ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

Visual SLAM Algorithm based on Dynamic Feature Point Filtering and Motion Probability Propagation

Wenbo Bai[1,2], Haiyun Gan[1,2,*], Jun Li[1,2], Jialin Wang[1,2]

[1] School of Automotive and Transportation, Tianjin University of Technology and Education, Tianjin, China

[2] National and Local Joint Engineering Center for Intelligent Vehicle-Road Cooperation and Safety Technology, Tianjin, China

[*]Corresponding author: ganhaiyun@aliyun.com

Abstract

To effectively filter out dynamic feature points in dynamic scenes and thus improve the localization accuracy of visual SLAM algorithms, a visual SLAM algorithm based on dynamic feature point filtering and motion probability propagation was proposed. This algorithm built upon ORB-SLAM3 by adding a new detection thread. This detection thread employed the Detectron2 algorithm for object detection and instance segmentation on keyframes. Subsequently, in the tracking thread, the motion probability of feature points was updated through feature matching and matched point propagation. After propagating the motion probability for each point, all dynamic points were removed. Experimental results showed that on the TUM dynamic scene fr3/walking_xyz dataset sequence, compared with ORB-SLAM3, the proposed algorithm reduced the average total time consumption in the local mapping thread by 13.39%, reduced the average time consumption in local BA optimization by 23.52%, and reduced the root mean square error (RMSE) of absolute trajectory error by 94.7%. The proposed algorithm ensures real-time performance while effectively filtering out dynamic feature points, thus enhancing the localization accuracy and precision of the SLAM algorithm.

Keywords

Dynamic Scene; ORB-SLAM3; Keyframe; Detectron2.

1. Introduction

Precise localization is the foundation for achieving autonomous movement, and the prerequisite for precise localization is that the robot can build an accurate map. The accuracy of the map directly determines the robot's localization accuracy, which in turn affects its navigation and decision-making accuracy. Therefore, the development of Simultaneous Localization and Mapping (SLAM) technology is crucial for enhancing the robot's autonomous movement capabilities. Traditional SLAM methods mainly rely on sensors like LiDAR to acquire environmental information, but these methods may face limitations in complex environments or scenes with significant lighting variations. In recent years, with the advancement of computer vision technology, Visual SLAM (vSLAM) has gradually become a research hotspot. Visual SLAM systems use image data captured by cameras and perform real-time analysis and processing of this data to achieve real-time localization and mapping, making it the current primary direction of SLAM technology development [1-2]. Standard vSLAM systems are primarily designed for static or slowly changing environments. However, when deployed in non-static, complex environments (such as busy city streets or crowded indoor spaces), they face

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

a series of challenges. The most notable of these challenges is the inaccurate judgment of dynamic/static objects, which can significantly impact the system's performance.

To address the aforementioned issues, researchers have proposed various methods and techniques [3-5]. In 2017, Rünz and Agapito introduced the Co-Fusion system [6], which effectively distinguished the foreground and background in images using multi-model fitting techniques, allowing moving objects to be tracked independently of the background. AN Lifeng et al. [7] combined direct methods with feature-based methods by calculating the total projection error of specific object pixels between consecutive image frames and incorporating semantic information to optimize camera pose and reduce the impact of dynamic objects on tracking accuracy. In 2018, YU Chao from Tsinghua University proposed the DS-SLAM system [8], which innovatively combined semantic information with dynamic feature point detection. By filtering out dynamic objects in keyframes, the system significantly improved the accuracy of camera pose estimation. Bescós B and colleagues developed the Dyna SLAM system [9], which optimized the ORB-SLAM2 system. It utilized instance-level semantic segmentation to accurately identify and remove dynamic objects from keyframes while simultaneously repairing the background, thus greatly enhancing localization accuracy and mapping in dynamic environments. In 2019, Xiao et al. introduced a monocular camera-based Dynamic-SLAM system [10], which integrated the object detection network SSD to identify moving objects. To address the low recall rate of SSD in detecting moving objects, they proposed a missed detection compensation algorithm based on the constant velocity of adjacent frames, significantly improving the detection recall rate. Wang et al. [11] proposed a semantic SLAM system for dynamic object environments, using the object detection network YOLOv3 [12] to detect specific moving objects. They employed a depth-map-based overspill filling algorithm to extract contours from the detection results, achieving high-precision semantic segmentation. In 2020, Zhang et al. introduced VDO-SLAM [13], a feature-based stereo/RGB-D dynamic SLAM system capable of outputting trajectories of dynamic objects and pose estimates over time, as well as extracting the velocity information of dynamic objects. VDO-SLAM is a highly robust dynamic object-aware SLAM system that can identify dynamic regions through motion estimation, even when the shape of the target object is unknown.

In summary, the common approach for handling dynamic objects in vSLAM within dynamic environments involves using deep learning for object detection to identify dynamic feature points, then adding semantic labels to the corresponding keyframes, followed by feature point tracking [14-16]. However, this method has a significant drawback-prolonged computation time, which makes it challenging to meet real-time system requirements. Additionally, camera shake and rotation during movement can affect the quality of keyframe selection, leading to tracking failure.

To address these issues, this paper proposes the following strategies for effective solutions:

a) Optimizing the keyframe selection strategy, specifically for scenarios involving camera shake and rotation.

b) Detecting objects only in keyframes, then updating the movement probability of feature points in the local map. During the tracking process, the movement probability is propagated through feature matching and matching point expansion, effectively removing feature points extracted from moving objects before camera pose estimation.

