HMC-SLAM: A Robust VSLAM Based on RGB-D Camera in Dynamic Environment Combined Hierarchical Multidimensional Clustering Algorithm

Bowen Xu, Zexuan Zheng, Zihao Pan, and Lei Yu, *Member, IEEE*

***Abstract*— Most of the current visual simultaneous localization and mapping (SLAM) systems assume that the working environment is static, but dynamic targets in the actual working environment will reduce the position estimation accuracy. Aiming at the problem that visual SLAM cannot filter dynamic targets in a dynamic environment, this article proposes a visual SLAM (VSLAM) system for a dynamic environment based on hierarchical multidimensional clustering (HMC) of feature points, called HMC-SLAM, which can effectively reduce the localization error in dynamic environments. The system identifies a dynamic detection box by combining semantic information and polar geometric constraints. In the dynamic detection box, feature point clusters are divided by fusing the polar distance and depth information of feature points. During the process of dynamic feature point cluster removal, the strongest dynamic point cluster is first identified, and then the adjacent point clusters in its depth range are removed together. To further refine the rejection, we define optimized weights for the feature points in the detection box to avoid the influence of potential dynamic points. In the experiments, we evaluate the performance of the proposed HMC-SLAM in the TUM and BONN datasets and real scenarios. In the TUM and BONN datasets, the algorithmic accuracy and stability of HMC-SLAM are improved by 80.37% and 87.48% relative to ORBSLAM3, respectively. The real scenario results show that HMC-SLAM maintains good localization accuracy, effectively rejects dynamic targets, and is more robust than the current state-of-the-art dynamic SLAM methods.**

***Index Terms*— Depth information, dynamic environment, hierarchical clustering, simultaneous localization and mapping (SLAM).**

Received 7 July 2024; revised 13 January 2025; accepted 9 February 2025. Date of publication 14 March 2025; date of current version 31 March 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 61873176; in part by the Opening Foundation of the Key Laboratory of Opto-Technology and Intelligent Control, Ministry of Education; in part by the Open Research Fund of Anhui Province Key Laboratory of Machine Vision Inspection under Grant KLMVI-2023-HIT-17 and Grant KLMVI-2024-HIT-01; in part by Jiangsu Provincial Graduate Research and Practice Innovation Program under Grant KYCX24_3317; in part by the Open Fund of Jiangsu Provincial Key Laboratory of Advanced Robot Technology; and in part by Tang Scholar of Soochow University and Jiangsu Province's "333 High Level Talent Training Project." The Associate Editor coordinating the review process was Dr. Jing Yuan. *(Corresponding author: Lei Yu.)*

Bowen Xu, Zexuan Zheng, and Zihao Pan are with the Department of Mechanical and Electric Engineering, Soochow University, Suzhou 215131, China (e-mail: 1791133188@qq.com; 1368448220@qq.com; 1337445350@qq.com).

Lei Yu is with the School of Mechanical and Electric Engineering, Soochow University, Suzhou 215006, China, and also with Anhui Province Key Laboratory of Machine Vision Inspection, Wuhu 241000, China (e-mail: slender2008@163.com).

Digital Object Identifier 10.1109/TIM.2025.3551451

I. Introduction

IN UNFAMILIAR working environments, robots achieve self-awareness and accurately compute their coordinate positions using simultaneous localization and mapping (SLAM) technology. This technology has been widely applied in various fields such as autonomous driving, AR/VR [1], and autonomous navigation. Visual SLAM (VSLAM) not only provides positioning capabilities for robots [2] but also delivers rich and advanced environmental information. Among them, due to the relatively low cost of the cameras used in VSLAM, its applications have become increasingly popular. Although VSLAM shows excellent performance under existing conditions, its applicability is somewhat limited because it is typically based on static environments. In dynamic environments, moving objects generate a large number of mismatched point pairs, and, in complex multitarget motion scenarios, the pure rotational motion of dynamic objects can also cause drift in the pose estimation of SLAM systems, all of which lead to a degradation in algorithm performance. Therefore, effectively eliminating the impact of moving objects on VSLAM systems has become one of the critical research issues. In traditional SLAM frameworks, such as ORBSLAM3, the algorithm can iteratively treat some dynamic feature points as outliers through the Random Sample Consensus (RANSAC) [3] algorithm and remove them during the relocalization phase. However, RANSAC can easily fail when there are multiple dynamic targets in the frame. To address the existing problems and challenges in dynamic SLAM systems, we propose a visual SLAM algorithm for dynamic scenes called hierarchical multidimensional clustering (HMC)-SLAM, which is based on YOLOv5 [4] and of feature points. HMC-SLAM adopts ORBSLAM3 [5] as its core framework and integrates object detection, epipolar constraints, and depth information to detect dynamic objects. The main contributions of this work are summarized as follows.

1) To accurately distinguish between dynamic and static regions in dynamic detection bounding boxes, a hierarchical clustering algorithm based on feature point depth and epipolar distance is proposed. This method can effectively categorize the feature points in the bounding box into different clusters, enhancing the accuracy of the segmentation between dynamic and static regions. 2) To eliminate the influence of dynamic

feature points, a dynamic filter (DF) based on the dynamic attributes of feature point clusters is introduced. This method identifies and eliminates the dynamic feature point clusters in the detection bounding box by anchoring to the largest dynamic cluster and incorporating depth information, thereby avoiding the impact of dynamic feature points on pose optimization. 3) To further mitigate the influence of potential dynamic points in the detection bounding box, we calculate the confidence weights of the feature points in the dynamic box and construct an optimization weight matrix to jointly optimize the camera pose, providing more accurate localization.

The remainder of this article will unfold in the following structure. In Section II, we will review and discuss the existing works related to this research. Subsequently, in Section III, we will elaborate on the primary theoretical framework and model of the dynamic point removal method proposed in our VSLAM system. Section IV will focus on the experimental section, where we will detail the specific process of the experiments and conduct an in-depth comparison and analysis of the experimental results obtained on the dynamic datasets TUM [6] and BONN [7], as well as in real-world scenarios. Finally, in Section V, we will summarize the main work of this article and present corresponding conclusions.

II. Related Works

For dynamic scenes, traditional frameworks such as LSD-SLAM [8] and ORB-SLAM2 [9] employ the RANSAC [3] algorithm to find the optimal estimates through iterative optimization. ARD-SLAM [10] combines geometric motion information with expected motion data to effectively segment moving objects. StaticFusion [11] and ReFusion [7] rely on K-means clustering and computing pixel residuals, respectively, to identify dynamic regions in images or videos. Nevertheless, both methods struggle to accurately identify all dynamic regions. Fan et al. [12] used a combination of a camera and a inertial measurement unit (IMU) to constrain the trajectory estimation of the notched region. This approach avoids the influence of dynamic objects on position optimization, but the IMU also introduces some cumulative errors. When the camera motion is too fast or complex, the reliability of traditional geometric-based dynamic SLAM methods decreases, and these methods make it difficult to cope with complex scenes. SCP-SLAM [13] applies CNN only to keyframes, reducing the computational cost. It introduces "static confidence," which characterizes dynamic object features using interframe geometric residuals and propagates across frames to improve efficiency. USD-SLAM [14] combines the 3-D spatial motion state constraint module with a large segmentation model, enabling it to accurately handle dynamic objects in the environment.

