Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.057460

ARTICLE

DKP-SLAM: A Visual SLAM for Dynamic Indoor Scenes Based on Object Detection and Region Probability

Menglin Yin[1], Yong Qin[1,2,3,4,*] and Jiansheng Peng[1,2,3,4]

[1]College of Automation, Guangxi University of Science and Technology, Liuzhou, 545000, China

[2]Department of Artificial Intelligence and Manufacturing, Hechi University, Hechi, 546300, China

[3]Key Laboratory of AI and Information Processing, Education Department of Guangxi Zhuang Autonomous Region, Hechi, 546300, China

[4]Guangxi Key Laboratory of Sericulture Ecology and Applied Intelligent Technology, School of Chemistry and Bioengineering, Hechi University, Hechi, 546300, China

*Corresponding Author: Yong Qin. Email: 05005@hcnu.edu.cn

Received: 18 August 2024 Accepted: 23 October 2024 Published: 03 January 2025

ABSTRACT

In dynamic scenarios, visual simultaneous localization and mapping (SLAM) algorithms often incorrectly incorporate dynamic points during camera pose computation, leading to reduced accuracy and robustness. This paper presents a dynamic SLAM algorithm that leverages object detection and regional dynamic probability. Firstly, a parallel thread employs the YOLOX object detection model to gather 2D semantic information and compensate for missed detections. Next, an improved K-means++ clustering algorithm clusters bounding box regions, adaptively determining the threshold for extracting dynamic object contours as dynamic points change. This process divides the image into low dynamic, suspicious dynamic, and high dynamic regions. In the tracking thread, the dynamic point removal module assigns dynamic probability weights to the feature points in these regions. Combined with geometric methods, it detects and removes the dynamic points. The final evaluation on the public TUM RGB-D dataset shows that the proposed dynamic SLAM algorithm surpasses most existing SLAM algorithms, providing better pose estimation accuracy and robustness in dynamic environments.

KEYWORDS

Visual SLAM; dynamic scene; YOLOX; K-means++ clustering; dynamic probability

1 Introduction

Simultaneous localization and mapping (SLAM) is a crucial component of mobile robots, helping them perceive their surroundings through sensors, establish their current pose in unknown environments, and progressively build a globally consistent map. SLAM systems are categorized into two main types based on the sensors used: laser SLAM and visual SLAM, corresponding to the use of lidar and cameras on mobile robots. lidar is expensive, bulky, and often used in autonomous vehicles. With advancements in camera technology, we can now use more affordable, faster, higher-quality, and smaller cameras as sensors in systems. Cameras can provide rich color information, enabling human-machine interaction and making vision-based SLAM applications more versatile and popular. Over time, many excellent vision-based SLAM systems have been developed, such as ORB-SLAM3 [1], LSD-SLAM [2], and VINS-Mono [3].

However, most visual SLAM algorithms assume static scenes, but mobile robots often operate in dynamic environments where moving objects like people and animals can degrade system performance by causing errors to accumulate over time. Accurately distinguishing between dynamic and static areas is crucial for precise pose estimation. To address this challenge, some researchers use geometric constraints like RANSAC (random sample consensus) [4] to remove mismatched points that do not meet certain criteria. However, this approach is only effective when dynamic points occupy a small area and struggles with larger dynamic objects. In recent years, the integration of deep learning into SLAM systems has become popular, with researchers applying semantic constraints to identify and remove potential dynamic objects within a scene. This typically involves detecting dynamic objects through object detection or semantic segmentation to obtain their position information in the image, which is then combined with geometric techniques to filter out feature points. Research on Crowd-SLAM [5] suggests that object detection is more effective and faster than semantic segmentation in handling unseen moving objects. However, the drawback of object detection is that detection boxes are often larger than the actual dynamic objects. If all feature points within the detection box are rejected, the SLAM system might lack sufficient static points for pose estimation, potentially leading to localization failure. On the other hand, semantic segmentation methods are harder to train, and balancing accuracy with real-time performance during segmentation is challenging, making it difficult to achieve an optimal balance.

To tackle the problem of moving objects in dynamic environments, this paper proposes the DKP-SLAM algorithm. This algorithm aims to reduce the impact of dynamic objects by integrating three key processes: YOLOX [6] object detection, an improved K-means++ clustering method, and a dynamic point removal strategy. By refining dynamic object contours and segmenting regions, along with utilizing dynamic probability and geometric constraints, dynamic feature points are effectively eliminated. The main contributions of this paper are as follows:

(1) Based on the ORB-SLAM3 framework, a real-time RGB-D visual SLAM system is proposed. It adds a YOLOX object detection thread, optimized using TensorRT [7] to speed up inference, which obtains 2D semantic information of dynamic objects for subsequent operations, and provides a method to compensate for missed detections.

(2) An improved K-means++ clustering algorithm is proposed. This algorithm can adaptively determine the number of clusters based on depth information from image streams and changes in dynamic points, reducing noise interference and more accurately extracting dynamic object contours.

(3) A dynamic point removal strategy is proposed. By using detection boxes and masks to divide the image into low dynamic, suspicious dynamic, and high dynamic areas, different dynamic probability weights are assigned to feature points in each area. Combining geometric constraints, dynamic points are removed to accurately solve the camera pose.

2 Related Work

2.1 Dynamic SLAM Improved by Geometry Method

Geometric methods primarily focus on retaining only feature points that satisfy geometric constraints to avoid interference from dynamic objects. Kundu et al. [8] constructed two geometric constraints using multi-view geometry. The most commonly used is the epipolar constraint. However, the choice of threshold directly affects system performance, as it may not effectively distinguish between static and dynamic points. Li et al. [9] introduced a static weighting method for keyframe edge points to reduce dynamic object influence on pose estimation. Additionally, Dai et al. [10] proposed separating dynamic objects by analyzing static point cloud distribution, thus minimizing moving object interference. Wang et al. [11] utilized two fundamental matrix constraints to filter dynamic objects by identifying dynamic regions through mismatched points. They then segmented these regions using depth information clustering for more accurate dynamic region identification. In addition to these methods, direct methods for motion detection have also been explored. Nguyen et al. [12] developed a fast and efficient dynamic-static separation method based on optical flow, effectively distinguishing between moving and static objects. Sun et al. [13] proposed an innovative online motion removal method based on RGB-D data, which incrementally updates the foreground model without requiring prior information about moving objects, such as semantics or visual appearance. Cheng et al. [14] detected dynamic feature points by combining optical flow vectors with the fundamental matrix. While optical flow methods can achieve effective dynamic segmentation, they may lose accuracy in cases of sudden changes in scene lighting or severe camera shake.

2.2 Dynamic SLAM Improved by the Semantic Method

Recent research in dynamic visual SLAM has increasingly concentrated on integrating deep learning to acquire semantic information, which provides prior knowledge for enhancing SLAM systems. Several methods have been developed, primarily utilizing semantic segmentation and object detection model. For example, DS-SLAM [15] introduces a semantic segmentation thread that uses SegNet [16] to create masks within the ORB-SLAM2 [17] framework. It then analyzes these masks through optical flow and motion consistency detection; if the number of dynamic points within a mask exceeds a threshold, all feature points within that mask are discarded, leaving only the points necessary for pose estimation. On the other hand, DynaSLAM [18] adopts Mask-R-CNN [19] for pixel-level image segmentation and combines it with multi-view geometry to address the challenge of incomplete dynamic point coverage due to limited semantic information. Although its performance is impressive, this combination faces slower operational speeds due to network demands. In response, RDS-SLAM [20] proposes a novel keyframe selection strategy aimed at reducing delays in the acquisition of semantic information, thus enhancing system responsiveness. PR-SLAM [21] combines an improved lightweight semantic segmentation model SOLOv2 [22], with dynamic probability propagation to effectively eliminate dynamic points, while YOLO-SLAM [23] utilizes the lightweight object detection model Darknet19-YOLOv3 to identify dynamic objects and applies depth-enhanced RANSAC to filter feature points in those regions.