2. System Architecture

ORB-SLAM3 [17] mainly consists of three threads: the Tracking thread, the Local Mapping thread, and the Loop Closing & Map Merging thread. The SLAM algorithm in this paper introduces an additional detection thread based on ORB-SLAM3, with the specific algorithm framework shown in Figure 1. By optimizing the keyframe selection strategy and calculating and updating the movement probability of feature points in the local map, this method ensures both the localization accuracy and real-time performance of the SLAM algorithm.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

This paper primarily focuses on the research of keyframe selection strategies and the handling of movement probabilities.

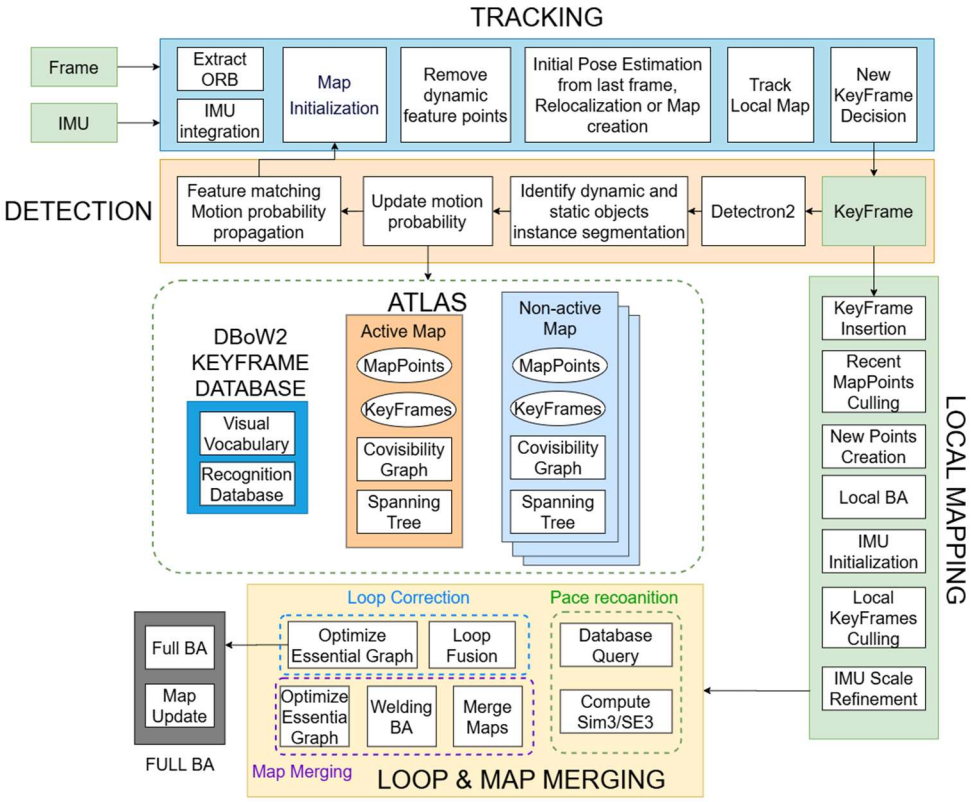


Fig. 1 Overall framework of SLAM algorithm

2.1 Optimization of Keyframe Selection Strategy.

Keyframes are selected from a series of ordinary frames to represent the most significant frame. Keyframes mainly serve the following purposes [18]:

a) Keyframes can reduce information redundancy.

b) Keyframes can decrease the computational burden and reduce error accumulation.

c) Keyframes can ensure the quality of images.

The selection of keyframes plays a crucial role in SLAM algorithms. By reasonably selecting keyframes, it can ensure that the SLAM system achieves high accuracy and efficiency in positioning and map construction in complex environments. If the camera's movement and rotation are too large in a short period and keyframes are not inserted in a timely manner, it may lead to tracking loss and reduced positioning accuracy. Therefore, this paper sets the following keyframe selection strategies for the special scenarios of camera jitter and rotation:

a) The minimum number of map points tracked by the keyframe is set to 50.

b) The proportion of current keyframe map points that belong to the previous keyframe is less than 75%.

c) Use IMU data to detect the extent of camera jitter and rotation, setting minimum and maximum thresholds. If the rotation falls within this range, the current frame can be designated as a keyframe.

d) Calculate the relative motion amount \(D\) between adjacent frames, setting minimum and maximum thresholds. If the relative motion amount falls within this range, the current frame can be designated as a keyframe.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

Regarding how to calculate the relative motion amount between adjacent frames in point d), the calculation process is provided below:

When the camera is monocular, we only know the 2D pixel coordinates; thus, the problem is to estimate the motion based on two sets of 2D points. This problem is solved using epipolar geometry. The calculation process is as follows: first, assume that the feature points of the current keyframe and the previous frame have been matched. The feature points of the current frame $p=\{p_1,p_2,p_3\cdots p_n\}$ and the feature points of the previous keyframe $p'=\{p_1',p_2',p_3'\cdots p_n'\}$ are used to construct a least squares problem to solve for the rotation matrix R and the translation vector t.

