Robotica (2025), 1–24
doi:10.1017/S0263574725102701

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Integrating gray area feature points and static probability to enhance the performance of indoor dynamic visual simultaneous localization and mapping

Huilin Liu[1], Lunqi Yu[1] and Shenghui Zhao[2]

[1]School of Computer Science and Engineering, Anhui University of Science and Technology, Huainan, 232001, People's Republic of China
[2]Anhui Engineering Research Center of Intelligent Perception and Elderly Care, Chuzhou, 239000, People's Republic of China
Corresponding author: Lunqi Yu; Email: 2023201221@aust.edu.cn

Received: 13 February 2025; **Revised:** 11 August 2025; **Accepted:** 23 September 2025

Keywords: visual SLAM; deep feature; dynamic environments; gray area feature points; static probability

Abstract

In dynamic environments, moving objects pose a great challenge to the accuracy and robustness of visual simultaneous localization and mapping (VSLAM) systems. Traditional dynamic VSLAM methods rely on hand-designed feature frames, and these methods usually make it difficult to fully utilize feature information in dynamic regions. To this end, this paper proposes a SLAM system (GAF-SLAM) that combines gray area feature points, weighted static probabilities, and spatio-temporal constraints. This method realizes the efficient fusion of key point detection and target detection by introducing YOLO-Point to extract gray area feature points from dynamic regions. These feature points are located within the detection frame and have potentially static feature point properties. By combining the reprojection error and polar geometry constraints, potential static feature points are effectively screened out and the identification of these gray area feature points is further optimized. Subsequently, a novel static probabilistic computational framework is designed to assign weights to these gray area feature points and dynamically adjust their influence on the optimization results during the attitude estimation process. By combining static probability with temporal continuity and spatial smoothness constraints, the system achieves significantly improved localization accuracy and robustness in dynamic environments. Finally, the proposed method was evaluated on the TUM RGB-D dataset. The experimental results demonstrate that GAF-SLAM significantly improves pose estimation accuracy and exhibits strong robustness and stability in dynamic indoor environments.

1. Introduction

Simultaneous localization and mapping (SLAM) technology is capable of real-time self-localization and map construction in unknown environments, and it is a key technology in the fields of robotics, autonomous driving, and augmented reality [1–5]. Compared with traditional laser SLAM, visual SLAM (VSLAM) has lower cost and richer environment sensing capability, which makes it show great potential in applications such as robot navigation, augmented reality (AR), virtual reality (VR), and intelligent surveillance. In recent years, with the rapid development of computer vision and machine learning, the performance and robustness of VSLAM have been significantly improved. However, most of the existing VSLAM algorithms are based on static environment assumptions, leading them to exhibit limitations in complex dynamic environments.

Current mainstream VSLAM methods, such as ORB-SLAM2 [6], ORB-SLAM3 [7], LSD-SLAM [8], and DSO [9], demonstrate good accuracy and stability in static scenes. However, in dynamic environments, moving objects (such as pedestrians and vehicles) can introduce mismatched feature points, disrupting the system's state estimation. This interference may lead to a significant decrease in the robustness of the visual SLAM system or even a crash. Therefore, effective detection and rejection

of dynamic objects are crucial for the accuracy and robustness of vision SLAM systems in dynamic scenes.

Recently, methods such as DS-SLAM [10], Dyna-SLAM [11], and DE-SLAM [12] have attempted to improve the system performance by using semantic segmentation and target detection networks to obtain semantic information about dynamic elements, and combining geometric constraints to remove these dynamic feature points. However, the high dependence of these methods on semantic information leads to a significant increase in computational overhead, posing a greater challenge to real-time performance. In addition, these methods rely on traditional indirect VSLAM system using the hand-crafted features, limiting the overall robustness of the system, and its ability to handle complex dynamic scenes. Meanwhile, the complete elimination of feature points in the dynamic region may lead to insufficient constraints in the position estimation and affect the localization accuracy. Therefore, how to balance the detection of dynamic objects and the preservation of static features in dynamic environments becomes a key issue for research.