With the flourishing development of deep learning technologies, numerous dynamic SLAM systems incorporating deep neural networks have emerged. Several advanced algorithms have combined semantic segmentation methods with SLAM systems. DynaSLAM [15], for instance, integrates Mask R-CNN and multiview geometry techniques to effectively handle dynamic points on humans or other dynamic objects. Pan et al. [16] used feature point re-extraction to solve the problem of an insufficient number of original feature points. DS-SLAM [17] employs a lightweight segmentation network SegNet and accelerates the processing speed of each frame by assigning the segmentation task to a separate processing thread. DGM-VINS [18] completes instance segmentation in complex environments by establishing temporal correlation of instance objects. Dynamic SLAM [19] calculates the spatial velocity of feature points between adjacent frames for fusion judgment and removes dynamic feature points. Amos-SLAM [20] proposes a method involving hyperpixel extraction and geometric clustering to classify potential motion regions based on color and geometric cues in the image.

Some algorithms utilize bounding boxes obtained from YOLO or SSD to identify potential dynamic objects. CD-SLAM [21] leverages scene flow to compute the distance between adjacent frames and determine their spatial velocities, compensating for potential static information through a velocity filtering algorithm. WF-SLAM [22] defines weights for feature points based on semantic segmentation networks and geometric masks and optimizes the camera pose based on these weighted feature points. RLD-SLAM [23] leverages Bayesian filtering to rapidly acquire high-quality static feature points. Crowd-SLAM [24] introduces a lightweight network named CYTi specifically designed to detect pedestrians in crowded scenes and remove the resulting dynamic points. OVD-SLAM [25] utilizes chi-square testing within YOLO detection boxes to filter out dynamic feature points. COEB-SLAM [26] proposes a method to handle motion blur and reassign feature points to accurately identify static and dynamic features. YOLO-SLAM [27] introduces a novel geometric constraint method, Depth-RANSAC, which can distinguish static points within bounding boxes. CFP-SLAM [28] employs the extended Kalman filter (EKF) and the Hungarian algorithm to compensate for potential missed detections in the YOLOv5 object detection network. SG-SLAM [29] proposes an innovative dynamic feature suppression method that can quickly and effectively suppress dynamic features. Although the R_CNN family of algorithms demonstrates excellent object detection accuracy and provides fine contours of target objects, its two-stage processing leads to relatively slow performance, making it unsuitable for SLAM systems with stringent real-time demands. Additionally, it requires the storage of numerous candidate regions and intermediate feature data, resulting in high memory consumption, which hinders deployment on platforms like mobile robots. Therefore, in this article, we opt for the YOLOv5 network, which requires less computational power and offers faster recognition speed.

III. System Introduction

Our proposed HMC-SLAM system uses four parallel threads of tracking, object detection, local mapping, and loopclosing to adapt to the dynamic environment, and the system framework is shown in Fig. 1.

A. Dynamic Property Checking

In complex motion scenarios involving multiple dynamic objects, YOLOv5 can effectively detect target objects;

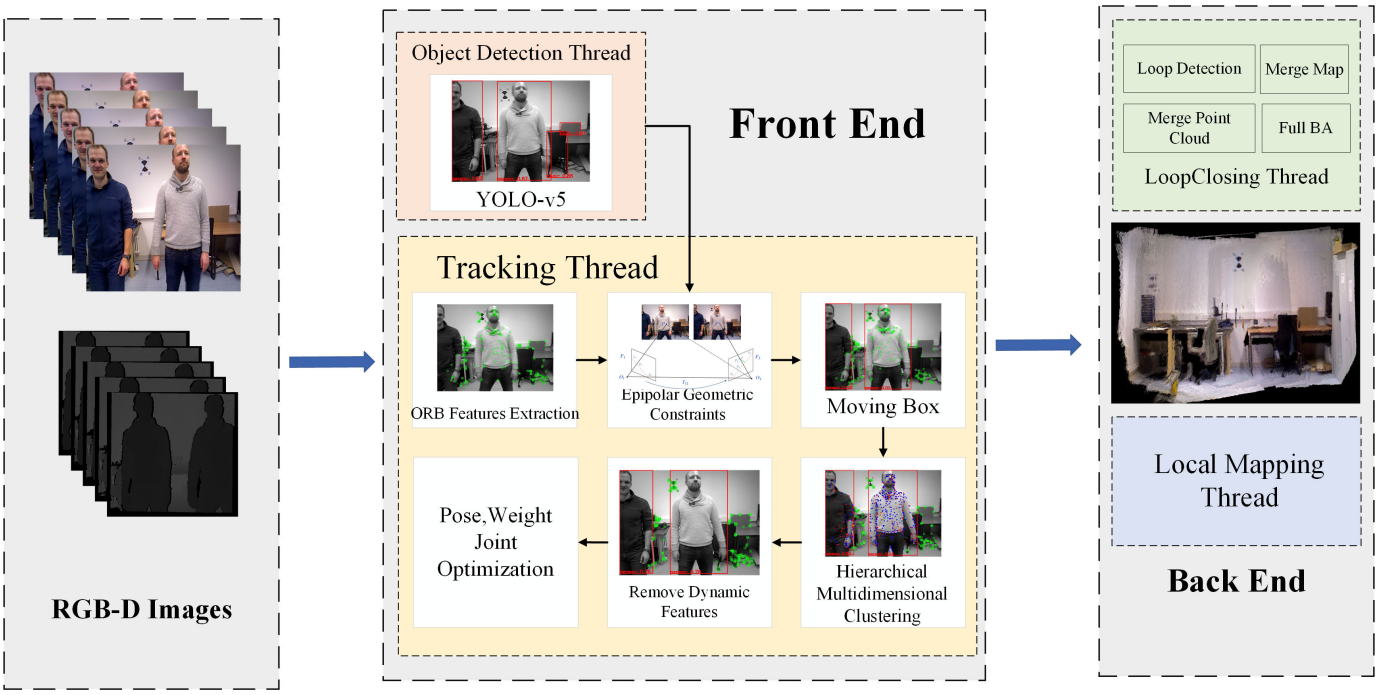


Fig. 1. System overview of HMC-SLAM. The RGB image is fed into the YOLOv5 neural network to obtain the detection box, while the feature points are extracted on the image, after which the dynamic detection box is filtered using the pair of polar geometric constraints, HMC is performed within the target detection box, followed by constructing the DF to reject the dynamic point clusters. Finally, the points within the target detection frame are set with confidence level to participate in joint optimization.

however, it is incapable of discerning the motion state of these objects.

To accurately judge and reject dynamic objects in the current frame, the current motion state of the target must be judged using the polar geometry constraints. We calculate the polar distance of each feature point to distinguish between dynamic and static feature points. Fig. 2 illustrates the pairwise polar geometry constraint relationship between the previous frame F_1 and the current frame F_2.

The construction of the constraints on the polar geometry is divided into the following steps: first, we use the pyramid-based Lucas–Kanade optical flow algorithm to compute the matching feature points in two neighboring images. Then, for each matched feature point, the polar coordinate position of the current image is determined by computing the fundamental matrix. Next, the distance from each feature point to its corresponding polar coordinate line is calculated. If the calculated distance is greater than a set threshold, we consider it a dynamic feature point; conversely, it is a static point.