Let the spatial position of P be: $P_i=[X_i,Y_i,Z_i]^T$

According to the pinhole camera model, the pixel positions of the two points p_i and $p_i^{'}$ are given by:

$$s_i p_i = KP_i, s_i p_i^{'} = K(RP_i + t) \tag{1}$$

Here, K is the camera intrinsic parameter matrix, and R and t represent the camera motion between the two coordinate systems. We use homogeneous coordinates to represent the pixel points. The points $s_i p_i$ and p_i have a projection relationship, and they are equal in the sense of homogeneous coordinates. Thus, the above formula can be written as:

$$p_i = KP_i, p_i^{'} = K(RP_i + t) \tag{2}$$

After rearranging, we obtain:

$$p_i^{'T} K^{-T} t^{\wedge} RK^{-1} p_i = 0 \tag{3}$$

This equation is known as the epipolar constraint, which simultaneously includes both translation and rotation. We denote the middle part as two matrices: the Fundamental Matrix (F) and the Essential Matrix (E), further simplifying the epipolar constraint [19]:

$$E = t^{\wedge} R, F = K^{-T} EK^{-1}, p_i^{'T} Fp_i = 0 \tag{4}$$

Next, we perform Singular Value Decomposition (SVD) on the Essential Matrix (E):

$$E = U\Sigma V^T \tag{5}$$

Where U and V are orthogonal matrices, and Σ is the singular value matrix. For any given E , there exist two possible pairs of R and t that correspond to it:

$$t_1^{\wedge} = UR_Z(\frac{\pi}{2})\Sigma U^T, R_1 = UR_Z^T(\frac{\pi}{2})V^T \tag{6}$$

$$t_2^{\wedge} = UR_Z(-\frac{\pi}{2})\Sigma U^T, R_2 = UR_Z^T(-\frac{\pi}{2})V^T \tag{7}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

Where $R_Z(\frac{\pi}{2})$ indicates the rotation matrix obtained by rotating 90° around the Z-axis. When decomposing E into R and t , there are a total of four possible solutions, but only one solution has positive depth in both cameras. Therefore, by substituting any point into the four solutions and checking the depth of that point in both cameras, we can determine which solution is the correct one.

At this point, the relative motion amount D can be calculated using the rotation matrix (R) and translation vector (t):

$$D = \alpha \| \Delta t \| + \beta(\min(2\pi - \| R \|, \| R \|)) \tag{8}$$

Where α is the weight coefficient for translation, and β is the weight coefficient for rotation, with the camera rotation angle serving as the primary reference variable for change. The computed relative motion amount is then compared with the set threshold:

$$\begin{cases} F_{key} = F_{cur}, D_{\min} \le D \le D_{\max} \\ F_{key} \ne F_{cur}, D \le D_{\min} \text{或} D > D_{\max} \end{cases} \tag{9}$$

Where F_{key} is the keyframe and F_{cur} is the current frame.

Through the optimization of the keyframe selection strategy described above, new keyframes can be inserted promptly when the camera's movement and rotation amplitudes are excessive, effectively reducing tracking loss and improving positioning and mapping accuracy.

2.2 Keyframe Processing

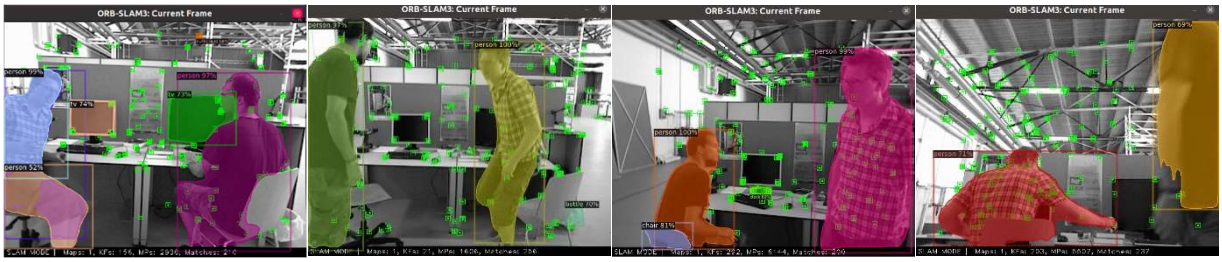
Detectron2 is a powerful deep learning library developed by Facebook AI Research, specifically designed for object detection and instance segmentation tasks. Detectron2 includes a wide range of state-of-the-art object detection and instance segmentation models, including Faster R-CNN, Mask R-CNN, and RetinaNet. Users can choose the appropriate model based on their needs.

In terms of object detection, YOLO offers fast detection speeds but relatively lower accuracy, especially performing poorly in detecting small objects and in complex scenes. On the other hand, Detectron2 features a modular design that supports various network architectures and tasks, achieving high-precision object detection and segmentation, particularly excelling in complex scenes and multitasking. Given Detectron2's advantages in object detection and the requirements for accuracy and real-time performance in this paper's algorithm, we choose to use the Detectron2 algorithm for object detection and instance segmentation on keyframes in the detection thread. The specific detection results are shown in Figure 2.

Figure 2(a) shows the original RGB images, with randomly selected pictures from the TUM dataset sequences: walking_xyz, walking_static, walking_rpy, and walking_halfsphere, arranged from left to right. Figure 2(b) illustrates the results of the algorithm's object detection and instance segmentation, with the detection results and recognition rates annotated in the images. It also performs instance segmentation on moving objects and some static objects. When static and moving objects overlap, it accurately segments the dynamic objects. Points within the moving object regions are treated as high-confidence dynamic points, which then update their movement probabilities. These dynamic points are subsequently removed in the tracking thread, enhancing the camera's pose estimation and SLAM localization accuracy.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

(a) Original RGB image



(b) Object detection and instance segmentation results

Fig. 2 Object detection and instance segmentation of keyframes

2.3 Movement Probability

In this paper, the probability that a feature point belongs to a moving object is referred to as the movement probability [20]. As shown in Figure 3, the feature points are classified into four states based on their movement probability. In the process of extending matching points, two high-confidence points are used: high-confidence static points and high-confidence dynamic points, after which their movement probability is propagated to neighboring unmatched points. Once the movement probability for each point is obtained through propagation, all dynamic points are removed, and RANSAC is applied to filter out other outliers for pose estimation.