In addition, certain methods optimize model performance by deploying them on accelerated platforms. CDS-SLAM [24] accelerates object detection model inference using TensorRT in ORB-SLAM3, proposing a nine-part detection algorithm and adaptive thresholds to better handle dynamic objects and benefit from a wider array of detection categories. SG-SLAM [25] integrates an object detection model on NCNN (ncnn convolutional neural network), combining geometric and semantic information to quickly remove dynamic points, thereby maintaining high speeds while creating semantic metric maps. However, it encounters robustness issues where the loss of semantic information due to model inference performance results in tracking failures during system operation. Semantic SLAM [26] attempts to strengthen the detection framework and reduce semantic information loss by tracking adjacent frames and using epipolar constraints to filter feature points, but it may misclassify some static points as outliers. In the current advanced detection domain, Ge et al. [27] proposed the NEAL (neural attention learning approach) method, which enhances the attention response of detection models through an end-to-end training process without introducing additional network structures, allowing the models to focus more accurately on important features in dynamic scenes, effectively improving detection accuracy. Additionally, DFD-SLAM [28] implements a precise strategy for removing dynamic points, enabling the identification of partially static regions on dynamic objects, while MDP-SLAM [29] introduces an adaptive mask expansion algorithm that utilizes semantic information from previous frames to extend the mask coverage of the current frame. This method applies K-means clustering to extract contours of dynamic objects within the mask, effectively combining semantic and geometric techniques to eliminate dynamic points. These systems aim to improve the real-time performance and accuracy of pose estimation by effectively managing the operational performance of network models and mitigating the impact of dynamic elements in various scenarios.

3 System Overview

3.1 System Framework

The DKP-SLAM algorithm enhances the classical ORB-SLAM3 system by integrating three key components: the detection module, the K-means++ clustering module, and the dynamic point removal module. In Fig. 1, red areas highlight these improvements, while blue areas show the unmodified ORB-SLAM3 parts. The detection module adds a parallel thread to process new RGB image frames, using the YOLOX model, accelerated by TensorRT, to extract 2D detection boxes. If detections are missed, optical flow matching is used for camera pose tracking. The clustering module then applies an improved adaptive threshold K-means++ algorithm to more accurately define dynamic object contours in the depth image. Lastly, the dynamic point removal module assigns weights to feature points based on their varying dynamic probabilities within different regions and effectively filters out dynamic points while retaining static ones by combining these weights with geometric constraints.

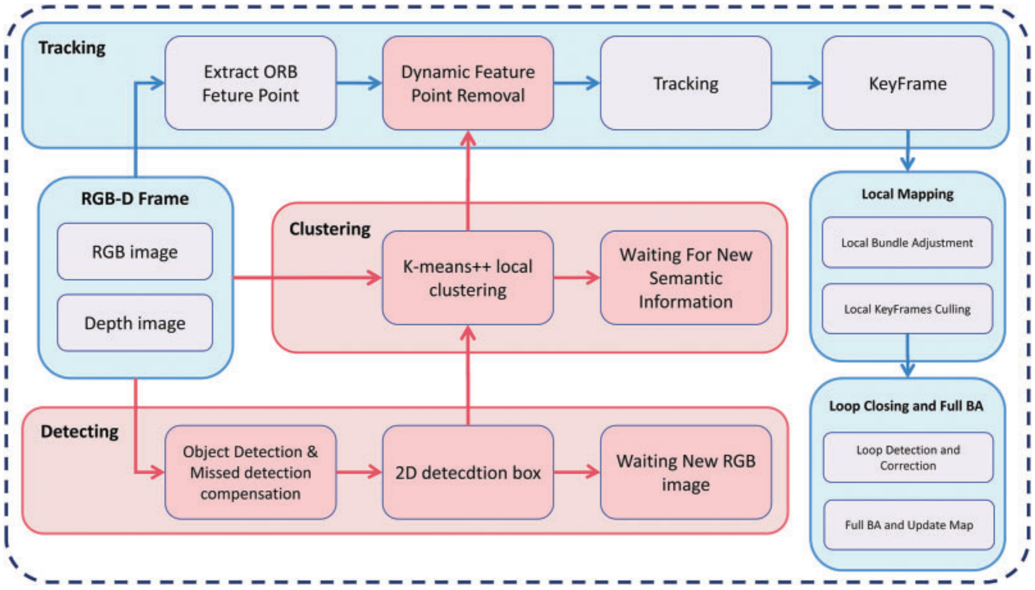


Figure 1: DKP-SLAM system diagram

3.2 Object Detection

In selecting the model for our system, we considered the portability of the system and the construction of accelerated inference with TensorRT, choosing from the relatively mature YOLO series. We compared YOLOv5, YOLOv8 [30], and YOLOX, as shown in Table 1.

Table 1: Comparison of different object detection models on the COCO dataset

Method	MAP (%)	PARAMS (M)	GFLOPs
YOLOv8n	37.3	3.2	8.7
YOLOv8s	44.9	11.2	28.6
YOLOv8m	50.2	25.9	78.9
YOLOv5s	36.7	7.3	17.1
YOLOv5m	44.5	51.4	51.4
YOLOXs	39.6	9	26.8
YOLOXm	46.4	25.3	73.8

It is important to state that the accuracy of semantic information does not have a significantly influence on the SLAM system. Due to the geometric verification strategy for dynamic objects, subsequent operations can still proceed even with detection errors. However, MAP (mean average precision) is not completely useless in this field. If a missed detection occurs, it will lead to a loss of semantic information, which in turn causes the removal of consecutive dynamic points to be incomplete, potentially leading to a system crash. Therefore, we need to refer to MAP, but it should not be considered an absolute reference. In contrast, we are more concerned with the number of model parameters, as this metric directly impacts the inference speed and computational power requirements. Naturally, we want each frame to be processed as quickly as possible.

The data in the Table 1 is based on the official COCO [31] test results, and we selected relatively small weight files for each model for comparison. We found that most M-level weights significantly increased processing time per frame without improving the system's accuracy. YOLOv8n and YOLOv5s exhibited numerous missed detections, while the differences between YOLOv8s and YOLOXs were negligible. Therefore, we ultimately selected YOLOXs as our inference model, and subsequent experiments confirmed that this model is sufficiently effective for our SLAM system.

To better acquire prior semantic information in dynamic environments, we use object detection to extract the location of various objects in RGB images. Specifically, in the detection module, we added a parallel thread to deploy the YOLOX model on the TensorRT platform. This platform leverages GPU (graphics processing unit) acceleration to improve the inference speed of object detection results. This setup avoids blocking while waiting for semantic information, optimizing system performance and ensuring that the SLAM system can perform real-time localization and mapping.

3.3 Missed Detection Compensation

We also considered the issue of detection failures that the YOLOX detection algorithm may encounter in certain situations. For instance, as shown in Fig. 2, detection failures are more common when dynamic objects are at the edge of the image or moving quickly. In dynamic scenarios, prior semantic information is crucial, and our goal is to avoid tracking failures caused by missed detections. Therefore, we have incorporated a missed detection compensation method in the detection module to address these potential detection failures. Due to the continuity of object movement and camera frame capture, two frames of images are captured within a short time interval. Therefore, we can utilize the distribution of dynamic points within the detection box from the previous frame to determine whether any semantic information related to dynamic points is missing in the current frame. If a loss is identified, we employ the Lucas-Kanade (LK) optical flow method to directly track the positions of all feature points from the previous frame to the current frame.