To cope with the above challenges, this paper proposes the GAF-SLAM algorithm, which introduces the YOLO-Point [13] deep learning network into the visual SLAM system to replace the traditional ORB feature extraction method. Combining gray area feature points screening and static probability calculation in dynamic environment, the system's localization accuracy and stability are improved by analyzing the feature points in the dynamic region, effectively identifying and removing the dynamic feature points, while retaining more static feature points. After calculating the static probability from all the static feature points, the weighted static probability and spatio-temporal constraints are fused to perform the dynamic pose estimation, and a more accurate position is solved. The specific contributions of this paper are as follows:

1. This paper introduces the concept of "gray area feature points" and designs a dynamic feature point screening and static probability calculation framework that integrates deep learning with geometric optimization. By incorporating the YOLO-Point network, the system achieves dynamic object detection and feature point extraction, and accurately identifies gray area feature points in dynamic regions using reprojection error and epipolar geometry constraints. In addition, a static probability calculation method is proposed, which assigns static weights to gray area feature points based on reprojection distance, epipolar distance, and observation state, thereby enhancing the robustness and accuracy of pose estimation in dynamic environments.
2. A dynamic pose estimation algorithm fusing weighted static probability and spatio-temporal constraints is proposed, which calculates the static probability of feature points and dynamically adjusts their weights in the optimization process. In addition, the algorithm further combines temporal continuity and spatial smoothness constraints to effectively optimize the weight allocation strategy of static feature points, thus enhancing the stability and robustness of feature point matching. Ultimately, the weighted sum of the reprojection error and the temporal consistency constraint is minimized by nonlinear least squares optimization, which achieves high-precision pose estimation in dynamic environments.
3. We integrated the method into the front-end of ORB-SLAM2 and evaluated the method on the TUM RGB-D datasets and Bonn RGB-D datasets, as well as tested it in real-world scenarios. The results show that GAF-SLAM achieves high localization accuracy and robust performance in various dynamic environments.

The paper is organized as follows. Section 2 is an introduction to the related work. Section 3 describes the main theoretical model and algorithm design of the method in this paper. Section 4 details the comparative analysis of the experimental results. The conclusions are presented in section 5.

2. Related work

2.1. Static SLAM

VSLAM algorithms can be categorized into two main types: direct methods and indirect methods. Direct methods rely on the assumption of pixel intensity invariance, utilizing photometric information directly

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

to minimize errors in pose estimation. Representative techniques such as LSD-SLAM and DSO are favored for their fast computational speed and adaptability to texture-scarce environments. However, these methods lack loop closure detection modules, which can lead to the accumulation of errors, and they exhibit insufficient robustness under varying lighting conditions.

In contrast, indirect methods estimate the camera pose by extracting and matching features. MonoSLAM [14] and ORB-SLAM2 are typical examples of this approach. Although they are somewhat slower in processing speed, they demonstrate greater robustness in scenarios with lighting changes and rapid camera motion. In ref. [15], the authors significantly improved the system's matching accuracy by introducing line features and utilizing IMU-assisted optical flow tracking to predict these line features. Liu [16] proposed a lightweight SLAM method based on pyramid IMU-predicted optical flow tracking, aiming to reduce the computational cost of feature tracking while enhancing the system's processing speed.

However, most existing VSLAM techniques assume that the external environment is static. In real-time applications, moving objects are prevalent, and this dynamic characteristic can significantly affect the localization accuracy and tracking performance of traditional VSLAM systems, which in turn seriously threatens the stability and accuracy of the system.

2.2. Dynamic SLAM

For the effect of dynamic scenes, the current VSLAM system mainly adopts two types of methods to recognize and reject dynamic feature points: geometric information methods and semantic information methods.