Let the chi-squared representation of the matching points in the previous frame and the current frame image be

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

where u and v are the coordinates of the point as observed in the corresponding frame, and we denote the basis matrix as F. The pairwise polar geometric constraint can then be expressed as

$$P_2^{\mathrm{T}} F P_1 = 0. \tag{2}$$

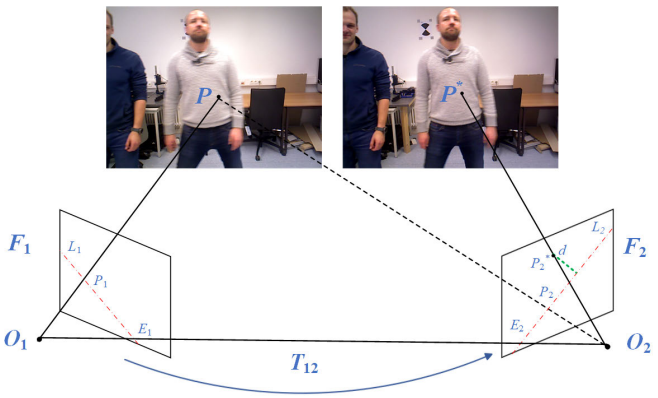


Fig. 2. O_1 and O_2 denote the optical center of the camera, P and P^* are the two feature points matched in the image, P_1 and P_2 denote the original position if P is a static point, and P_2^* denotes the false position due to the motion of the point. O_1, O_2, and point P together form the polar plane and E_1 and E_2 denote the intersection of the baselines O_1, O_2, and the two frames of the image. The polar plane intersects F_1 and F_2 at the polar lines L_1 and L_2, as shown by the red lines in the figure, and the green line d indicates the distance from the feature point to its polar line.

This allows us to obtain the polar line L of the feature point P_1 of the previous frame in the current frame as

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

Then, the distance from the feature point P_2 to the corresponding polar line L in the current frame can be expressed as

$$d = \frac{\left|P_2^{\mathrm{T}} F P_1\right|}{\sqrt{\|X\|^2 + \|Y\|^2}}. \tag{4}$$

After referring to the excellent work of DS-SLAM [17] and WF-SLAM [22], this article chooses to set the threshold value of d to 1. Specifically, a feature point is classified as dynamic if its distance d exceeds 1. Considering that humans are the most prevalent dynamic entities in the environment, any feature point located within a detection box labeled as "person" is considered dynamic if its distance d exceeds 0.9. Furthermore, if the proportion of dynamic points within a detection box exceeds 10% of the total feature points in that box, the box is deemed to contain dynamic objects. These thresholds have been empirically determined through extensive experimentation.

B. Hierarchical Multidimensional Clustering

Although the target detection method is efficient in execution, the detection box may contain a significant number of static features. Removing these static features may lead to tracking failures.

To address the above problems, we propose an HMC algorithm based on depth and polar distance, which clusters discrete feature points into different clusters of feature points by fusing the depth information of the 3-D space and the polar distance of the planar image.

Before executing the clustering algorithm, this system will preprocess the feature points in the image, first, it will screen out all the feature points in the current detection box and check the depth value of each feature point, this algorithm takes the camera position under the camera coordinate system as the origin, and if the depth value of the feature point is 0 or a negative number, it will be excluded from the next clustering calculation.

First, determine the nearest neighbor radius d_{eps} for clustering and the minimum number of feature points threshold $MinPoints$ for nearest neighbors. The distance between two points is described by the Minkowski distance, and it may be useful to set the representation of the two points on the n-dimensional space as, respectively,

$$x = (x_1, x_2, \ldots, x_n), \quad y = (y_1, y_2, \ldots, y_n). \tag{5}$$

Then, the distance between the two using the Min distance measure can be expressed as

$$D(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}. \tag{6}$$

Here, the p-value is related to the dimensionality of the data and here the p-value is chosen to be 2. The clustering process is shown in Fig. 3.

Fig. 3 illustrates the process of clustering feature points in the detection box. For the sake of simplicity, this section assumes that the value of $MinPoints$ is 4. First, all the feature points in the detection box are marked as pending clustering status, and at the beginning of the clustering, one feature point is randomly selected from all the pending clustered feature points for the core point decision, as shown in Fig. 3(a), since there are four points in the neighborhood of point a in region A, point a is marked as a core point. However, at this point in time, no new core point is found in the feature

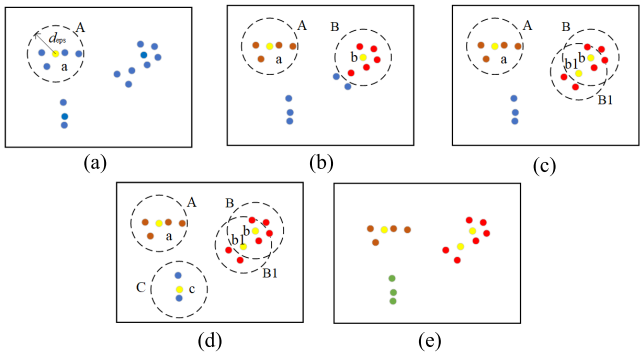


Fig. 3. Diagram of the clustering process, where the blue points represent the data points that have not been accessed, the yellow points are the data points that have been selected for the core point judgment, the circles A, B, C, and so on are the detection areas with a radius of d_{eps}, and points of the same color indicate that they belong to the same point cluster. (a)–(e) Describe the clustering process of feature points.

Fig. 4. First column shows the original RGB image. The second column displays the inlier points retained by ORB-SLAM3. The third column shows all the extracted feature points. The fourth column presents the results of removing all feature points within the dynamic detection box. The fifth column presents the results of the clustering algorithm proposed in this article.

points of region A. Therefore, the clustering of region A is finished and the point cluster is marked in orange, as shown in Fig. 3(b). Among the remaining feature points to be clustered, another feature point b is randomly selected for judgment, and there are five points in its neighborhood region B, then feature point b is the core point. Among the five points in region B, point b1 is also found to be a core point, at this time there is no more core point in region B1, then the clustering of feature points in region B, B1 is finished, and the feature points in the two regions are classified as the same point cluster, as shown in Fig. 3(c). Finally, the unclustered feature point c is selected for core point judgment, in its neighborhood C, the number of feature points is less than 4, so feature point c is not a core point, and the points in region C are marked as noise points, as shown in Fig. 3(d). At this point, there is no unclustered feature point in the detection box, and the result at the end of clustering is shown in Fig. 3 (e). The clustering results of feature points during the experiment are shown in Fig. 4. Before the clustering algorithm, we perform a filtering step on the feature points to remove those with anomalous depth values. Therefore, the number of points in the clustering algorithm will not be identical to the number of extracted feature points.

C. Dynamic Point-Cluster Filter

Relying solely on polar distance thresholding is insufficient for effectively handling scenarios where epipolar geometry constraints fail, which may result in incomplete removal of dynamic feature point clusters and consequently lead to significant performance degradation of the algorithm.

To solve the above problem, we propose a filtering method that anchors the maximum dynamic cluster, and the maximum dynamic cluster must meet the following conditions: first, the number of feature points in this point cluster should not be less than the set value. Second, among the point clusters for which the number of feature points has been satisfied, the point clusters with the largest mean value of the polar distance should be selected, which indicates that such point clusters have the strongest dynamic attributes in the current detection box.

The screening of feature point clusters is divided into two stages.

Rough Screening of Polar Distance: If the mean polar distance of each type of point cluster is greater than the global mean polar distance of all feature points, then we consider this type of point cluster to be a dynamic point cluster.

Maximum Dynamic Cluster Screening: Since the point clusters with adjacent depth values often describe the same object, we use depth values to participate in the screening together. First, we calculate the difference Δd between the maximum depth point and the minimum depth point in the largest dynamic cluster, then we calculate the depth difference between the average depth $\{d_{\text{aver1}}, d_{\text{aver2}}, \ldots, d_{\text{avern}}\}$ of the rest of the feature point clusters and the average depth d_{maxaver} of the largest dynamic point cluster, if it is less than Δd, then we consider the point cluster to be the nearest neighbor of the largest dynamic cluster, and mark the cluster as a dynamic cluster and reject it.