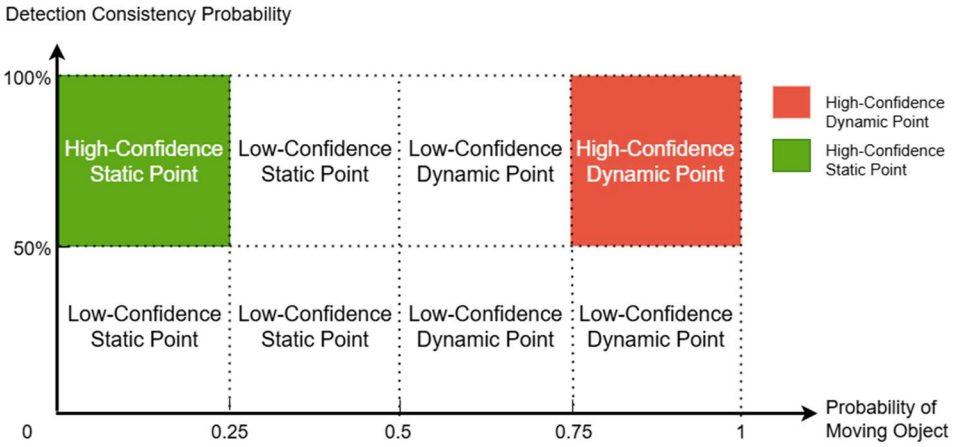


Fig. 3 The four states of feature points

2.3.1 Updating Movement Probability

Considering detection delays and the spatiotemporal consistency of consecutive frames, we only select keyframes for detection. An initial movement probability is assigned to the initially extracted feature points in the keyframe. Since there is no prior information to detect which state these points belong to, the initial movement probability is set to 0.5. Once the detection results are obtained, the keyframe is inserted into the local map, and the movement probability in the local map is updated. Then, the image is preprocessed and forward-propagated through a deep neural network, while the movement probability is propagated frame by frame in the tracking thread.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

2.3.2 Movement Probability Propagation

Movement probability propagation refers to estimating the movement probability of each feature point frame by frame in the tracking thread through two operations: feature matching and matching point extension. Since the movement probability of the feature points in the current frame is propagated from the feature points in the previous frame, no prior detection is required.

2.3.3 Feature Matching

Based on the robustness and efficiency of ORB feature matching [21], we use the feature matching of the ORBSLAM3 algorithm. During the feature matching process, once a feature point x_t^i from the last frame successfully matches another feature point x_{t-1}^i, the movement probability $P_t(x_{t-1}^i)$ is immediately propagated. Additionally, if a feature point matches any 3D point X_t^i in the local map, it is also assigned a movement probability, which equals the value of the matched point. If a point finds a match not only in the last frame but also in the local map, the probability from the local map is prioritized.

2.3.4 Matching Point Extension

Matching point extension refers to propagating the movement probability of high-confidence feature points to neighboring points that do not correspond to any matched point in the feature matching operation. This extension relies on the assumption that the state of neighboring points is consistent in most cases. After propagating through feature matching, we select high-confidence points χ_t, including static and dynamic points. Then, the influence area of the high-confidence points is extended into a circular region with a radius of r, and unmatched points within the region are searched for. The movement probability of the found feature points is updated according to the following formula:

$$P(x_t^j) = P_{init} + \sum_{x_t^j \in \chi_t} \lambda(d)(P(x_t^j) - P_{init}) \tag{10}$$

where P_{init} represents the initial movement probability. If a point is influenced by multiple high-confidence points, the effects from all adjacent high-confidence points are summed. The influence of high-confidence points mainly considers the movement probability difference $P^m(x_t^j) - P_{init}$ and the distance factor $\lambda(d)$. If a point is within the influence area $(d \le r)$ of a high-confidence point, the distance factor is $\lambda(d) = Ce^{-d/r}$, a constant C; otherwise, $\lambda(d) = 0$ it is zero. The value of r depends on the image resolution, scene complexity, feature point density, and real-time requirements of the system. In the TUM dataset, the image size is 640 pixels × 480 pixels, the experiment scene is dynamic, and the feature point density is moderate. Therefore, the initial value of the expansion radius r in this paper is set to 15 pixels.

2.3.5 Filtering Out Dynamic Feature Points

After target detection and instance segmentation, feature points that fall within the region of moving objects in the keyframe are considered high-confidence dynamic points, with their movement probability set to $P_t(x_t^i) \ge 0.75$. For feature points near the edges of moving objects and unmatched feature points, their movement probability is updated through feature point matching and matching point extension. Before camera pose estimation, dynamic feature points with a movement probability of $P_t(x_t^i) \ge 0.55$ are removed, and RANSAC is used to filter out other outliers for pose estimation. The actual effect of filtering out dynamic feature points is shown in Figure 8.

3. Experimental Results

The experimental system was run on Ubuntu 20.04, using Python version 3.8.18, with an NVIDIA GeForce GTX-3060Ti GPU. The test datasets were selected from four sequences in the TUM dataset: fr3/walking_xyz, fr3/walking_static, fr3/walking_rpy, and fr3/walking_halfsphere. Simulations were

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

conducted using both the ORB-SLAM3 algorithm and the algorithm proposed in this paper. Finally, the experimental results were compared and validated using the EVO evaluation tool.