Geometric information methods utilize geometric constraints to detect and reject dynamic points. Such methods usually recognize dynamic features by detecting the motion consistency or geometric properties of the feature points. For example, Zou *et al.* [17] projected feature points from the previous frame to the current frame and classified static and dynamic feature points according to the magnitude of the 2D reprojection error. Wang *et al.* [18] combined polar constraints and RGB-D depth clustering information to identify outliers in neighboring frames to detect moving targets. Dai *et al.* [19] succeeded in distinguishing dynamic targets from static backgrounds by analyzing the correlation of map points, effectively reducing the influence of dynamic objects on position estimation. Song *et al.* [20] employ density-based spatial clustering of applications with noise (DBSCAN) [21] in conjunction with geometric consistency and epipolar constraints to remove dynamic feature points. However, these geometric methods rely on localized feature motion variations and are poorly adapted to large-scale dynamic scenes, which may affect the reliability of position estimation and map construction accuracy.

The semantic information approach extracts semantic information from images with the help of deep learning models to remove potential dynamic objects, which provides a new solution for the application of SLAM in dynamic environments. In recent years, the combination of deep learning and SLAM algorithms has made significant progress. For example, Dyna-SLAM combines Mask-R-CNN [22] and a multi-view geometry approach in the ORB-SLAM2 framework to effectively remove the dynamic point. Yang *et al.* [23] used faster R-CNN [24] to detect dynamic objects and further confirmed dynamic objects by geometric matching. DS-SLAM combined with Seg-Net [25] and motion consistency detection for accurate recognition of dynamic objects. Wen *et al.* [26] combined semantic information with pixel spatial motion features to effectively improve localization accuracy. The OVD-SLAM [27] utilizes pixel-level dynamic object segmentation to distinguish foreground from background, and recovers static points on moving objects by minimizing reprojection errors, thereby mitigating the negative impact of dynamic points on system performance. These methods perform superiorly in dynamic environments, but the high dependence on semantic information significantly increases the computational overhead and poses a serious challenge to real-time performance.

For this reason, other approaches have begun to try to optimize the use of semantic information. For example, YOLO-SLAM [28] combines a YOLO target detection network with VSLAM to cull out feature points in dynamic regions using real-time dynamic object detection, thus improving robustness and accuracy in dynamic environments. COEB-SLAM [29] proposed real-time dynamic SLAM algorithms

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

based on deep learning, extracting semantic information from the scene using object detection networks, and combining optical flow techniques to remove dynamic feature points, significantly reducing localization errors in dynamic environments. SG-SLAM [30] further enhances the system's robustness by combining semantic information with geometric information, enabling more accurate identification, and elimination of dynamic point interference. GGC-SLAM [31] calculates the fundamental matrix distance and uses object detection results to eliminate dynamic feature points, effectively reducing the impact of dynamic scenes on the SLAM system [32] improves traditional VSLAM by combining object detection to filter dynamic objects and focusing on static points. Yang *et al.* [33] combined deep learning with probabilistic filtering methods, significantly improving the robustness of VSLAM in dynamic environments. MPOC-SLAM [34] utilizes object category and motion probability modeling to significantly improve localization and map-building capabilities in highly dynamic environments.

Although these semantic approaches have achieved good results in dynamic environments, the complete elimination of semantic information from dynamic objects may lead to insufficient feature points and affect the matching performance of the system. Moreover, these methods still mainly rely on the traditional indirect SLAM framework, which uses hand-designed feature point detection and description methods and fails to fully exploit the potential of deep learning models for efficient joint feature extraction. This limitation provides a research direction to further improve the performance of SLAM in dynamic environments.

3. Improved VSLAM system

In most feature-based VSLAM methods, the camera rotation R and translation t are estimated by minimizing the reprojection error between the key points $x_i = (u_i, v_i)^T$and their corresponding 3D points $X_i = (x, y, z)^T$.

$$\begin{aligned}\{R^*, t^*\} &= \arg\min_{R,t} \frac{1}{2} \sum_{i=1}^{n} \|x_i - (RX_i + t)\|_2^2 \\ &= \arg\min_{R,t} \sum_{i=1}^{n} \frac{1}{2} \left\| x_i - \pi \left(\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ 1 \end{bmatrix} \right) \right\|_2^2 \end{aligned} \tag{1}$$

$$x_i = \pi\ (X_i) = \left(x f_x / z + c_x, y f_y / z + c_y \right)^T \tag{2}$$

In this context, $\pi(\cdot)$ denotes the camera transformation model that converts 3D coordinates to pixel coordinates, R^* *and* t^* represent the optimized camera pose, f_x and f_y are the camera focal length, c_x and c_y are the camera center coordinate.