Finally, we perform a thorough traversal of all points within the static point clusters and eliminate misclassified dynamic points by utilizing the polar distance criterion. The results of this dynamic filtering process are presented in Fig. 5.

In the real scene demonstration shown in the fourth row of pictures, we constructed static occluders to occlude dynamic objects, and it can be seen that HMC-SLAM can accurately differentiate between dynamic and static point clusters in the dynamic detection box, which improves the static constraints and the robustness of the system operation. From the third column, it can be seen that there will still be some dynamic points relying only on the polar distance for screening.ORBSLAM3 is unable to eliminate the feature points on dynamic targets.

D. Joint Optimization of Feature Point Pose Weights

In the current frame position optimization of ORBSLAM3 [5], all feature points are included in the position optimization. However, due to the presence of dynamic points, incorrect observations will lead to incorrect estimation of the reprojection error, which will affect the accuracy of the positioning.

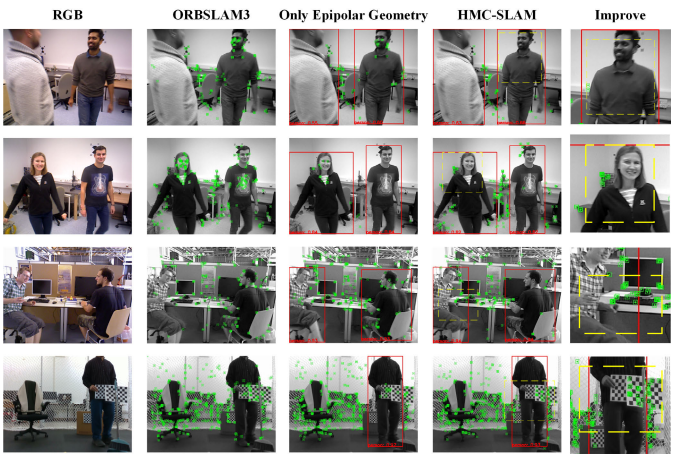


Fig. 5. Green dots indicate the preserved feature points, and the yellow boxes are the regions of superior performance improvement. The first column is the RGB image, the second column displays the inlier points retained by ORB-SLAM3, the third column is the dynamic feature point rejection based only on a single thresholded polar distance, the fourth column is our algorithm, and the fifth column is the region where the effect of our algorithm is significantly improved.

In the dynamic feature filtering process, the system has already removed the feature points within the detection box based on both the average polar distance of feature point clusters and the individual polar distances. If a dynamic feature point remains at this stage, it must have a smaller polar distance, and from the perspective of view geometry, it is no longer possible to eliminate it.

To address the above problems, we propose a method based on the depth difference between the feature points and the largest dynamic cluster to calculate the dynamic weights of the feature points to reduce the influence of dynamic points that are not filtered out in the DF. We judge from the perspective of the depth of the feature point, for any point in the detection box, if the difference δd between the depth d_{pt} of the point and the average depth d_{maxaver} of the largest dynamic cluster is smaller, then it is considered that the point is more likely to be dynamic.

Assuming that the depth distribution of all points in the detection box follows a Gaussian distribution, the standard deviation (S.D.) of the depth values of all points is solved for σd and the expected value of the Gaussian distribution is d_{maxaver}, for the dynamic points that we have already judged we set the point weights to 0, for the purely static points outside the box we set their weights to 1, and for the suspected points inside the box we use the Gaussian distribution probability density function to describe their dynamic weights:

$$\lambda_i = \begin{cases} 1, & \text{Point Out Of Box} \\ 1 - G(d_{\text{maxaver}}, \sigma d), & \text{Point In Box} \\ 0, & \text{Point Dynamic .} \end{cases} \tag{7}$$

Define $G(d_{\text{maxaver}}, \sigma d)$ as

$$G(d_{\text{maxaver}}, \sigma d) = e^{-\frac{(d_{\text{pt}} - d_{\text{naxaver}})^2}{2(\sigma d)^2}}. \tag{8}$$

We assign point weights based on the depth of the feature points in the current frame. As a result, the optimization cost function transitions from optimizing solely the current frame

pose to a joint optimization of both the current frame pose and the feature point weights. We define the projection error

$$E_i(z_i) = \lambda_i |z_i - h(RZ_i + t)| \tag{9}$$

where z_i is the observation point in the current frame, Z_i is the corresponding map point, h is the projection transformation from the camera coordinate system to the plane coordinate system R, and t is the camera position to be optimized. Then, the joint weight optimization function is

$$\{R^*, t^*\} = \underset{R,t,\lambda_i}{\operatorname{argmin}} \sum_{i=1}^{n} \left(E_i(z_i)^T Q^{-1} E_i(z_i)\right) \tag{10}$$

where Q^{-1} is the information matrix and R^* and t^* are the optimized camera poses.

In this article, we use the Levenburg–Marquardt method from the G2O library for the solution. Although joint optimization can mitigate the impact of dynamic points, experimental observations indicate that it consumes significant computational resources. Therefore, joint position optimization is performed only when the ratio of the number of feature points within the dynamic detection box to the total number of feature points exceeds 0.6, thus striking a balance between computational efficiency and accuracy.

IV. Experimental Results

To quantitatively evaluate the effectiveness of the HMC-SLAM algorithm, we conducted experiments on two publicly available datasets, TUM [6], BONN [7], and real-world scenarios.

We use absolute trajectory error (ATE) and relative position error (RPE) to evaluate the accuracy. The root mean square error (RMSE) and S.D. are commonly used to describe the trajectory accuracy and stability of the system. The RPE includes relative translation error and relative rotation error. For each estimated trajectory, we compute the accuracy error relative to the true value, considering the inherent uncertainty of the algorithm. We performed five independent experiments for each sequence and reported the mean value as the final result. The optimal result for each sequence is highlighted in bold, while the suboptimal result is underlined for clarity.

All experiments were conducted on a computer with Ubuntu 20.04 operating system, Intel i5 CPU, RTX4070ti GPU, and 32 GB of RAM, and we used an Intel REALSENSE D455 camera to capture the real scene dataset.

A. Ablation Experiments

We conducted ablation experiments to evaluate the impact of each of the system's modules on the system's operational accuracy, runtime, and tracking rate. We evaluated the Region Segmentation (RS) module, the DF module, and the Optimization of Position Weights (OPW) module. For the different modules, we chose different ablation methods; for the RS module, we changed the hierarchical multidimensional clustering to remove all feature points in the detection box; for the filter module, we changed the joint filtering to filter the dynamic points by relying only on the polar distance thresholds; and for the Optimization of Position Weights module, we turned it off for the experiments.