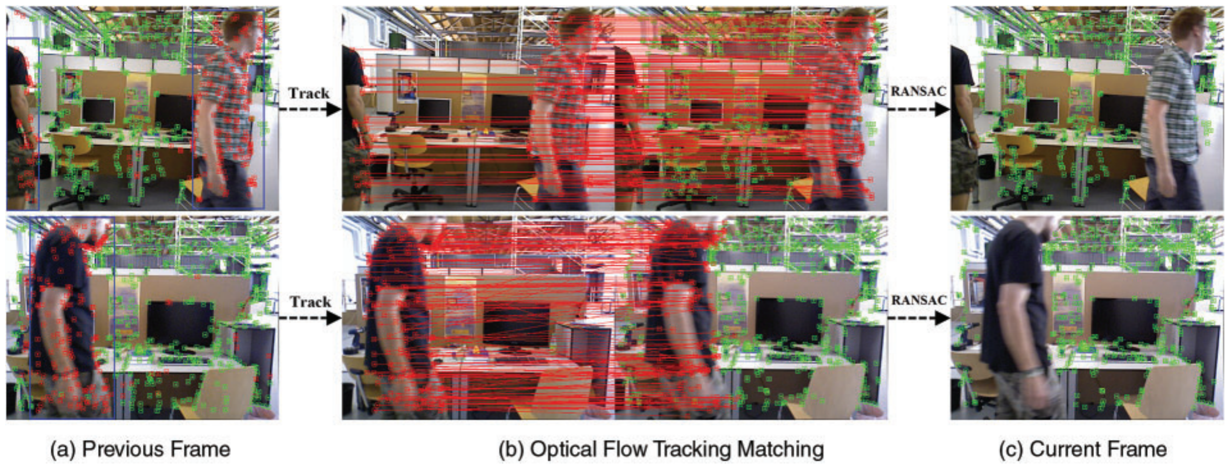


Figure 2: Missed detection compensation process

Our goal is to retain as many static points as possible to mitigate the issue of semantic loss. Consequently, the matching points of dynamic feature points on dynamic objects from the previous frame will be discarded in the current frame, leaving the remaining points for tracking. However, the optical flow method may lead to static feature points from the previous frame being incorrectly tracked to dynamic objects. To address this, we utilize RANSAC to further filter outliers. The primary reason for selecting RANSAC is its ability to effectively estimate model parameters in the presence of outliers. RANSAC can identify and discard these incorrect matches by randomly sampling points and evaluating their consistency, thus preserving point pairs that more accurately reflect the environment. This is crucial for camera pose estimation as it ensures the quality of the input feature points.

Fig. 2 illustrates our entire compensation process for omitted detections. Image (a) depicts the previous frame, which includes semantic information and the detected dynamic feature points. Image (b) shows the tracking of feature points from the previous frame to the current frame using the optical flow method, with matching pairs of dynamic points within the detection box connected by red lines. Image (c) displays the current frame with semantic information lost due to missed detections, along with the results of further RANSAC filtering applied to the remaining feature point pairs to eliminate incorrect matches. It is evident that there are almost no dynamic points left on the dynamic objects, indicating that our method effectively compensates for the semantic loss caused by missed detections.

3.4 Clustering Module

Although YOLOX object detection can quickly obtain detection boxes for dynamic objects, these boxes are often large and cannot provide precise contours of dynamic objects. Directly removing these boxes would result in the loss of many static feature points. Therefore, we designed a clustering module based on the K-means++ clustering algorithm for a more detailed extraction of dynamic objects.

The K-means++ clustering algorithm is an unsupervised clustering algorithm that is simple to implement and operates quickly. Compared to the traditional K-means clustering algorithm, K-means++ can better optimize the selection of initial cluster centers, usually converging to the optimal solution more quickly and avoiding local optima more effectively, leading to better clustering results. However, it is very sensitive to changes in the number of clusters, which can lead to under-clustering or over-clustering, preventing dynamic objects from being correctly distinguished. To address this issue, this paper proposes an adaptive threshold K-means++ clustering algorithm based on prior depth information and 2D semantic information. This algorithm can adaptively determine the number of clusters during SLAM operation. Algorithm 1 demonstrates our specific improvements. The steps are as follows:

(1) First, we extract the target detection boxes from both the previous and current frames. This step aims to reduce computational load and focus on the detection box areas to achieve better clustering results. Subsequent clustering is limited to the target box regions.

(2) During the SLAM system's operation, the time interval between two consecutive frames is very short, and the average depth of dynamic objects between the two frames falls within a small range. Therefore, we use the dynamic points and their average depth value T from the previous frame as priors for the current frame. By calculating the proportion R of dynamic object pixels within the detection box in the previous frame, we search the depth value range in the current frame starting from the average depth T, ensuring that the proportion in the current frame approximates R. This determine the number of dynamic points N in the current frame.

(3) To accurately segment the dynamic objects, we determine the number of clusters using the maximum depth value of the current frame and the number of dynamic points N, which results in determining the number of clusters K:

$$K = \frac{D_{\max}}{N}. \tag{1}$$

We use the value K as the number of clusters for the K-means++ algorithm to segment the image. The category with the smallest distance between its average depth value and the previous frame's average depth value T is identified as the dynamic object. This object is marked in red.

(4) We performed post-processing on the image to ensure it met our expectations. After converting the image to a binary image, we applied morphological erosion and dilation operations, filtering out color blocks with morphological differences and mismatched appearances. This reduced noise interference and ultimately obtained the contours of the dynamic objects.

Fig. 3 illustrates the comparison between the improved K-means++ clustering algorithm and the original version, highlighting the effectiveness of the enhancements. Image (a) displays the original depth image, which serves as the baseline for comparison. In Image (b), the original K-means++ algorithm is applied; however, due to an improper setting of the number of clusters, both underclustering and overclustering are observed. This inadequacy results in incomplete and inaccurate contours of the dynamic objects, causing them to partially blend into the background environment, thereby making it difficult to distinguish between dynamic objects and the surrounding area. As demonstrated in Image (c), our improved method significantly enhances the clustering process by more effectively determining the optimal number of clusters. This improvement allows for a much clearer separation between dynamic objects and the background in the depth images, resulting in more accurate and complete object contours. The enhanced algorithm better captures the details of the dynamic objects, ensuring they are distinctly separated from the background. Finally, Image (d) demonstrates the binary image of the dynamic object contours after noise has been removed through the post-processing process, leaving only clean contours.