In general, in the case of static environments, all extracted feature points participate in the optimization process. However, feature points from dynamic elements may interfere with this optimization process. Specifically, due to the lack of observable motion information, dynamic feature points cannot be matched to the original camera transform model without the support of other sensors. This mismatch negatively affects the optimization process of Eq. (1), which leads to an increase in the camera position error. Therefore, in order to significantly improve the adaptability of the system in dynamic scenes, it is necessary to exclude the participation of dynamic feature points while introducing the weights of static feature points in order to solve a more accurate bit pose.

GAF-SLAM is implemented on the basis of the ORB-SLAM2 framework, which is a traditional feature-point based SLAM method. As shown in Figure 1 the specific process is to input the image frames into the YOLO-Point network, and the output includes depth feature points, descriptors, and relevant information about the target detection frame. Next, the feature points in the detection box are re-evaluated by reprojection error and limiting geometric constraints to filter out gray area feature points. Subsequently, based on our proposed static probability algorithm for feature points, these gray area feature points are discriminated, and low probability feature points are eliminated in order to retain as many static feature points as possible. Ultimately, these preserved static feature points are utilized in

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

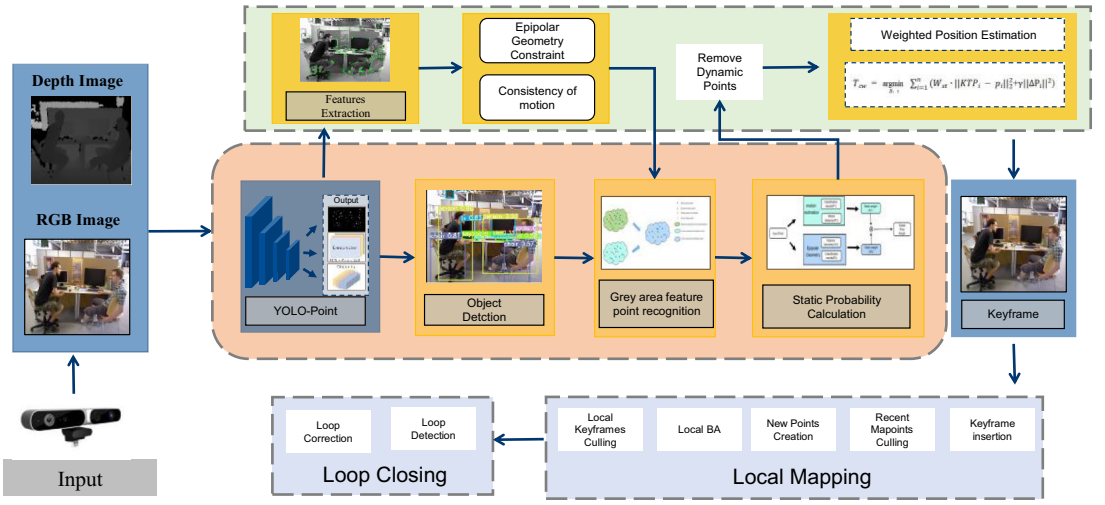

Figure 1. *To enhance the robustness and accuracy of ORB-SLAM2 in dynamic environments, a gray area feature point recognition module and a static probability calculation module are introduced. Traditional ORB feature points are replaced with YOLO-Point deep learning features, and the resulting feature points and descriptors are formatted in ORB style for seamless integration. Gray area screening and static probability calculation dynamically adjust feature point weights in pose estimation, optimizing the final pose for accurate tracking and localization.*

combination with their own static weights as well as in conjunction with spatio-temporal constraints for pose estimation.