3.1 Real-Time Performance Analysis

Real-time performance is a key evaluation criterion for SLAM algorithms. The ORB-SLAM3 code itself provides functionality for analyzing runtime performance. During SLAM execution, an ExecMean.txt file is generated, which records and analyzes the average execution time of each module. From the analysis of the ExecMean.txt file, it is evident that the tasks consuming the most time in the tracking thread and local mapping thread are local map optimization, map point creation, and ORB feature extraction. The algorithm proposed in this paper adds a detection thread that performs object detection and instance segmentation on keyframes and removes a large number of dynamic feature points in the tracking thread. As a result, it reduces the execution time of subsequent tasks such as feature point matching, local map optimization, and map point creation.

In this algorithm, the minimum keyframe interval is set to 10 frames. Since the RGB-D camera typically operates at about 30 FPS, the minimum time interval for selecting a keyframe is approximately 0.33 seconds. When slight jitter and rotation are detected during camera movement, the minimum keyframe interval is set to 5 frames, with the minimum time interval for selecting a keyframe being about 0.17 seconds. The detection thread, which uses the Detectron2 algorithm trained on the dataset, detects a keyframe in an average time of approximately 0.16 seconds, indicating that this detection thread meets the real-time performance requirements.

Table 1 records the total execution time of the tracking thread and the local mapping thread, as well as the time taken for ORB feature extraction and local BA optimization. The data was generated by running the fr3/walking_xyz dataset sequence with both the ORB-SLAM3 algorithm and the algorithm proposed in this paper. As shown in Table 1, the proposed algorithm shows a significant improvement in the total time consumption of the local mapping thread and the time taken for local BA optimization. Compared to the ORB-SLAM3 algorithm, the proposed algorithm reduces the average total time consumption of the local mapping thread by 13.39%, and the average time consumption of local BA optimization by 23.52%.

Table 1. The time consumption for running the fr3/walking_xyz dataset(ms)

Algorithm		1	2	3	Average
ORB-SLAM3	Tracking	13.79	14.02	13.85	13.89
	Local Mapping	63.01	62.65	67.78	64.48
	Local BA	44.61	44.95	49.11	46.22
Proposed Algorithm	Tracking	13.73	13.85	13.68	13.75
	Local Mapping	50.56	53.24	52.14	51.98
	Local BA	**34.23**	**36.76**	**35.06**	**35.35**

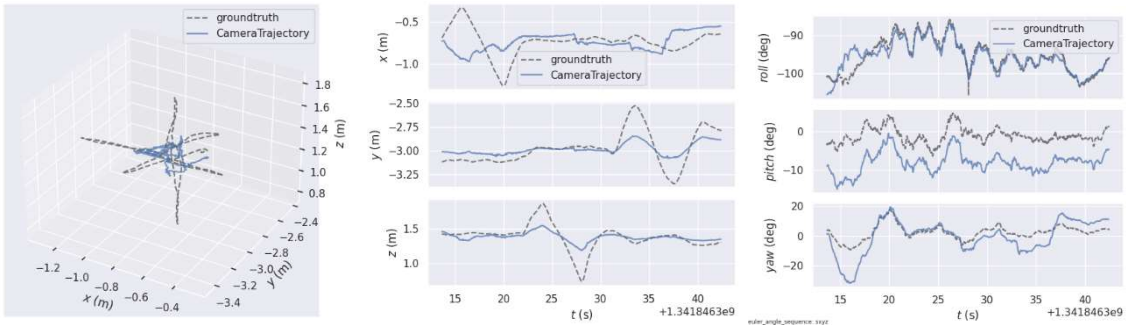
3.2 Localization Accuracy Analysis

Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) are two important metrics for evaluating the localization accuracy of SLAM algorithms. ATE measures the global error between the estimated trajectory and the ground truth trajectory by calculating the Euclidean distance between the estimated and actual poses. RPE evaluates the local consistency of the estimated trajectory by calculating the relative transformation error between consecutive poses. The ORB-SLAM3 algorithm and the proposed algorithm were run on four dataset sequences: fr3/walking_xyz, fr3/walking_static, fr3/walking_rpy, and fr3/walking_halfsphere.

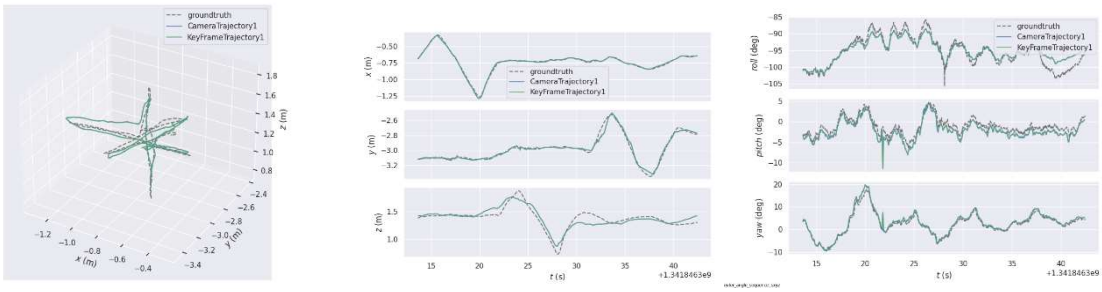
ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

Figure 4 shows the comparison of trajectory poses between the ORB-SLAM3 algorithm and the proposed algorithm when running the fr3/walking_xyz dataset. Figure 5 shows the Absolute Trajectory Error, and Figure 6 shows the Relative Pose Error. From Figures 4-6, it is evident that the estimated camera trajectory and pose errors are smaller with the proposed algorithm compared to the ORB-SLAM3 algorithm.