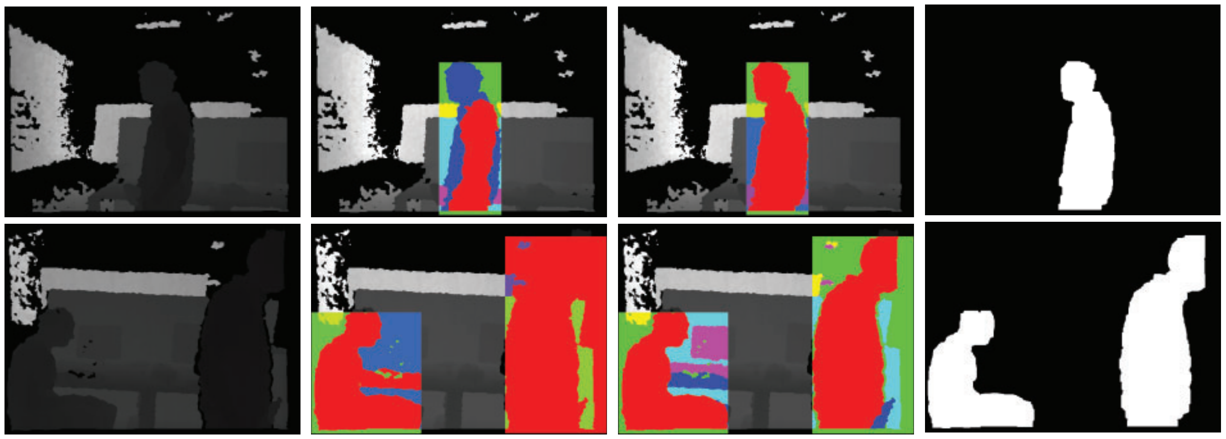


Figure 3: Comparison of clustering effects

Algorithm 1: Improve adaptive threshold K-means++ clustering algorithm

```
Input: Previous frame depth image F_pre, Current frame depth image F_cur
Output: Dynamic object mask
 1:  F_pre_box, F_cur_box = ExtractDetectionBoxRegion (F_pre, F_cur)
 2:  T = CalculateMeanDepthOfDynamicPoints (F_pre_box)
 3:  R = CalculateProportionOfDynamicPoints (T, F_pre_box)
 4:  N = MatchDomainPixels (R, T, F_cur_box)
 5:  K = MaxDepthValue (F_cur_box)/N
 6:  labels = KMeansPlusPlus (F_cur_box, K)
 7:  mask = ComputeMinDepthDistance (labels, F_cur_box, T)
 8:  Convert the mask into a binary image and perform image opening operation on it
 9:  Remove color blocks with morphological differences and appearance that do not match
10:  Return the filtered mask
```

3.5 Dynamic Point Removal Module

In this module, to eliminate the impact of dynamic points on the SLAM system, we have designed a dynamic feature point removal algorithm that combines regional probability and geometric constraints. The specific improvements are detailed in Algorithm 2.

First, we used the LK optical flow to track the positions of the feature points from the previous frame to the current frame and eliminated the matching point pairs with morphological differences. Then, the RANSAC 7-point method is used to calculate the matching point pairs, which are divided into inliers and outliers. The inliers can obtain the stable basic matrix of the previous and subsequent frames, and the epipolar line of the current frame is calculated. Specifically:

$$P_1 = [u_1, v_1, 1], P_2 = [u_2, v_2, 1], \tag{2}$$

where P_1 and P_2 represent the matching points from the previous and current frames, with u and v denoting the horizontal and vertical coordinates of the feature points in the image. If we denote the fundamental matrix as F, the calculation formula for the polar line L_1 is as follows:

$$L_1 = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = FP_1 = F \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}, \tag{3}$$

where X, Y, and Z represent the line vector of the epipolar line. The epipolar line constraint can be expressed as follows:

$$P_2^T FP_1 = P_2^T L_1 = 0. \tag{4}$$

The distance D between the matching point and its corresponding epipolar line can be calculated as follows:

$$D = \frac{\left|P_2^T FP_1\right|}{\sqrt{\|X\|^2 + \|Y\|^2}}. \tag{5}$$

If the distance D between the feature point and the epipolar line exceeds a predefined threshold, the feature point is considered an outlier. However, relying solely on geometric methods has its limitations and may not be very effective. In the previous sections, semantic information was obtained from the detection and clustering modules. To achieve a more accurate initial pose, we further refined the dynamic probability within the region. The steps are as follows:

(1) Divide the mask obtained from clustering into highly dynamic regions. The parts within the object detection box that do not belong to the mask are classified as suspicious dynamic regions, while the remaining parts are classified as low dynamic regions.

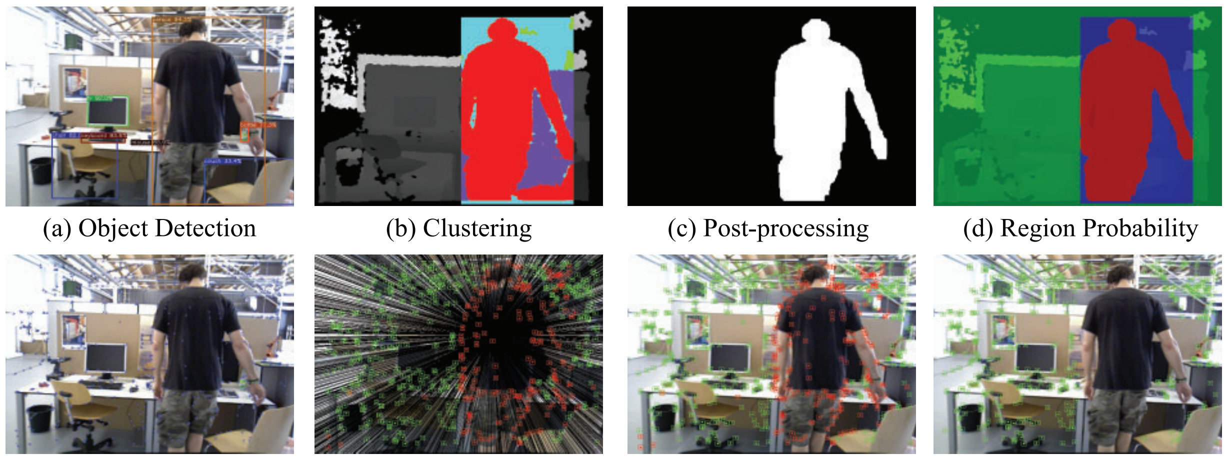
(2) For highly dynamic and low dynamic regions, dynamic weight values W ranging from 1 to 10 are introduced based on their likelihood of movement. For example, a person usually maintains a motion posture, so $W = 10$; whereas tables and bookshelves typically serve as static backgrounds, so $W = 1$. Dynamic points are detected using these weights in conjunction with our predefined empirical thresholds.

(3) For suspicious dynamic regions, we consider that the mask may sometimes be incomplete due to missing depth values in the depth image. As the feature points become closer to the dynamic object, the probability of being dynamic increases accordingly, and vice versa. Therefore, to better handle the feature points in suspicious dynamic regions, we design the dynamic weight values as follows:

$$W = W_{high} - \left(W_{high} - W_{low}\right) \cdot \left(1 - \frac{d}{d_{\max}}\right)^s, \tag{6}$$

where W_{high} and W_{low} represent the dynamic weights of the high and low regions, respectively, d represents the distance of the feature points from the mask, $d_{\max}$ is the maximum distance of the feature points within the detection box from the mask, and S represents the proportion of the suspicious region within the detection box area.

Fig. 4 illustrates the detailed process of our dynamic point removal. In Image (a), after obtaining the 2D semantics, our clustering method, as shown in Image (b), successfully captures the dynamic object mask marked in red. Image (c) shows the post-processing step, where the mask is refined to remove any excess dynamic mask portions. The final binary image provides prior knowledge for our dynamic point removal module. In Image (d), the image is divided into different regions: green represents low dynamic areas, blue represents suspicious areas, and red represents high dynamic areas. Image (e) depicts the optical flow vector map for feature point tracking. Finally, in Image (f), by combining the dynamic weights of feature points with epipolar constraints, dynamic feature points are successfully detected. Image (g) shows the clean feature point map without the epipolar lines, and Image (h) displays the final result after removal.