3.1. YOLO-Point

Super-Point [35] is a multi-task neural network that realizes the tight integration of key point detection and descriptor generation by sharing the feature output of the backbone network. It is designed to be able to accomplish both tasks in a single forward propagation, which dramatically improves the computational efficiency and is particularly suitable for real-time scenarios. In addition, in recent years, there has been a trend to incorporate YOLO series of deep learning algorithms into SLAM systems. These algorithms provide a better solution for SLAM in dynamic scenes by accurately detecting dynamic objects and effectively reducing their interference with the localization and mapping system.

YOLO-Point proposes a unified framework that fuses key point detection with object detection. Unlike traditional methods that rely on separate feature extraction and post-processing modules, YOLO-Point is able to achieve multi-task learning in a single forward propagation, which significantly reduces computational complexity and improves real-time performance. Meanwhile, the feature points extracted by deep learning show stronger robustness in complex scenes such as lighting changes, view angle changes, and motion blurring. With the object detection capability, YOLO-Point is able to radically reduce the accumulation of localization errors in dynamic scenes, whereas traditional SLAM methods usually lack effective dynamic feature point processing strategies in dynamic environments.

The core design concept of YOLO-Point is to share the backbone network for fast and efficient predictive capability. In a single forward propagation, YOLO-Point not only synchronizes key point detection, descriptor generation, and target bounding box prediction, but also improves adaptability and robustness in dynamic environments through multi-task co-design. Compared to hand-designed feature points such as traditional ORB, deep feature points provide significantly better perception in complex scenes, as well as higher accuracy in bit-position estimation. Despite the high computational complexity of deep feature points, YOLO-Point achieves a balance between high accuracy and real-time performance by sharing feature extraction modules.

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

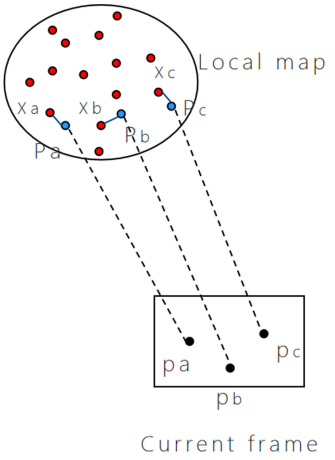

Figure 2. *Schematic diagram of reprojection error.*

In model training, YOLO-Point is first pretrained on the synthetic shape dataset and COCO dataset [36], and combined with single-response adaptive and mosaic data augmentation methods to reduce the error caused by padding, thus improving the accuracy of key point detection. Subsequent fine-tuning on the KITTI dataset [37] in stages further optimized the model's performance on new data and new object classes by freezing and unfreezing the weights of the different layers. In addition, to further enhance the closed-loop detection capability, we use the OpenLORIS dataset [38] to generate a vocabulary of in-depth features and achieve efficient loop detection through FBoW [39] search, which effectively reduces the cumulative error in the bit-position estimation. This training strategy not only ensures the accuracy of the model but also enhances its robustness in complex dynamic environments. YOLO-Point is pretrained on synthetic shape dataset and COCO dataset, combining single-response adaptive with mosaic data enhancement to reduce the filling error and improve the accuracy of key point detection. Subsequently, the model is fine-tuned in stages on the KITTI dataset to ensure that it performs well on new datasets and new object classes, ultimately achieving efficient and accurate detection performance.

3.2. Reprojection error

Reprojection error is a metric used in vision SLAM to measure the positional error of a 3D point projected onto an image. Specifically, the reprojection error measures the deviation of a 3D point from the actual detected 2D feature point after mapping it to the image coordinate system. A smaller reprojection error indicates that the 3D point is more consistent with the detected point on the image, which can usually be used as an indicator that the point belongs to a static environment, while a larger reprojection error may mean that the point is affected by a dynamic object.