Tables 2 and 3 provide a comparison of the Absolute Trajectory Error and Relative Pose Error for the four dataset sequences mentioned above. The tables list four types of data: Root Mean Square Error (RMSE), Standard Deviation (S.D.), Mean, and Median. RMSE is the square root of the average of the squared errors, reflecting the overall deviation between the estimated and actual values and is used to assess the global error between the estimated and ground truth trajectories. From Table 2, it can be seen that in the fr3/walking_xyz dataset sequence, the RMSE value of the Absolute Trajectory Error is reduced by 94.7% with the proposed algorithm compared to the ORB-SLAM3 algorithm; in the fr3/walking_static dataset sequence, the RMSE value is reduced by 65.0%; in the fr3/walking_rpy dataset sequence, the RMSE value is reduced by 83.1%; and in the fr3/walking_halfsphere dataset sequence, the RMSE value is reduced by 84.1%. Table 4 provides a comparison of Absolute Trajectory Error between the proposed algorithm and other algorithms. The experimental data shows that in high-dynamic environments, these algorithms perform better than ORB-SLAM3. The proposed algorithm achieves the smallest Absolute Trajectory Error in the fr3/walking_xyz and fr3/walking_rpy dataset sequences, indicating that it can effectively handle such high-dynamic environments.



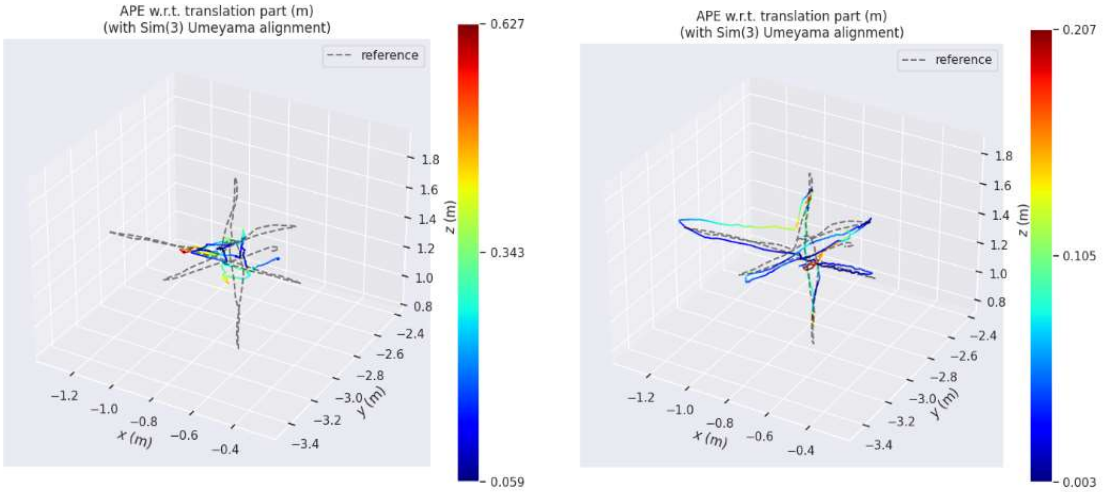
(a) The ORB-SLAM3 estimates the camera trajectory (b) xyz (c) rpy



(d) The proposed algorithm estimates the camera trajectory (e) xyz (f) rpy

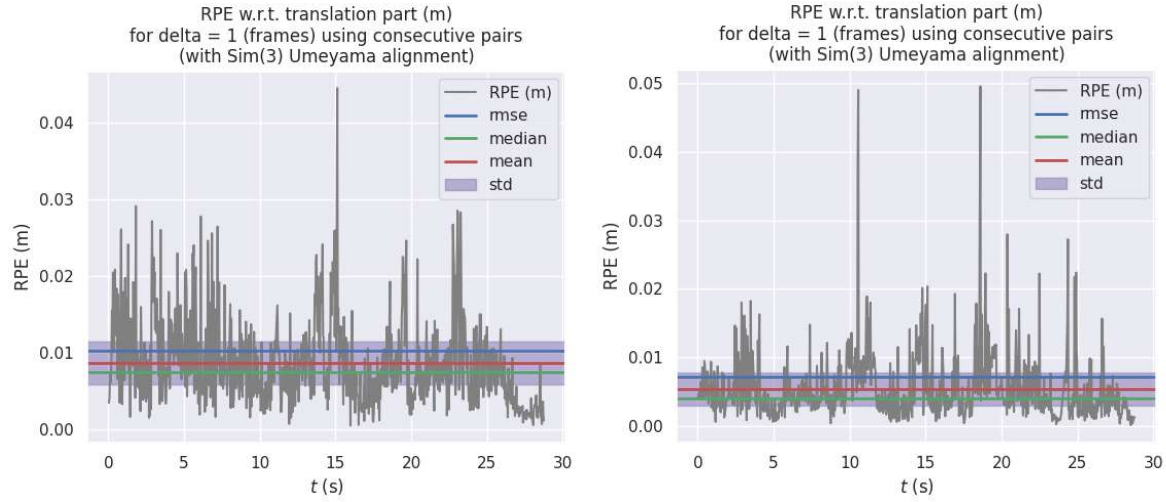
Fig. 4 The trajectory poses of the ORB-SLAM3 and the proposed algorithm on the walking_xyz sequence

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013



(a) Absolute trajectory error of ORB-SLAM3 (b) Absolute trajectory error of the proposed algorithm

Fig. 5 The absolute trajectory error of the ORB-SLAM3 and the proposed algorithm on the walking_xyz sequence



(a) Relative Pose Error of ORB-SLAM3 Algorithm (b) Relative Pose Error of the Proposed Algorithm

Fig. 6 The relative pose error of the ORB-SLAM3 and the proposed algorithm on the walking_xyz sequence