(a) Object Detection (b) Clustering (c) Post-processing (d) Region Probability

(e) Optical Flow (f) Epipolar Constraint (g) Detected Dynamic Points (h) Removal of Dynamic Points

Figure 4: Dynamic feature point removal process

Algorithm 2: Dynamic feature point removal algorithm

```
Input: Previous frame, $F_1$; Current frame, $F_2$; Previous frame's feature points, $P_1$; Current frame's
        feature points, $P_2$; Standard empirical thresholds, $\varepsilon_{std}$;
Output: The set of static points, S;
 1:  $P_2$ = CalcOpticalFlowPyrLK($F_1$, $F_2$, $P_1$)
 2:  Remove outliers in $P_2$
 3:  FundmentalMatrix = FindFundamentalMat($P_1$, $P_2$, 7-point method based on RANSAC)
 4:  for each matched pair $p_1$, $p_2$ in $P_1$, $P_2$ do
 5:    if (IsWithinMask($p_2$)) then
 6:      if (CalcEpiLineDistance($p_1$, $p_2$, FundmentalMatrix) $\times$ GetMaskWeight ($p_2$) $< \varepsilon_{std}$) then
 7:        Append $p_2$ to S
 8:      end if
 9:    else if (IsWithinDetectionBox($p_2$)) then
10:      if (CalcEpiLineDistance($p_1$, $p_2$, FundmentalMatrix) $\times$ GetBoxWeight ($p_2$) $< \varepsilon_{std}$) then
11:        Append $p_2$ to S
12:      end if
13:    else
14:      if (CalcEpiLineDistance($p_2$, $p_1$, FundmentalMatrix) $< \varepsilon_{std}$) then
15:        Append $p_2$ to S
16:      end if
17:    end if
18:  end for
```

4 Experiment and Analysis

4.1 Experiment Introduction

In this paper, we conducted experiments using the TUM RGB-D dataset [32], a well-recognized benchmark in the field of SLAM. The dataset includes a substantial number of image sequences captured by an RGB-D camera in dynamic environments, along with the corresponding ground truth trajectory data. We selected four sequences from highly dynamic environments and two sequences from less dynamic environments to test performance. These sequences are labeled as "W/half," "W/rpy," "W/static," "W/xyz," "S/static," and "S/xyz." Here, "W" denotes sequences involving a walking person, while "S" refers to sequences with a sitting person. The latter part of the labels indicates the different camera motion patterns: "half" represents hemispherical motion, "rpy" signifies rotation about different angles, "static" indicates a stationary state, and "xyz" denotes movement along the xyz axes.

In the experiment, we rigorously evaluate the performance of SLAM algorithms using two essential error metrics: Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). These metrics are assessed through root mean square error (RMSE) and standard deviation (S.D). ATE is a crucial metric that quantifies the deviation between the estimated trajectory of the SLAM algorithm and the actual ground truth trajectory. By measuring this discrepancy, ATE offers insights into the algorithm's precision, accuracy, and global consistency, which are vital for applications requiring reliable navigation and mapping. On the other hand, RPE focuses on the incremental errors between consecutive poses, evaluating both rotational (r.RPE) and translational (t.RPE) components. These errors are directly obtained from odometry data, which is fundamental for understanding the short-term consistency and stability of the SLAM system. The experiments were performed on a laptop equipped with an Intel i7-11700 CPU, NVIDIA GeForce RTX 2080Ti GPU, and 16 GB of RAM, running Ubuntu 18.04.

The algorithm proposed in this paper, DKP-SLAM, is an improvement based on ORB-SLAM3. To ensure a unified experimental benchmark, we selected RDS-SLAM and CDS-SLAM, which are also improved based on ORB-SLAM3, for comparison. Additionally, to further evaluate the advancement of the DKP-SLAM algorithm, we conducted detailed experimental comparisons with various other advanced SLAM algorithms. To comprehensively validate the effectiveness of each module of our algorithm, we designed ablation experiments and conducted real-time analysis. These experiments and analyses collectively confirmed the performance of DKP-SLAM in different application scenarios.

4.2 Comparative Experimental Results

As shown in Tables 2 and 3, the comparison between our proposed DKP-SLAM and ORB-SLAM3 (denoted as O3 in the table), along with other systems based on ORB-SLAM3, such as RDS-SLAM and CDS-SLAM, indicates that DKP-SLAM achieved the best ATE performance for both highly dynamic and low dynamic sequences. RDS-SLAM's dynamic probability propagation struggles in complex scenarios, while CDS-SLAM removes non-dynamic static points within detection boxes. In contrast, our algorithm more effectively distinguishes dynamic objects, preserving static points within detection boxes and yielding higher accuracy. This leads to significantly improved localization accuracy in dynamic scenes by avoiding error accumulation caused by dynamic objects.

To further evaluate our algorithm, Tables 4 and 5 compare our system with other advanced SLAM systems using ATE and t.RPE metrics. This includes two semantic segmentation-based algorithms, DS-SLAM and DynaSLAM, and three object detection-based systems: Crowd-SLAM, YOLO-SLAM, and SG-SLAM. The results show that DKP-SLAM performs well across both high and low dynamic sequences. In the high-dynamic W/rpy sequence, involving camera rotation, object detection's semantic information becomes less accurate due to camera movement, degrading system performance. DynaSLAM, which uses pixel-level semantic segmentation, effectively segments dynamic objects during rotation but suffers from slow frame processing, reducing overall performance. In contrast, DKP-SLAM operates in real-time and achieves state-of-the-art pose estimation accuracy across all tested sequences.

Table 2: Compared with SLAM systems based on ORB-SLAM3, the RMSE of ATE

Sequence	ATE (RMSE)			
	O3	RDS	CDS	OURS
W/half	0.2953	0.0807	0.0295	**0.0243**
W/rpy	0.5947	0.1604	0.0384	**0.0345**
W/static	0.2641	0.0206	0.0082	**0.0068**
W/xyz	0.3796	0.0571	0.0165	**0.0156**
S/static	0.0112	0.0084	0.0080	**0.0053**
S/xyz	0.0095	0.0135	0.0091	**0.0072**

Table 3: Compared with SLAM systems based on ORB-SLAM3, the RMSE of t.RPE

Sequence	t.RPE (RMSE)			
	O3	RDS	CDS	OURS
W/half	0.0195	0.0274	0.0187	**0.0183**
W/rpy	0.0533	**0.0245**	0.0352	0.0374
W/static	0.0251	0.0221	**0.0064**	0.0081
W/xyz	0.0267	0.0269	0.0176	**0.0174**
S/static	0.0128	0.0050	0.0053	**0.0051**
S/xyz	0.0081	0.0113	0.0121	**0.0079**

Table 4: Compared with other advanced SLAM systems, the RMSE of ATE

Sequence	ATE (RMSE)					
	DS	Dyna	Crowd	YOLO	SG	OURS
W/half	0.0303	0.0289	0.026	0.0283	0.0268	**0.0243**
W/rpy	0.4442	**0.0298**	0.044	0.2164	0.0324	0.0345
W/static	0.0081	0.0071	0.007	0.0073	0.0073	**0.0068**
W/xyz	0.0247	0.0176	0.020	**0.0146**	0.0152	0.0156
S/static	0.0065	0.0058	0.008	0.0066	0.0060	**0.0053**
S/xyz	0.0084	0.0136	0.018	–	0.0093	**0.0072**

Table 5: Compared with other advanced SLAM systems, the RMSE of t.RPE

Sequence	t.RPE (RMSE)					
	DS	Dyna	Crowd	YOLO	SG	OURS
W/half	0.0297	0.0273	0.037	0.0268	0.0279	**0.0183**
W/rpy	0.1503	**0.0236**	0.065	0.0933	0.0450	0.0374
W/static	0.0102	**0.0079**	0.010	0.0094	0.0100	0.0081
W/xyz	0.0333	0.0242	0.025	0.0198	0.0191	**0.0174**
S/static	0.0078	0.0057	0.010	0.0089	0.0075	**0.0051**
S/xyz	0.0092	0.0145	0.020	–	0.0084	**0.0079**