TABLE I
Ablation Experiment ATE Results (M)

Sequence	RS		DF		OPW		HMC-SLAM	
	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓
CROWD	0.0221	0.0127	0.4652	0.2403	0.0212	0.0138	**0.0172**	**0.0087**
CROWD2	0.0331	0.0208	0.0273	0.2635	0.0258	0.0151	**0.0254**	**0.0146**
CROWD3	0.0292	0.0143	0.0389	0.0259	0.0226	0.0121	**0.0204**	**0.0095**
SITSTATIC	0.0143	0.0069	**0.0055**	**0.0026**	0.0063	0.0027	0.0059	**0.0026**
WALKHALF	0.2182	0.3112	0.0216	0.0905	0.0207	0.0118	**0.0195**	**0.0099**
WALKRPY	0.0488	0.0208	0.0277	0.0142	0.0289	0.0208	**0.0257**	**0.0139**
WALKXYZ	0.0351	0.0088	0.0278	0.0216	0.0165	0.0088	**0.0152**	**0.0083**

TABLE II
Running Time and Tracking Rate of Each Module of the Ablation Experiment (Ms)

Sequence	RS		DF		OPW		HMC-SLAM Tracking	
	Time	Trj	Time	Trj	Time	Trj	Time	Trj
CROWD	2.71	88.1%	\	99.8%	11.55	94.8%	28.33	94.8%
CROWD2	3.12	78.9%	\	100.0%	16.92	100.0%	38.14	100.0%
CROWD3	13.86	74.3%	\	99.2%	3.26	99.2%	33.61	99.2%
SITSTATIC	3.43	96.1%	\	96.2%	0.64	96.2%	17.32	96.2%
WALKHALF	2.39	98.7%	\	96.8%	3.74	96.8%	22.32	96.8%
WALKRPY	2.45	85.8%	\	99.1%	1.21	99.1%	21.75	99.1%
WALKXYZ	1.84	96.2%	\	96.2%	3.82	96.2%	23.99	96.2%

For our experiments, we used three multitarget complex scene CROWD sequences from the BONN dataset, one low-speed dynamic SIT scene sequence, and three high-speed dynamic WALK scene sequences from the TUM dataset. To verify the effectiveness of the tracking of the SLAM system, we introduce the metric of the tracking rate Trj. The value of Trj is calculated as follows:

$$Trj = \frac{F_{\mathrm{O}}}{F_{\mathrm{I}}} \times 100\% \tag{11}$$

where F_{I} denotes the total number of frames entered into the system and F_{O} denotes the number of frames successfully tracked by the system.

The experimental results are presented in Tables I and II. As indicated in Table I, the system (RS) and the system (DF) exhibit the lowest trajectory accuracy. This is primarily because the system (RS) removes all feature points within the detection box, resulting in a significant number of frames that cannot be tracked. The system (DF) relies only on a single polar distance threshold to filter dynamic feature points, which reduces the system's ability to cope with complex motion scenarios and cannot effectively deal with algorithmically degraded scenarios. The existence of joint optimization of the system (OPW) position weights compensates for the lack of DF screening to some extent and improves position accuracy. The average RMSE improvement of HMC-SLAM for module RS, module DF, and module OPW is 47.06%, 29.40%, and 8.75%, respectively. The average S.D. improvement is 41.84%, 58.13%, and 17.20%, respectively.

From Table II, it can be seen that the system (RS) removes a large number of static constraints resulting in the worst tracking rate, from the time point of view the system (RS) runs only 10%–20% of the total tracking time, but makes the average tracking rate improve by 11.65%, and the average tracking rate under the complex multitarget CROWD sequences improves by 22.62%, which shows that the region splitting is very effective. The system (OPW) spends a lot of

TABLE III
RESULTS OF ATE (M)

Sequence	ORBSLAM3 [5]		OVD-SLAM [25]		SG-SLAM [29]		REFUSION [7]		HMC-SLAM	
	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓
CROWD	1.1212	0.7169	0.0199	0.0121	0.0274	0.0158	0.1166	0.0439	**0.0172**	**0.0087**
CROWD2	0.9695	0.4991	**0.0233**	**0.0126**	0.0554	0.0394	1.6265	0.4982	0.0254	0.0146
CROWD3	0.7251	0.3052	0.0209	**0.0091**	0.0958	0.0501	0.0665	0.0366	**0.0204**	0.0095
SYCH2	1.3405	0.1583	0.0076	0.0039	1.3924	0.1689	0.0214	0.0127	**0.0074**	**0.0038**
SITHALF	0.0207	0.0108	0.0285	0.0135	0.0156	0.0077	0.0486	0.0269	**0.0132**	**0.0046**
SITSTATIC	0.0109	0.0052	0.0081	0.0037	0.0096	0.0047	0.0101	0.0054	**0.0059**	**0.0026**
WALKHALF	0.0323	0.1141	0.0251	0.0122	0.0311	0.0171	0.0486	0.0269	**0.0195**	**0.0099**
WALKXYZ	0.3781	0.2645	0.0153	**0.0074**	0.0164	0.0086	0.0741	0.0566	**0.0152**	0.0083
WALKSTATIC	0.3071	0.0722	**0.0071**	**0.0028**	0.0077	0.0037	0.0174	0.0159	0.0095	0.0059
WALKRPY	0.7642	0.4692	0.0531	0.0271	0.0402	0.0286	0.2588	0.1879	**0.0257**	**0.0139**

TABLE IV
TRANSLATIONAL RPE RESULTS (M/S)

Sequence	ORBSLAM3 [5]		OVD-SLAM [25]		SG-SLAM [29]		REFUSION [7]		HMC-SLAM	
	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓
CROWD	0.7917	0.6613	0.1295	0.0619	0.1414	0.0661	**0.1252**	0.0654	0.1265	**0.0594**
CROWD2	0.5405	0.3667	0.1255	**0.0584**	0.1531	0.0779	0.4851	0.3255	**0.1235**	0.0592
CROWD3	0.4725	0.2796	**0.0955**	**0.0403**	0.1852	0.1129	0.1201	0.0569	0.0987	0.0421
SYCH2	0.5701	0.2116	**0.0434**	**0.0262**	0.3903	0.2442	0.3499	0.1479	0.1228	0.1196
SITHALF	0.0201	0.0121	0.0311	0.0155	0.0193	0.0088	0.0408	0.0296	**0.0151**	**0.0075**
SITSTATIC	0.0103	0.0049	0.0094	0.0049	0.0127	0.0066	0.0079	0.0045	**0.0077**	**0.0036**
WALKHALF	0.2298	0.1857	0.0275	0.0135	0.0328	0.0178	0.0548	0.0326	**0.0217**	**0.0119**
WALKXYZ	0.2091	0.1722	0.0207	0.0099	0.0214	0.0112	0.0854	0.0643	**0.0189**	**0.0089**
WALKSTATIC	0.1158	0.1002	0.0088	0.0039	0.0101	0.0048	0.0171	0.0114	**0.0082**	**0.0035**
WALKRPY	0.2962	0.2041	**0.0414**	0.0251	0.0521	0.0341	0.2736	0.2158	0.0432	**0.0246**

time on optimization when facing complex motion scenarios, the optimization time, in general, dynamic scenarios only accounts for 3.69%–16.76% of the total tracking time, and the optimization time in complex multitarget scenarios only increases to 44.36%, but at the same time the average RMSE improves by 10.05%. We believe that such optimization time consumption is worthwhile for the improvement of accuracy. The impact of the system (DF) on the tracking rate is minimal, and its runtime is negligible; as a result, the runtime data is not included in the table.

From the results in Tables I and II, it can be seen that the HMC-SLAM system performs best, the RS system improves the tracking rate of the system, and the DF and OPW systems work together to improve the trajectory accuracy.

B. Overall System Evaluation

We selected representative dynamic sequences for comparison experiments on two publicly available datasets, TUM [6] and BONN [7], for visual SLAM algorithms with superior performance. The TUM RGB-D dataset consists of RGB-D images captured using the Kinect depth camera, with ground-truth data obtained through a motion capture system, covering a variety of indoor scenes. It is widely utilized for evaluating RGB-D SLAM algorithms. In contrast, the BONN dataset, captured with the ASUS Xtion Pro LIVE camera, includes numerous dynamic scenes, which present greater challenges compared to the TUM RGB-D dataset. For evaluation, we selected four multitarget, complex dynamic sequences (CROWD, SYCH2) from the BONN dataset, as well as two low-speed dynamic sequences (SIT) and four high-speed dynamic sequences (WALK) from the TUM dataset.