Table 2. Comparison of absolute trajectory error between ORB-SLAM3 and the proposed algorithm (m)

dataset sequence	ORB-SLAM3 algorithm				proposed algorithm			
	RMSE	S.D.	Mean	Median	RMSE	S.D.	Mean	Median
walking_xyz	0.2701	0.1306	0.2364	0.2043	0.0142	0.0128	0.0119	0.0094
walking_static	0.0163	0.0075	0.0144	0.0130	0.0057	0.0032	0.0026	0.0018
walking_rpy	0.1504	0.0677	0.1343	0.1215	0.0254	0.0232	0.0241	0.0208
walking_halfsphere	0.2669	0.1002	0.2473	0.2342	0.0425	0.0379	0.0345	0.0319

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

Table 3. Comparison of relative pose error between ORB-SLAM3 algorithm and the proposed algorithm (m)

dataset sequence	ORB-SLAM3 algorithm				proposed algorithm			
	RMSE	S.D.	Mean	Median	RMSE	S.D.	Mean	Median
walking_xyz	0.0102	0.0056	0.0086	0.0075	0.0072	0.0048	0.0054	0.0040
walking_static	0.0040	0.0033	0.0023	0.0014	0.0027	0.0012	0.0016	0.0010
walking_rpy	0.0060	0.0037	0.0047	0.0038	0.0044	0.0025	0.0018	0.0011
walking_halfsphere	0.0131	0.0086	0.0099	0.0077	0.0084	0.0065	0.0069	0.0058

Table 4. Comparison of absolute trajectory error between the proposed algorithm and other algorithms (m)

	Det-SLAM		DS-SLAM		Dyna-SLAM		proposed algorithm	
	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.
walking_xyz	0.0553	0.0376	0.0247	0.0161	0.0150	**0.0086**	**0.0142**	0.0128
walking_static	**0.0049**	**0.0027**	0.0081	0.0036	0.0060	0.0034	0.0057	0.0032
walking_rpy	0.0386	0.0235	0.4442	0.2350	0.0350	0.0437	**0.0254**	**0.0232**
walking_halfsphere	0.0626	0.0347	0.0303	**0.0159**	**0.0250**	0.0161	0.0425	0.0379

3.3 Real-World Scenario Experiments

To verify the effectiveness of the proposed algorithm in real dynamic environments, we designed an experiment where the camera moves from top to bottom, then returns to a horizontal position. The camera first rotates to the left, immediately rotates to the right, and then moves backward, while a person walks around a table. Figure 7 shows the comparison of Absolute Trajectory Error between the ORB-SLAM3 algorithm and the proposed algorithm, and Figure 8 illustrates the effectiveness of dynamic feature point filtering in real-world scenarios. Table 5 records the number of keyframes extracted by ORB-SLAM3 and the proposed algorithm. Since the camera experienced left and right rotations during movement, and the proposed algorithm optimizes the keyframe selection strategy, it extracted more keyframes. From Figures 7 to 8, it is clear that extracting more keyframes during rotation helps ensure keyframe quality, thereby preventing tracking loss. This shows that in real dynamic scenes, ORB-SLAM3 fails to extract enough keyframes during camera rotations, resulting in inaccurate localization and even erroneous trajectories in some areas. In contrast, the proposed algorithm estimates the camera's motion trajectory with smaller errors and higher overlap with the true trajectory, demonstrating that the proposed algorithm handles high-dynamic environments more effectively.

Figure 8(a) shows an RGB image randomly selected from a real-world scene. To evaluate the algorithm’s ability to recognize dynamic objects and dynamic feature points, a person was used as a moving object in a real scene, walking around the table at a normal pace while the camera moved and recorded the experimental environment information for later experiments. Figure 8(b) shows the original feature points initially extracted by the algorithm. The original algorithm extracted feature points from all objects without effectively distinguishing dynamic objects. Figure 8(c) shows the result after filtering out dynamic feature points. After performing object detection and instance segmentation on the original feature points, dynamic objects were accurately identified, and the feature points located in the moving object areas were treated as high-dynamic feature points. The movement probability of these feature points was updated, and through feature matching in the

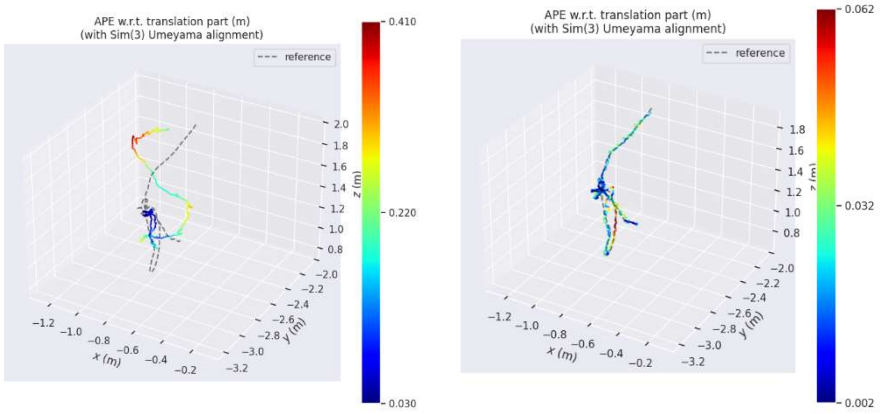
ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

tracking thread, the movement probability was propagated. Before camera pose estimation, the feature points in the moving regions were effectively removed.

In summary, the experimental results show that the ORB-SLAM3 algorithm has larger trajectory and relative pose errors in dynamic scenes, whereas the proposed algorithm reduces Absolute Trajectory Error and Relative Pose Error in dynamic scenes while maintaining the real-time performance of the SLAM algorithm.