Fig. 5 shows the processing details of different RGB-D frames by our proposed algorithm during system operation. The clustering module effectively segments dynamic object contours, benefiting from the improved adaptive threshold K-means++ algorithm. Clustering is performed only within the detection box area, as shown in the third row, allowing accurate segmentation even of small dynamic objects. The number of clusters is determined based on prior knowledge of dynamic point depth information, followed by post-processing to remove noise. For instance, in the first row of frames, the dynamic object and the chair are initially classified together, and in the second and fourth rows, the dynamic object is grouped with the table edge. Our method filters these out, leaving cleaner contours of dynamic objects. Finally, after obtaining semantic priors, the dynamic point removal module divides the scene into three regions and filters dynamic feature points based on probability and motion constraints. In the second row, static feature points are retained on the stationary person on the left, while all feature points on the moving person in black on the right are marked as dynamic and correctly removed during system tracking. Fig. 6 illustrates the error between the ground truth trajectory and the estimated trajectories by ORB-SLAM3, DynaSLAM, CDS-SLAM, and DKP-SLAM, showing that ORB-SLAM3 deviates noticeably from the ground truth, whereas DKP-SLAM closely matches the actual trajectory.

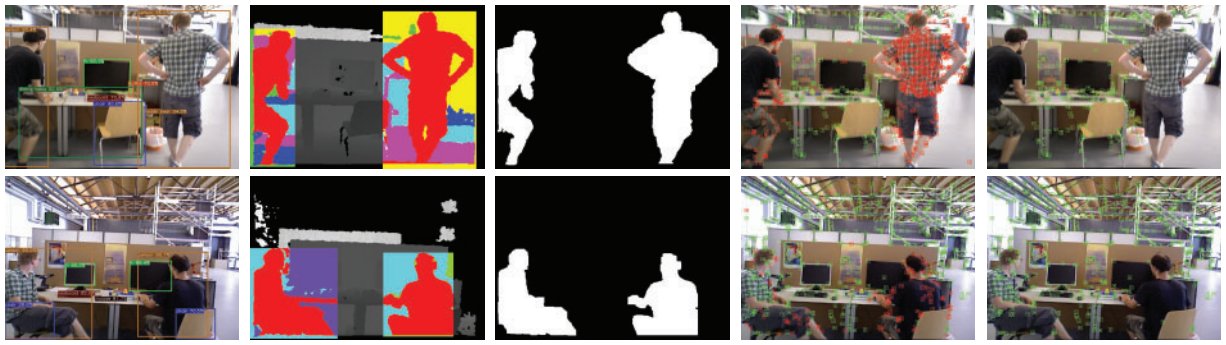


Figure 5: (Continued)

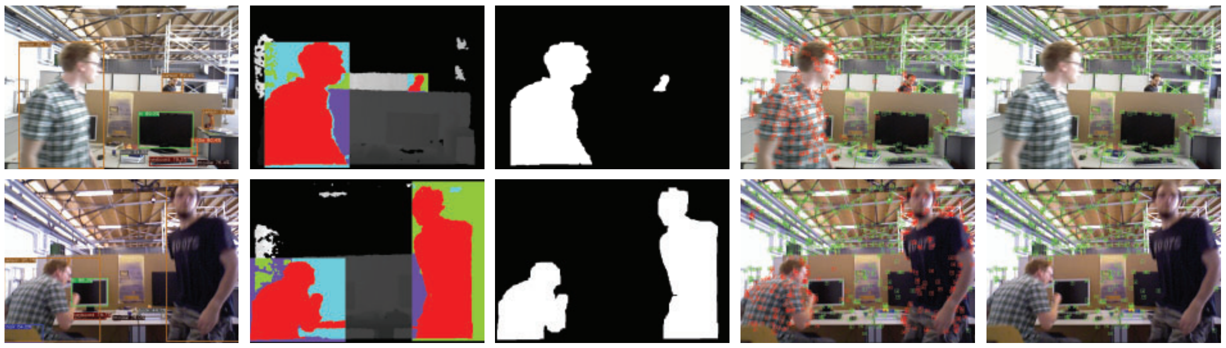


Figure 5: Example of system operation diagram

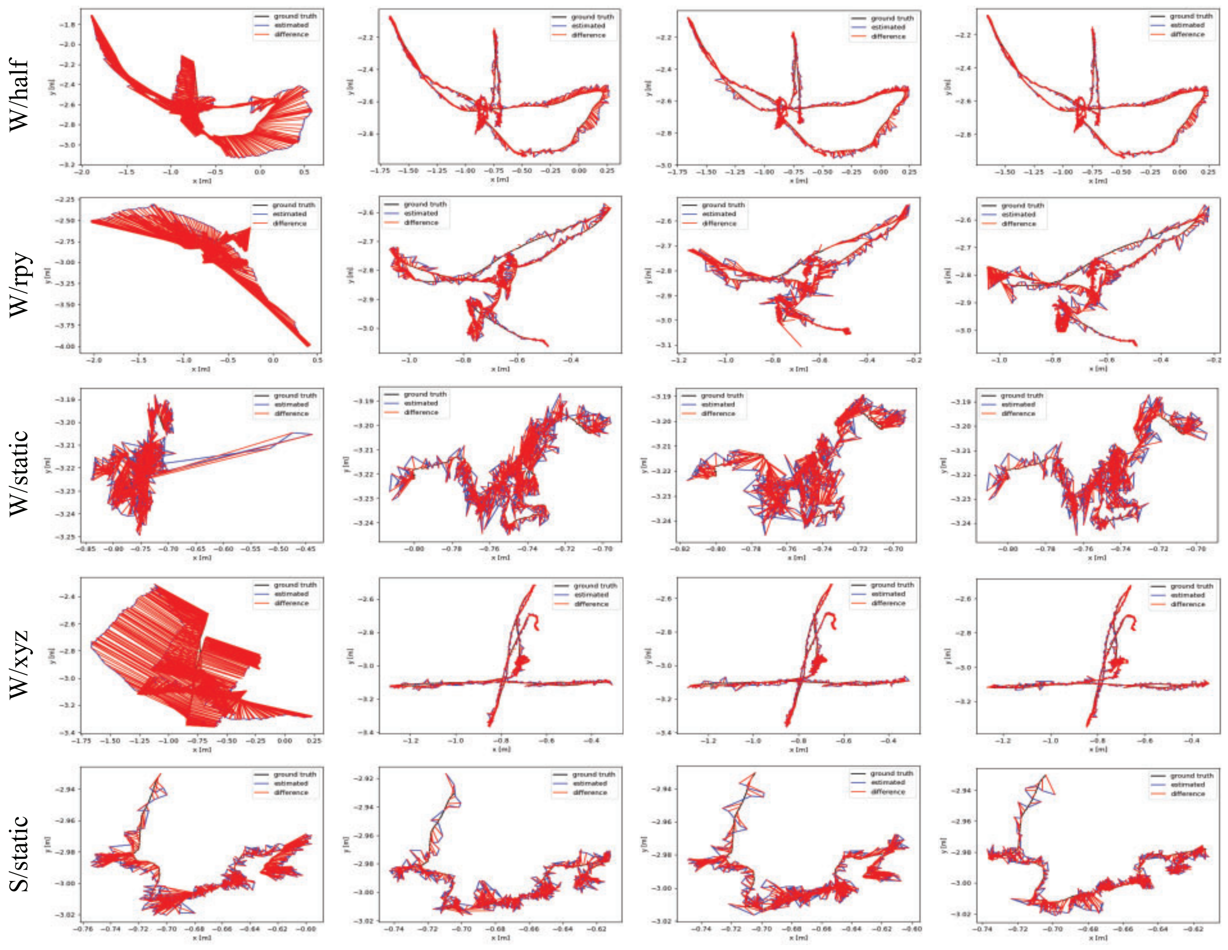


Figure 6: (Continued)