This section compares the performance of the HMC-SLAM algorithm with the mainstream dynamic visual SLAM algorithms. ORBSLAM3 [5] is the main framework of HMC-SLAM. OVD-SLAM [25] identifies dynamic points in the foreground using a chi-square test. SG-SLAM [29] combines the results of NCNN target detection networks with geometric information to achieve fast dynamic segmentation. REFUSION [7] uses a purely geometric approach to robustly cope with dynamic environments. The algorithms run on our experimental platform and the ATE of the different algorithms is shown in Table III, the translational RPE is shown in Table IV, and the rotational RPE is shown in Table V.

The ATE results are presented in Table III, indicating that our algorithms achieve optimal or near-optimal performance in most scenarios. In contrast, ORB-SLAM3 fails to effectively reject dynamic features in dynamic environments; only in the WALKSTATIC and CROWD2 sequences, we run with worse results than OVD-SLAM; the presence of DFs allows our algorithm to overcome the degradation problem of dynamic objects moving the polar line direction, so the accuracy of

TABLE V
ROTATIONAL RPE RESULTS (°/S)

Sequence	ORBSLAM3 [5]		OVD-SLAM [25]		SG-SLAM [29]		REFUSION [7]		HMC-SLAM	
	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE↓	S.D.↓
CROWD	20.1609	13.7686	**11.7041**	**6.3471**	11.8152	6.4322	12.5459	6.8549	11.9845	6.4938
CROWD2	20.2919	11.0185	15.8281	8.2955	15.9054	8.3251	17.7611	9.9692	**15.8103**	**8.2844**
CROWD3	17.1712	7.6058	15.0246	7.3872	15.0428	7.3823	15.6552	7.7432	**15.0169**	**7.3819**
SYCH2	9.8482	3.4973	3.9006	2.5827	6.4704	4.0335	**1.9494**	**0.6586**	2.4367	2.2282
SITHALF	**0.5774**	**0.2409**	0.8091	0.4071	0.6997	0.3393	2.3555	1.6577	0.5808	0.2792
SITSTATIC	0.3072	0.1299	0.2735	0.1183	0.2784	0.1223	0.3177	0.1518	**0.2605**	**0.1129**
WALKHALF	4.3501	3.5526	0.7906	0.3928	0.7851	**0.3839**	1.9661	1.0327	**0.7429**	0.4256
WALKXYZ	3.9489	3.2623	0.6268	0.3831	0.6211	**0.3768**	1.9141	1.3215	**0.6218**	0.3822
WALKSTATIC	2.0924	1.7789	0.2437	0.1061	0.2702	0.1186	0.3971	0.2088	**0.2385**	**0.1055**
WALKRPY	5.7739	3.9271	0.9266	0.5761	1.1637	0.7456	10.0115	8.3898	**0.8536**	**0.4557**

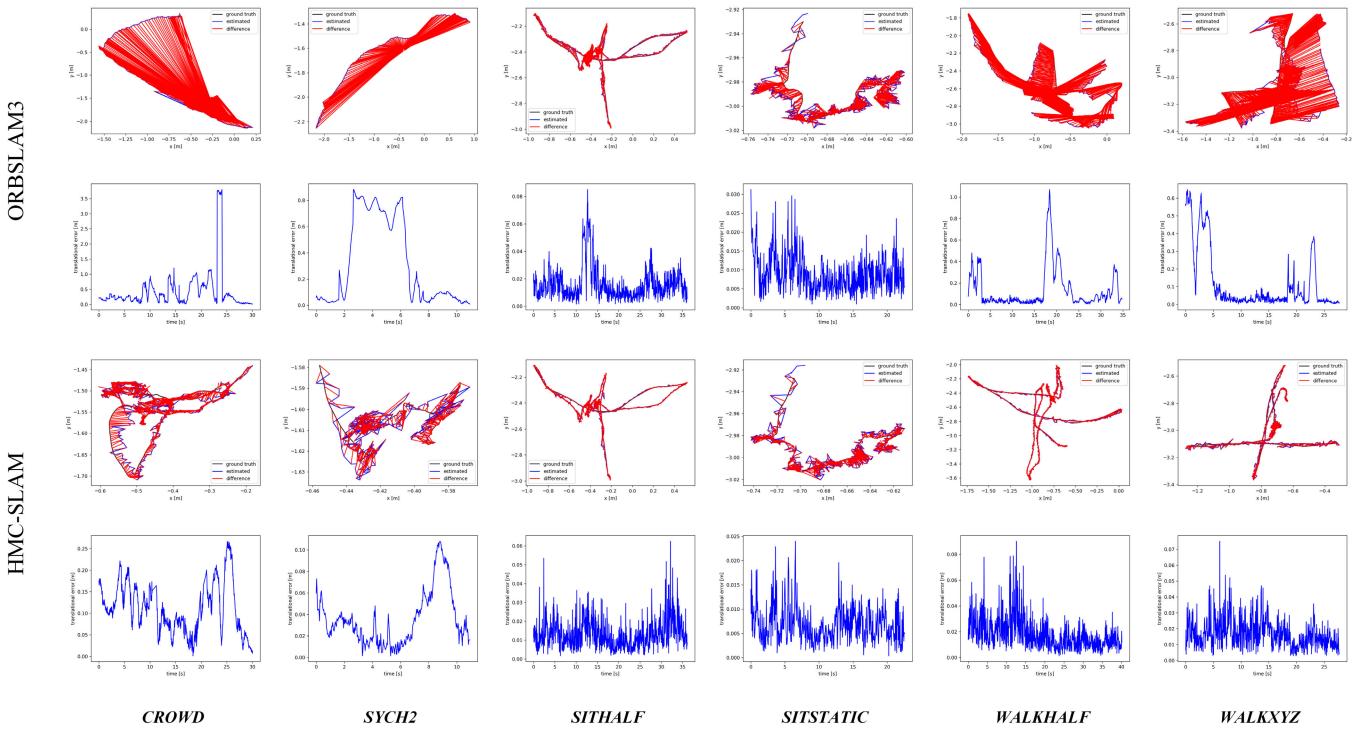


Fig. 6. ATE and RPE results for ORBSLAM3 and HMC-SLAM in some sequences. The first and third rows show the ATE results, where the black line represents the true trajectory, the blue line represents the trajectory estimated by the algorithm, and the red line represents the difference between the estimated and true values. The second and fourth rows show the RPE results, where the blue lines represent the RPE results for each time point.

our algorithm is better than SG-SLAM in most scenarios; the pure geometric method cannot completely eliminate the dynamic features, so the accuracy of the REFUSION algorithm is not as good as our algorithm. For fast dynamic sequences and complex multitarget sequences, the ATE of our algorithm improves the average RMSE by 88.88% and the average S.D. by 95.69% compared to ORBSLAM3.

As presented in Tables IV and V, the RPE results of our algorithm demonstrate superior localization accuracy and stability compared to other algorithms across most sequences. Specifically, the average RMSE for translational and rotational RPE in our approach improves by 72.83% and 50.61%, respectively, while the average S.D. shows improvements of 74.07% and 47.28% relative to ORBSLAM3. These experimental results indicate that our front-end integrated dynamic filtering algorithm significantly outperforms ORBSLAM3, particularly in handling complex multitarget, high-speed dynamic scenarios.