Table 5. The number of keyframes extracted

Algorithm	ORB-SLAM3	Proposed Algorithm
Number of Keyframes	260	296



(a) Absolute trajectory error of ORB-SLAM3(b) Absolute trajectory error of the proposed algorithm

Fig. 7 Absolute trajectory error of ORB-SLAM3 and the proposed algorithm in real-world scenarios

(a) RGB image from real scene

(b) Original feature points map

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

(c) Image after filtering out dynamic feature points

Fig. 8 The effect of filtering the dynamic feature points in real-world scenarios

4. Conclusion

To address the issue of localization accuracy errors in SLAM algorithms within dynamic environments, this paper proposes a visual SLAM algorithm based on dynamic feature point filtering and motion probability propagation. Building upon the ORB-SLAM3 framework, an additional detection thread is introduced, utilizing the Detectron2 algorithm for object detection and instance segmentation on keyframes. In the tracking thread, the motion probabilities of feature points are updated through feature matching and match point expansion. Once the motion probabilities are propagated to each point, all dynamic points are removed. Experimental results show that the proposed SLAM algorithm improves localization accuracy in dynamic scenes, achieving high precision and accuracy. To further enhance the robustness of SLAM systems, future work will consider using 360-degree panoramic cameras to capture image frames, further improving the localization accuracy of the SLAM algorithm.

References

[1] Cadena C, Carlone L, Carrillo H, et al. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age[J]. IEEE Transactions on Robotics, 2016, 32(6): 1309-1332.

[2] Taketomi T, Uchiyama H, Ikeda S. Visual SLAM algorithms: A survey from 2010 to 2016[J]. IPSJ Transactions on Computer Vision and Applications, 2017, 9. DOI: 10.1186/s41074017-0027-2.

[3] Saputra M R U, Markham A, Trigoni N. Visual SLAM and structure from motion in dynamic environments: A survey [J]. ACM Computing Surveys, 2018, 51(2). DOI: 10.1145/3177853.

[4] Liu Q, Duan F H, Sang Y, et al. A survey of loop-closure detection method of visual SLAM in complex environments[J]. Robot, 2019, 41(1): 112-123,136.

[5] Merrill N, Huang G Q. CALC2.0: Combining appearance, semantic and geometric information for robust and efficient visual loop closure[C]//IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway, USA: IEEE, 2019: 4554-4561.

[6] Rünz M, Agapito L. Co-Fusion: Real-time segmentation, tracking and fusion of multiple objects[C]//IEEE International Conference on Robotics and Automation. Piscataway, USA: IEEE, 2017: 4471-4478.

[7] An Lifeng, Zhang Xinyu, Gao Hongbo, et al. Semantic segmentation-aided visual odometry for urban autonomous driving[J]. International Journal of Advanced Robotic Systems, 2017, 14(5): 1-11.

[8] Yu Chao, Liu Zuxin, Iu Xinjun, et al. DS-SLAM: A semantic visual SLAM towards dynamic environments[C]//IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscata -way, USA: IEEE, 2018: 1168-1174.

[9] Bescos B, Fácil J M, Civera J, et al. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes[J]. IEEE Robotics and Automation Letters, 2018, 3(4): 4076-4083.

[10] Xiao L H, Wang J G, Qiu X S, et al. Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment[J]. Robotics and Autonomous Systems, 2019, 117: 1-16.

[11] Wang Z M, Zhang Q, Li J S, et al. A computationally efficient semantic SLAM solution for dynamic scenes[J]. Remote Sensing, 2019, 11(11). DOI: 10.3390/rs11111363.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202502_11(2).0013

[12] Redmon J, Farhadi A. YOLOv3: An incremental improvement [DB/OL]. (2018-04-08) [2019-06-03]. https://arxiv.org/abs/1804.02767v1.

[13] Zhang J, Henein M, Mahony R, et al. VDO-SLAM: A visual dynamic object-aware SLAM system[DB/OL]. (2020-05-22) [2020-08-20]. https://arxiv.org/abs/2005.11052.

[14] Chen Zhihuan, Wang Zuao, Li Xiangcheng. VSLAM Method Based on Depth Camera in Indoor Dynamic Scenes[J]. Journal of Inertial Technology, 2023, 31(04): 390-400.

[15] Cui An, Zhang Xinying, Ma Yaohui. AGV Visual SLAM Algorithm Based on Adaptive Epipolar Constraint in Complex Environments[J]. Journal of Inertial Technology, 2024, 32(03): 234-241.

[16] Liu Fengyu, Cheng Xianghong, Cao Yi. Indoor Dynamic SLAM Method Based on Deep Learning and Feature Point Velocity Constraint[J]. Journal of Inertial Technology, 2023, 31(05): 438-443.

[17] Campos C, Elvira R, Rodríguez G J J, et al. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam[J]. IEEE Transactions on Robotics, 2021, 37(6): 1874-1890.

[18] Cheng Xiaoliu. Visual-Inertial SLAM: Theory and Source Code Analysis[M]. Beijing: Electronic Industry Press, 2023: 153-155.

[19] Gao Xiang, Zhang Tao, et al. Fourteen Lectures on Visual SLAM: From Theory to Practice[M]. Beijing: Electronic Industry Press, 2019: 165-169.

[20] Zhong F W, Wang S, Zhang Z Q, et al. Detect-SLAM: Making object detection and SLAM mutually beneficial[C]//IEEE Winter Conference on Applications of Computer Vision. Piscat-away, USA: IEEE, 2018: 1001-1010.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to sift or surf. In Computer Vision (ICCV), 2011 IEEE international conference on, pages 2564–2571. IEEE, 2011.