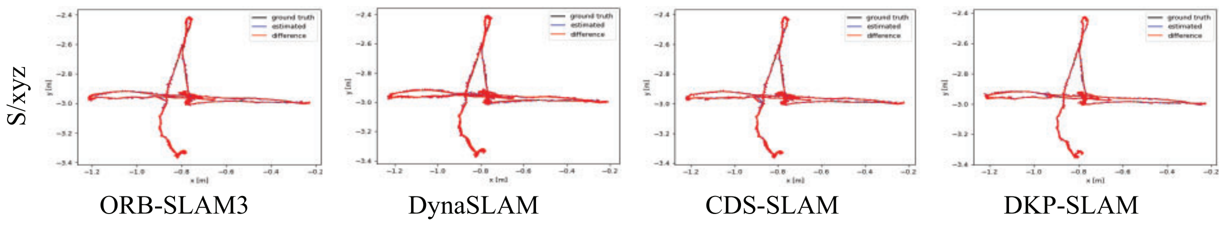


Figure 6: Comparison of ORB-SLAM3, DynaSLAM, CDS-SLAM, and DKP-SLAM trajectories. The figure shows the ground truth trajectory (black), estimated trajectory (blue), and the difference (red) between the estimated and ground truth trajectories. The x and y axes represent the position in meters on a 2D plane

4.3 Ablation Experiment

In this section, we conducted ablation experiments on DKP-SLAM to analyze the impact of each module, using ATE (RMSE) and standard deviation (S.D) metrics. We incrementally added modules for experimental analysis. Specifically, in the D-SLAM module, YOLOX is used to detect image frames, remove all feature points within dynamic object detection boxes, and perform missed detection compensation. DK-SLAM builds on D-SLAM by integrating an improved clustering module to eliminate all feature points within the masks. Finally, we tested the proposed DKP-SLAM. The experiments were structured into the above three parts for testing.

As shown in Table 6, the experimental results indicate that for all highly dynamic sequences prefixed with "W," D-SLAM outperforms DK-SLAM. Although clustering refines dynamic object detection and retains more static feature points, the presence of more dynamic feature points and their greater motion amplitude in the four highly dynamic sequences we selected means dynamic points have a larger impact than static points. Therefore, removing more dynamic points within the detection boxes results in higher accuracy compared to mask-based methods. Specifically, in the W/static sequence, D-SLAM and DK-SLAM exhibit similar accuracy, indicating a balance between the influence of dynamic and static points on the system.

Table 6: Comparison of absolute trajectory error ATE of ablation experiment

Sequence	D-SLAM		DK-SLAM		DKP-SLAM	
	RMSE	S.D	RMSE	S.D	RMSE	S.D
W/half	0.0293	0.0134	0.0316	0.0141	**0.0243**	**0.0105**
W/rpy	0.0632	0.0435	0.0841	0.0523	**0.0345**	**0.0321**
W/static	0.0082	0.0044	0.0085	0.0045	**0.0068**	**0.0037**
W/xyz	0.0184	0.0103	0.0228	0.0116	**0.0156**	**0.0073**
S/static	0.0090	0.0058	0.0076	0.0053	**0.0053**	**0.0045**
S/xyz	0.0173	0.0083	0.0121	0.0074	**0.0072**	**0.0057**

In contrast, in the "S/static" and "S/xyz" sequences, DK-SLAM outperforms D-SLAM. In these sequences, the dynamic objects are in a sitting posture, and more refined masks can retain more static feature points for camera estimation. DKP-SLAM, by adding a dynamic point removal strategy, achieved the highest accuracy across all sequences. By combining regional probability and geometric methods to differentiate feature points, it effectively distinguishes which areas of dynamic objects are moving and which feature points can be retained. Retaining more static feature points thus led to the highest pose estimation accuracy.

4.4 Real Time Analysis

In the practical applications of mobile robots, system real-time capability is crucial. As shown in Table 7, we tested the processing time of different modules in DKP-SLAM. Module A represents the YOLOX detection module; B represents the improved K-means++ module; and C represents the dynamic point removal module. The results show that our modules (A and B) obtained accurate dynamic object contours in a very short time, increasing the track time by 20.72 milliseconds compared to ORB-SLAM3.

Table 7: Average running time of different modules

Sequence	A (ms)	B (ms)	C (ms)	Tracking (ms)
ORB-SLAM3	/	/	/	21.65
DKP-SLAM	11.62	8.12	6.33	42.37

In Table 8, we also recorded the processing times of other systems. DynaSLAM, due to its pixel-level semantic segmentation, incurs a high time cost per frame and cannot achieve real-time performance. Similarly, RDS-SLAM and DS-SLAM, which also use semantic segmentation model, face speed limitations. SG-SLAM and CDS-SLAM, on the other hand, accelerate their object detection model using the NCNN and TensorRT platforms, respectively, resulting in faster system operation. Additionally, while YOLO-SLAM uses an end-to-end object detection model, it did not employ GPU acceleration in the experiments, resulting in a high per-frame processing time of 696.09 milliseconds. DKP-SLAM also employs TensorRT for GPU acceleration of the object detection model and performs clustering only within the detection boxes, resulting in a total processing time of 42.37 milliseconds per frame, thereby meeting the real-time requirements for robot operation.

Table 8: Time analysis of different systems

Algorithm	Average processing (ms)	Experimental Platform
CDS-SLAM	37.96	Ryzen7-5800H CPU, RTX 3070 GPU
RDS-SLAM	57.5	RTX 2080Ti GPU
DS-SLAM	58.4	RTX 1060 GPU
DynaSLAM	192	Tesla M40 GPU
SG-SLAM	39.51	Ryzen7-4800H CPU, GTX 1650 GPU
YOLO-SLAM	696.09	Inter i5-4288U CPU
DKP-SLAM	42.37	Inter i7-11700 CPU, RTX 2080Ti GPU

5 Conclusion

This paper proposes a real-time DKP-SLAM algorithm for dynamic scenes. Based on ORB-SLAM3, DKP-SLAM deploys the YOLOX model through TensorRT to obtain semantic priors and implements compensation measures for missed detections. The K-means++ algorithm in the clustering module is then improved to obtain more accurate dynamic object contours. Finally, in our dynamic point removal module, the current frame is divided into multiple regions, with dynamic probability weights assigned to feature points in different regions. By combining geometric methods, our algorithm effectively removes dynamic feature points to reduce the interference of dynamic objects on the SLAM system. Experiments conducted on the TUM dataset compared our method with other advanced SLAM systems, demonstrating excellent localization accuracy in many sequences and showcasing its effectiveness in dynamic scenes. Future work will consider more precise probability allocation for regional feature points and address the issue of inaccurate semantic information caused by camera rotation.

Acknowledgement: The authors are highly thankful to the National Natural Science Foundation of China, to the Guangxi Natural Science Foundation under Grant, to the Innovation Fund of Chinese Universities Industry-University-Research, to the Special Research Project of Hechi University. Guangxi Colleges and Universities Key Laboratory of AI and Information Processing (Hechi University), and to the Central Foreign Trade and Economic Development Special Project, Education Department of Guangxi Zhuang Autonomous Region.