From the experimental results, it can be seen that the three performance metrics of the present algorithm are optimal on most of the experimental data in most of the sequences of the TUM dataset, and the performance is much better than the other algorithms in low and high dynamic environments. Out of a total of 24 experimental results in the BONN dataset, the performance of the present algorithm reaches optimal or suboptimal in 22 of them and optimal in 11 of them. And the BONN data is more difficult, and the performance metrics of the algorithms are generally worse. Due to the presence

TABLE VI
SYSTEM RUNTIME (MS)

Algorithm	Average processing time per frame	Hardware Platform
ORBSLAM3 [5]	47.91	Intel i5+RTX4070Ti
OVD-SLAM [25]	46.09	Intel i5+RTX4070Ti
SG-SLAM [29]	46.21	Intel i5+RTX4070Ti
REFUSION [7]	128.99	Intel i5+RTX4070Ti
HMC-SLAM	51.02	Intel i5+RTX4070Ti

of some moving low-texture objects in the BONN dataset, which makes the randomness of feature point extraction larger, the algorithm faces a higher challenge, but the experimental results show that the present algorithm still has a considerable advantage over other algorithms in the complex multidynamic target scene.

Fig. 6 shows the results of the trajectory visualization for ORBSLAM3 and HMC-SLAM. It can be seen that the estimated trajectory obtained by our algorithm has a much smaller error between it and the true value.

C. Runtime Analysis

Real-time performance is a critical criterion for evaluating the efficacy of SLAM systems. To assess the real-time capability of the SLAM system, we compared the average processing time required by each algorithm to process a single frame.

Since the processing time per frame is different for different sequences, we counted the time on multiple datasets and calculated the average. The experimental results and hardware platforms are shown in Table VI.

The running time of our algorithm is improved by 60.45% compared to REFUSION, and the trajectory accuracy of our algorithm is improved by 90% compared to ORBSLAM3, but the average processing time per frame is only increased by 3.11 ms, so the real-time requirement can still be met.

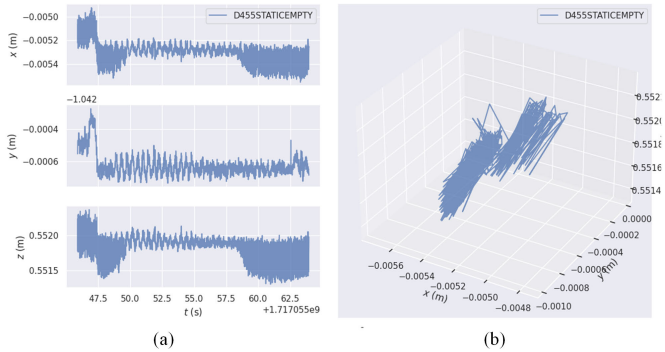
D. Performance Verification in Real Scenarios

To further evaluate the performance of HMC-SLAM, we designed a real-world environment to assess the algorithm's trajectory and reconstruction accuracy. We conducted the experiments by constructing a real scene in a motion capture lab, which is approximately 6×6 m in size. We captured RGB-D information using an Intel Realsense D455 handheld camera with motion capture flags at a resolution of 640 × 480 and a frame rate of 30 fps. The trajectory of the camera operation was also captured using a NOKOV metric optical 3-D motion capture system. The experimental environment and equipment are shown in Fig. 7.

First, we placed the camera statically without applying any motion to it and used a motion capture device to capture the camera's motion in a static environment and plotted the trajectory as shown in Fig. 8, which shows that the motion capture device we used can accurately capture the camera's position. We recorded five datasets to assess the effectiveness of our algorithm, namely HALF (where the camera follows a hemispherical trajectory), XYZ (where the camera moves along the XYZ three-axis), TRI (where the camera moves

(a) (b) (c)

Fig. 7. (a) Intel Realsense D455 camera. We built an experimental scene as shown in (b) using a broom, a cardboard box, and a chair. (c) NOKOV metric optical 3-D motion capture device.



(a) (b)

Fig. 8. (a) Fluctuation of the acquisition trajectory broken down into the three X-, Y-, and Z-axes, the fluctuation values are all within 0.5mm, indicating that our acquired camera trajectory is accurate. (b) Position of the camera in 3-D space.

TABLE VII
REAL SCENARIO EXPERIMENT ATE RESULTS (M)

Sequence	ORBSLAM3 [5]		HMC-SLAM		IMPROVED	
	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE	S.D.
HALF	1.1224	0.4001	0.0547	0.0244	95.12%	93.90%
XYZ	0.0677	0.0271	0.0659	0.0259	2.65%	4.42%
TRI	0.0483	0.0325	0.0189	0.0119	60.87%	63.38%
FRONT	0.6681	0.2941	0.0466	0.0232	93.03%	92.11%
STATIC	0.0053	0.0046	0.0017	0.0009	67.92%	80.43%

along an approximate triangular trajectory), FRONT (where the camera moves forward and backward), and STATIC (where the camera remains stationary).

We have compared the performance of ORBSLAM3 and our algorithm, the experimental ATE results are shown in Table VII and the panning RPE is shown in Table VIII, the first two columns of the table are the running results obtained by ORBSLAM3 and our algorithm, respectively, and the third column is the improvement in the effectiveness of our algorithm compared to ORBSLAM3.

To better demonstrate the stability and robustness of this algorithm, we give the improvement values of this algorithm compared to ORBSLAM3 in Tables VII and VIII, with reference to the method in DS-SLAM [17]

$$\eta = \frac{o - r}{o} \times 100\% \tag{12}$$

where η is the improvement value, o is the ORBSLAM3 value, and r is the HMC-SLAM value.

As shown in Table VII, the average RMSE and average S.D. of ATE of this algorithm in real scenarios are improved by 63.92% and 66.85%, respectively, when compared with that of ORBSLAM3. The improvement is most significant

TABLE VIII
REAL SCENE EXPERIMENTAL TRANSLATION RPE RESULTS (m/S)

Sequence	ORBSLAM3 [5]		HMC-SLAM		IMPROVED	
	RMSE↓	S.D.↓	RMSE↓	S.D.↓	RMSE	S.D.
HALF	0.4671	0.2440	0.3876	0.1918	17.02%	21.39%
XYZ	0.2075	0.1163	0.1982	0.1089	4.48%	6.36%
TRI	0.4205	0.1712	0.4201	0.1711	0.09%	0.06%
FRONT	0.4557	0.2121	0.3997	0.1807	12.29%	14.80%
STATIC	0.0082	0.0069	0.0024	0.0012	70.73%	82.61%

TABLE IX
SYSTEM RUNTIME IN REAL SCENARIOS (MS)

Sequence	Average processing time per frame	Hardware Platform
ORBSLAM3 [5]	44.11	Intel i5+RTX4070Ti
HMC-SLAM	50.48	Intel i5+RTX4070Ti

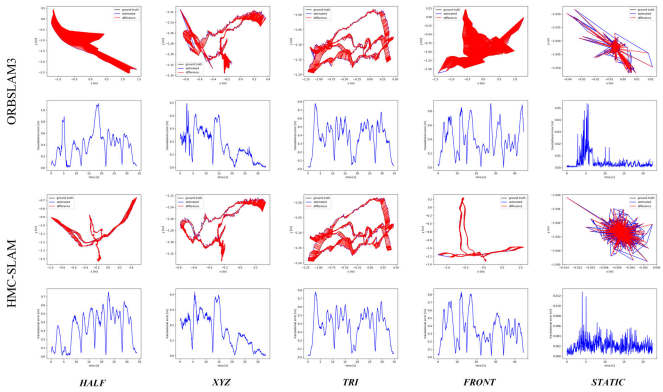


Fig. 9. ATE and RPE results for ORBSLAM3 and HMC-SLAM in real scenarios. The first and third rows show the ATE results, where the black lines represent the real trajectories, the blue lines represent the trajectories estimated by the algorithm, and the red lines represent the difference between the estimated and real values. The second and fourth rows show the RPE results, where the blue lines represent the RPE results at each time point.

on two sequences of HALF and FRONT, and the average enhancement reaches 94.07%.

