEEE (2018)
14. Ji, T., Wang, C., Xie, L.: Towards real-time semantic rgb-d slam in dynamic environments. In: 2021 IEEE International Conference on Robotics and Automation, pp. 11175–11181. IEEE (2021)
15. Dai, W., Zhang, Y., Li, P., et al.: Rgb-d slam in dynamic environments using point correlations. IEEE Trans. Pattern Anal. Mach. Intell. **44**(1), 373–389 (2020)
16. Du, Z.J., Huang, S.S., Mu, T.J., et al.: Accurate dynamic SLAM using CRF-based long-term consistency. IEEE Trans. Vis. Comput. Graph. **28**(4), 1745–1757 (2020)
17. Wang, C., Luo, B., Zhang, Y., et al.: DymSLAM: 4D dynamic scene reconstruction based on geometrical motion segmentation. IEEE Robot. Autom. Lett. **6**(2), 550–557 (2020)
18. Zhang, C., Zhang, R., Jin, S., et al.: PFD-SLAM: A new RGB-D SLAM for dynamic indoor environments based on non-prior semantic segmentation. Remote Sens. **14**(10), 2445 (2022)
19. Zhang, B., Ma, X., Ma, H.J., et al.: DynPL-SVO: a robust stereo visual odometry for dynamic scenes. IEEE Trans. Instrum. Meas. **73**, 1–10 (2024)
20. Fan, Y., Zhang, Q., Tang, Y., et al.: Blitz-SLAM: a semantic SLAM in dynamic environments. Pattern Recognit. **121**, 108225 (2022)
21. Long, R., Rauch, C., Zhang, T., et al.: RGB-D SLAM in indoor planar environments with multiple large dynamic objects. IEEE Robot. Autom. Lett. **7**(3), 8209–8216 (2022)
22. An, L., Zhang, X., Gao, H., et al.: Semantic segmentation-aided visual odometry for urban autonomous driving. Int. J. Adv. Robot. Syst. **14**(5), 1729881417735667 (2017)
23. Runz, M., Buffier, M., Agapito, L.: Maskfusion: real-time recognition, tracking and reconstruction of multiple moving objects. In: 2018 IEEE International Symposium on Mixed and Augmented Reality, pp. 10–20. IEEE (2018)
24. He, K., Gkioxari, G., Dollár, P., et al.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969. IEEE (2017)

25. Jin, S., Chen, L., Sun, R., et al.: A novel vSLAM framework with unsupervised semantic segmentation based on adversarial transfer learning. Appl. Soft Comput. **90**, 106153 (2020)
26. Chang, J., Dong, N., Li, D.: A real-time dynamic object segmentation framework for SLAM system in dynamic scenes. IEEE Trans. Instrum. Meas. **70**, 1–9 (2021)
27. Wu, W., Guo, L., Gao, H., et al.: YOLO-SLAM: a semantic SLAM system towards dynamic environment with geometric constraint. Neural Comput. Appl. **34**, 6011–6026 (2022)
28. Hu, Z., Zhao, J., Luo, Y., et al.: Semantic SLAM based on improved DeepLabv3$^{+}$ in dynamic scenarios. IEEE Access. **10**, 21160–21168 (2022)
29. Cheng, S., Sun, C., Zhang, S., et al.: SG-SLAM: a real-time RGB-D visual SLAM toward dynamic scenes with semantic and geometric information. IEEE Trans. Instrum. Meas. **72**, 1–12 (2022)
30. Jin, S., Dai, X., Meng, Q.: Focusing on the right regions-guided saliency prediction for visual SLAM. Expert Syst. Appl. **213**, 119068 (2023)
31. He, J., Li, M., Wang, Y., et al.: OVD-SLAM: an online visual SLAM for dynamic environments. IEEE Sens. J. **23**(12), 13210–13219 (2023)
32. Chen, L., Ling, Z., Gao, Y., et al.: A real-time semantic visual SLAM for dynamic environment based on deep learning and dynamic probabilistic propagation. Complex Intell. Syst. **9**(5), 5653–5677 (2023)
33. Wang, Y., Cui, Z., Li, Y.: Distribution-consistent modal recovering for incomplete multimodal learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 22025–22034. IEEE (2023)
34. Wang, Y., Li, Y., Cui, Z.: Incomplete multimodality-diffused emotion recognition. Adv. Neural Inf. Process. Syst. **36,** 17117-17128 (2023)
35. Lin, T.Y., Maire, M., Belongie, S., et al.: Microsoft coco: common objects in context. In: 13th European Conference, pp. 740–755. Springer, Cham (2014)
36. NCNN. High performance neural network inference framework. Tencent. https://github.com/Tencent/ncnn (2017)
37. Sturm, J., Engelhard, N., Endres, F., et al.: A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 573–580. IEEE (2012)
38. Palazzolo, E., Behley, J., Lottes, P., et al.: ReFusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 7855–7862. IEEE (2019)