Funding Statement: The authors are highly thankful to the National Natural Science Foundation of China (No. 62063006), to the Guangxi Natural Science Foundation under Grant (Nos. 2023GXNSFAA026025, AA24010001), to the Innovation Fund of Chinese Universities Industry-University-Research (ID: 2023RY018), to the Special Guangxi Industry and Information Technology Department, Textile and Pharmaceutical Division (ID: 2021 No. 231), to the Special Research Project of Hechi University (ID: 2021GCC028), and to the Key Laboratory of AI and Information Processing, Education Department of Guangxi Zhuang Autonomous Region (Hechi University), No. 2024GXZDSY009.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Menglin Yin, Yong Qin; data collection: Menglin Yin, Jiansheng Peng; analysis and interpretation of results: Jiansheng Peng, Yong Qin. draft manuscript preparation: Menglin Yin, Yong Qin. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, 2021. doi: 10.1109/TRO.2021.3075644.

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," presented at the Eur. Conf. Comput. Vis., Cham, Switzerland, Sep. 6–12, 2014, pp. 834–849.

[3] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, 2018. doi: 10.1109/TRO.2018.2853729.

[4] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981. doi: 10.1145/358669.358692.

[5] J. C. V. Soares, M. Gattass, and M. A. Meggiolaro, “Crowd-SLAM: Visual SLAM towards crowded environments using object,” *J. Intell. Robot. Syst.*, vol. 102, no. 1, 2021, Art. no. 50. doi: 10.1007/s10846-021-01414-1.

[6] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, “YOLOX: Exceeding YOLO series in 2021,” 2021. doi: 10.48550/arXiv.2107.08430.

[7] Nvidia, “TensorRT,” 2019. Accessed: Aug. 21, 2024. [Online]. Available: https://developer.nvidia.com/tensorrt

[8] A. Kundu, K. M. Krishna, and J. Sivaswamy, “Moving object detection by multi-view geometric techniques from a single camera mounted robot,” presented at the IEEE/RSJ Int. Conf. Intell. Robots Syst., St.Louis, MO, USA, Oct. 11–15, 2009, pp. 4306–4312.

[9] S. Li and D. Lee, “RGB-D SLAM in dynamic environments using static point weighting,” *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2263–2270, 2017. doi: 10.1109/LRA.2017.2724759.

[10] W. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, “RGB-D SLAM in dynamic environments using point correlations,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 373–389, 2020. doi: 10.1109/TPAMI.2020.3010942.

[11] R. Wang, W. Wan, Y. Wang, and K. Di, “A new RGB-D SLAM method with moving object detection for dynamic indoor scenes,” *Remote Sens.*, vol. 11, no. 10, 2019, Art. no. 1143. doi: 10.3390/rs11101143.

[12] D. Nguyen, C. Hughes, and J. Horgan, “Optical flow-based moving-static separation in driving assistance systems,” presented at the IEEE Int. Conf. Intell. Transp. Syst., Las Palmas, Spain, Sep. 15–18, 2015, pp. 1644–1651.

[13] Y. Sun, M. Liu, and M. Q. H. Meng, “Motion removal for reliable RGB-D SLAM in dynamic environments,” *Robot Auton. Syst.*, vol. 108, no. 4–5, pp. 115–128, 2018. doi: 10.1016/j.robot.2018.07.002.

[14] J. Cheng, Y. Sun, and M. Q. H. Meng, “Improving monocular visual SLAM in dynamic environments: An optical-flow-based approach,” *Adv. Robot.*, vol. 33, no. 12, pp. 576–589, 2019. doi: 10.1080/01691864.2019.1610060.

[15] C. Yu *et al.*, “DS-SLAM: A semantic visual SLAM towards dynamic environments,” presented at the IEEE/RSJ Int. Conf. Intell. Robots Syst., Madrid, Spain, Oct. 1–5, 2018, pp. 1168–1174.

[16] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, 2017. doi: 10.1109/TPAMI.2016.2644615.

[17] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras,” *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017. doi: 10.1109/TRO.2017.2705103.

[18] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, “DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes,” *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, 2018. doi: 10.1109/LRA.2018.2860039.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” presented at the IEEE Int. Conf. Comput. Vis., Venice, Italy, Oct. 22–29, 2017, pp. 2961–2969.

[20] Y. Liu and J. Miura, “RDS-SLAM: Real-time dynamic SLAM using semantic segmentation methods,” *IEEE Access*, vol. 9, pp. 23772–23785, 2021. doi: 10.1109/ACCESS.2021.3050617.

[21] H. Zhang, J. Peng, and Q. Yang, “PR-SLAM: Parallel real-time dynamic SLAM method based on semantic segmentation,” *IEEE Access*, vol. 12, no. 3, pp. 36498–36514, 2024. doi: 10.1109/ACCESS.2024.3373308.

[22] Q. Yang, J. Peng, D. Chen, and H. Zhang, “Road scene instance segmentation based on improved SOLOv2,” *Electronics*, vol. 12, no. 19, 2023, Art. no. 4169. doi: 10.3390/electronics12194169.

[23] W. Wu, L. Guo, H. Gao, Z. You, Y. Liu and Z. Chen, “YOLO-SLAM: A semantic SLAM system towards dynamic environment with geometric constraint,” *Neural Comput. Appl.*, vol. 34, no. 8, pp. 6011–6026, 2022. doi: 10.1007/s00521-021-06764-3.

[24] Q. Zhang and C. Li, “Semantic SLAM for mobile robots in dynamic environments based on visual camera sensors,” *Meas. Sci. Technol.*, vol. 34, no. 8, 2023, Art. no. 085202. doi: 10.1088/1361-6501/acd1a4.

[25] S. Cheng, C. Sun, S. Zhang, and D. Zhang, "SG-SLAM: A real-time RGB-D visual SLAM toward dynamic scenes with semantic and geometric information," *IEEE Trans. Instrum. Meas.*, vol. 72, no. 3, pp. 1–12, 2022. doi: 10.1109/TIM.2022.3228006.

[26] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu and J. Song, "Semantic SLAM based on object detection and improved octomap," *IEEE Access*, vol. 6, pp. 75545–75559, 2018. doi: 10.1109/ACCESS.2018.2873617.

[27] C. Ge, Y. Song, C. Ma, Y. Qi, and P. Luo, "Rethinking attentive object detection via neural attention learning," *IEEE Trans. Image Process.*, vol. 33, pp. 1726–1739, 2023. doi: 10.1109/tip.2023.3251693.

[28] W. Qian, J. Peng, and H. Zhang, "DFD-SLAM: Visual SLAM with deep features in dynamic environment," *Appl. Sci.*, vol. 14, no. 11, 2024, Art. no. 4949. doi: 10.3390/app14114949.

[29] X. Zhang and Z. Shi, "MDP-SLAM: A visual SLAM towards a dynamic indoor scene based on adaptive mask dilation and dynamic probability," *Electronics*, vol. 13, no. 8, 2024, Art. no. 1497. doi: 10.3390/electronics13081497.

[30] G. Jocher, A. Chaurasia, and J. Qiu, "YOLOv8," 2023. Accessed: Sep. 25, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[31] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," presented at the Eur. Conf. Comput. Vis., Zurich, Switzerland, Sep. 6–12, 2014, pp. 740–755.

[32] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," presented at the IEEE/RSJ Int. Conf. Intell. Robots Syst., Vilamoura, Portugal, Oct. 7–12, 2012, pp. 573–580.