As shown in Table VIII, the average RMSE and average S.D. results of the translational RPE are improved by 20.92% and 25.04%, respectively. Our algorithm effectively improves the estimation of trajectory accuracy and system stability in real dynamic scenarios.

Table IX presents the average processing time per data frame for ORBSLAM3 and HMC-SLAM across five real-world demonstration sequences. Although HMC-SLAM shows a 14.44% decrease in efficiency compared to ORBSLAM3, the average RMSE of the ATE and translational RPE improve by 63.92% and 20.92%, respectively, which represents an acceptable tradeoff.

The results of the visualization of the estimated trajectories versus the true values are shown in Fig. 9. It can be seen that this algorithm has better position accuracy compared to ORBSLAM3.

We also performed a real-time dense reconstruction of the dynamic scene to verify the trajectory accuracy of the SLAM system, and the reconstruction results are shown in Fig. 10.

Various experimental results demonstrate that the RS module of HMC-SLAM significantly enhances the tracking

Fig. 10. First four columns (a) group is the scene of the recorded dataset, and the fifth column (b) group is the reconstruction result. From the reconstruction results, our SLAM system can provide accurate real-time positions.

rate of the SLAM system, while the dynamic filtering and joint optimization module improves localization accuracy. Compared to conventional dynamic vision SLAM algorithms, our approach exhibits superior performance and robustness. Experiments conducted in real-world scenarios further validate that the algorithm is highly effective in handling dynamic environments.

V. CONCLUSION

In this article, we propose a real-time robust visual SLAM algorithm, which improves the tracking accuracy of the system in dynamic environments and effectively filters out dynamic objects in the environment. First, the YOLOv5 neural network, feature point sparse optical flow, and pair of pole geometric constraints are used to filter out the detection box where dynamic objects exist, and then the dynamic region within the detection box is accurately segmented using multidimensional hierarchical clustering and DFs. Finally, to avoid the influence of potential dynamic points, we assign confidence weights to the feature points within the detection box to participate in a joint optimization that optimizes the loss function and compensates for the influence of dynamic feature points.

Experiments on public datasets show that our method performs well in dynamic environments and outperforms current mainstream visual SLAM algorithms. Furthermore, experiments in a real-world environment with highly dynamic scenes show that our proposed method can be applied to realistic scenes. However, the average processing time per frame of our algorithm is longer compared to similar applications of the YOLO detection algorithm. In the future, we plan to improve the running efficiency of our algorithm to better meet real-time requirements.

ACKNOWLEDGMENT

The authors would like to thank the referees for their constructive comments.

REFERENCES

[1] Y. Sato, K. Minemoto, M. Nemoto, and T. Torii, "Construction of virtual reality system for radiation working environment reproduced by gamma-ray imagers combined with SLAM technologies," *Nucl. Instrum. Methods Phys. Res. A, Accel. Spectrom. Detect. Assoc. Equip.*, vol. 976, Oct. 2020, Art. no. 164286.

[2] J. An, H. Mou, R. Lu, and Y. Li, "Localization and navigation analysis of mobile robot based on SLAM," *J. Phys., Conf.*, vol. 1827, no. 1, Mar. 2021, Art. no. 012089.

[3] M. A. Fischler and R. 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.

[4] G. Jocher et al., "Ultralytics/YOLOv5: V7.0—YOLOv5 sota real-time instance segmentation," Zenodo, 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7347926

[5] C. Campos, R. Elvira, J. J. G. Rodríguez, 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, Dec. 2021.

[6] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.

[7] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss, "ReFusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 7855–7862.

[8] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Sep. 2014, pp. 834–849, doi: 10.1007/978-3-319-10605-2_54.

[9] 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, Oct. 2017.

[10] Q. Ul Islam, H. Ibrahim, P. Kok Chin, K. Lim, M. Zaid Abdullah, and F. Khozaei, "ARD-SLAM: Accurate and robust dynamic SLAM using dynamic object identification and improved multi-view geometrical approaches," *Displays*, vol. 82, Apr. 2024, Art. no. 102654.

[11] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background reconstruction for dense RGB-D SLAM in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 3849–3856.

[12] C. Fan, J. Hou, and L. Yu, "Large-scale dense mapping system based on visual-inertial odometry and densely connected U-Net," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–16, 2023.

[13] M.-F. Yu, L. Zhang, W.-F. Wang, and J.-H. Wang, "SCP-SLAM: Accelerating DynaSLAM with static confidence propagation," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces (VR)*, Mar. 2023, pp. 509–518.

[14] J. Wang et al., "USD-SLAM: A universal visual SLAM based on large segmentation model in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 9, no. 12, pp. 11810–11817, Dec. 2024.

[15] 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, Oct. 2018.

[16] Z. Pan, J. Hou, and L. Yu, "Optimization RGB-D 3-D reconstruction algorithm based on dynamic SLAM," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–13, 2023.

[17] C. Yu et al., "DS-SLAM: A semantic visual SLAM towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1168–1174.

[18] B. Song, X. Yuan, Z. Ying, B. Yang, Y. Song, and F. Zhou, "DGM-VINS: Visual-inertial SLAM for complex dynamic environments with joint geometry feature extraction and multiple object tracking," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–11, 2023.

[19] S. Wen et al., "Dynamic SLAM: A visual SLAM in outdoor dynamic scenes," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–11, 2023.

[20] Y. Zhuang et al., "Amos-SLAM: An anti-dynamics two-stage RGB-D SLAM approach," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–10, 2024.

[21] S. Wen, S. Tao, X. Liu, A. Babiarz, and F. R. Yu, "CD-SLAM: A real-time stereo visual–inertial SLAM for complex dynamic environments with semantic and geometric information," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–8, 2024.

[22] Y. Zhong, S. Hu, G. Huang, L. Bai, and Q. Li, "WF-SLAM: A robust VSLAM for dynamic scenarios via weighted features," *IEEE Sensors J.*, vol. 22, no. 11, pp. 10818–10827, Jun. 2022.

[23] Z. Zheng, S. Lin, and C. Yang, "RLD-SLAM: A robust lightweight VI-SLAM for dynamic environments leveraging semantics and motion information," *IEEE Trans. Ind. Electron.*, vol. 71, no. 11, pp. 14328–14338, Nov. 2024.

[24] J. C. V. Soares, M. Gattass, and M. A. Meggiolaro, "Crowd-SLAM: Visual SLAM towards crowded environments using object detection," *J. Intell. Robot. Syst.*, vol. 102, no. 2, p. 50, Jun. 2021.

[25] J. He, M. Li, Y. Wang, and H. Wang, "OVD-SLAM: An online visual SLAM for dynamic environments," *IEEE Sensors J.*, vol. 23, no. 12, pp. 13210–13219, Jun. 2023.

[26] F. Min, Z. Wu, D. Li, G. Wang, and N. Liu, "COEB-SLAM: A robust VSLAM in dynamic environments combined object detection, epipolar geometry constraint, and blur filtering," *IEEE Sensors J.*, vol. 23, no. 21, pp. 26279–26291, Nov. 2023.

[27] 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, Apr. 2022.

[28] X. Hu et al., "CFP-SLAM: A real-time visual SLAM based on coarse-to-fine probability in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 4399–4406.

[29] 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, pp. 1–12, 2023.