As shown in Figure 2, assume that a 3D point $X_i = (X_i, Y_i, Z_i)$ in the local map is projected onto the image plane through the camera's projection matrix P, resulting in the projected 2D coordinates $p_i = (u_i, v_i)$. Ideally, this point should coincide with the position of the feature point in the image frame. If the actual detected position of the feature point is $p_i^{'}$, then the reprojection error can be expressed as:

$$e_{reproj} = \left\| p_i - p_i^{'} \right\| = \sqrt{\left(u_i - u_i^{'}\right)^2 + \left(v_i - v_i^{'}\right)^2} \tag{3}$$

where u_i, v_i represents the coordinates of the 3D point after projection to the image and $u_i^{'}$, $v_i^{'}$ are the coordinates of the image where the point is actually detected.

In dynamic environments, static feature points usually originate from fixed scene elements, and thus, the reprojection error between frames is small. Dynamic feature points, on the other hand, originate from

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

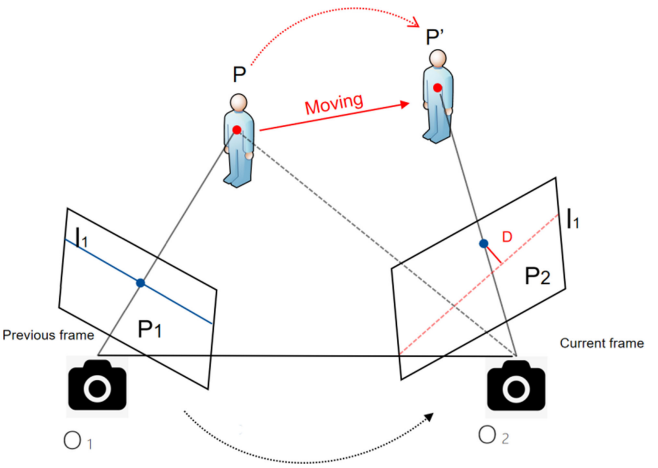

Figure 3. *the epipolar geometric constraints between the previous frame I1 and the current frame I2. L represents the nuclear line.*

moving objects, such as pedestrians or vehicles, and the reprojection error is usually large. The specific process is as follows: First, feature point detection and matching are carried out between the current frame and the previous frame. Then, using the estimated camera pose, 3D points from the local map are projected onto the current frame to calculate the reprojection error e_{reproj} for each matched feature point. By setting a threshold, if $e_{reproj} > \epsilon$, the feature point is marked as a dynamic feature point; otherwise, it is classified as a static feature point.

3.3. Epipolar geometry constraint

By epipolar geometry constraint, we can classify the current state of an object. The feature points of dynamic objects do not satisfy the constraint of epipolar geometry because they are not accurately located on the corresponding epipolar lines. Therefore, we measure the distance between feature points and their corresponding epipolar lines, and consider distances exceeding a specific threshold as outliers.

The process of epipolar geometry constraints can be divided into three steps. First, the pyramid-based Lucas-Kanade optical flow algorithm [40] is used to calculate matching feature points in two adjacent images. Next, the fundamental matrix is employed to compute the epipolar lines for each matching feature point in the current frame. Finally, the distances between the feature points and their corresponding epipolar lines are calculated. Based on the comparison of these distances with a predetermined threshold, we can determine the state of the feature points: if the distance exceeds the threshold, the feature point is considered to be in a moving state; otherwise, it is classified as static.

Figure 3 illustrates the epipolar geometry constraints between the previous frame image I1 and the current frame image I2. The camera observes the same spatial point P from different angles. In a dynamic scene, the point P moves to P' with the optical centers O1 and O2 defining the epipolar plane through the spatial point P. P1 and P2 are the feature points projected from the spatial point P in the previous and current frames, respectively. The intersection lines L1 and L2 of the epipolar plane with the two image planes are referred to as the epipolar lines. We denote the matched feature points in the previous frame and the current frame as:

$$p_1 = [u_1, v_1], \quad p_2 = [u_2, v_2] \tag{4}$$

Among them, u and v are pixel coordinate values, while the homogeneous coordinates of p_1 and p_2 can be expressed as:

$$P_1 = [u_1, v_1, 1], \quad P_2 = [u_2, v_2, 1] \tag{5}$$

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

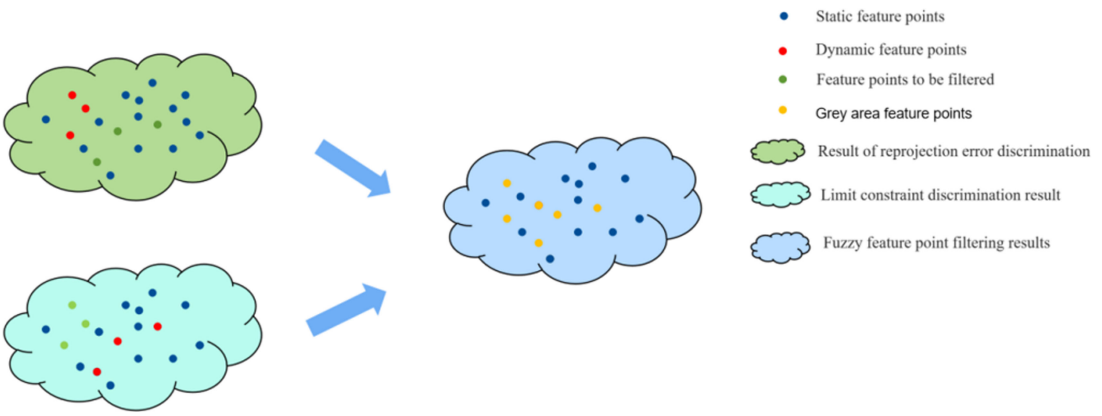

Figure 4. *Schematic diagram of filtering gray area feature points.*

The kernel line L2 in the current frame can be determined from the basic matrix F using Eq. (6):

$$L_2 = FP_1 = [u_1, v_1, 1]^T \tag{6}$$

where P_1 represents the feature point in the previous frame. u_1 and v_1 represent the horizontal and vertical coordinates of the point. The epipolar constraint is represented as follows:

$$P_2^T FP_1 = P_2^T L_2 = 0 \tag{7}$$

where P_2 represents the feature point in the current frame. The kernel line corresponding to the feature point P_1 of the previous frame in the current frame is l:

$$l = FP_1 = F \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{8}$$

where X, Y, and Z represent the 3D coordinates of the feature point. Then, the distance D from the feature point P_2 of the current frame to the kernel line l is:

$$D = \left| P_2^T FP_1 \right| / \sqrt{\|X\|^2 + \|Y\|^2} \tag{9}$$

3.4. Gray area feature points recognition strategy

As shown in Figure 4, dynamic feature points are extracted from image frames containing target detection frames by gray area feature point filtering technique. The core of the algorithm is to determine the dynamic feature points for each point in order to generate the grey area feature point set M.

At the beginning of the algorithm, dynamic feature point labels are stored based on reprojection error and epipolar geometry constraints. Reprojection error is a standard used to assess the dynamics of a point by comparing the difference between the actual observed point and the predicted point calculated through the camera model. While traversing all feature points, if the reprojection error is below a predetermined threshold, the point is labeled as a dynamic feature point and added to set S, otherwise, it is labeled as a static feature point.

Next, a second round of assessment is conducted on the same feature points using epipolar geometry constraints. This process also traverses all points, and if they satisfy the geometric constraint conditions, they are labeled as dynamic feature points and added to set F. Through these two assessments, sets S and F will each contain labels indicating the dynamics of the corresponding feature points. After completing the dynamic feature point assessment, the algorithm proceeds to the filtering stage. By comparing the labels in sets S and F, the algorithm determines which points are gray area feature points. Specifically, if a point has inconsistent labels in sets S and F, or is labeled as a static feature point in set S, it is saved to

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

Algorithm 1 Grey area feature point filtering

Input: frames with object detection boxes
Output: grey area feature point set M

```
Initialize sets S and F
for each point in points:
  Judging whether it is a dynamic feature point through reprojection error:
    if True:
      S.append(1)
    else:
      S.append(0)
for each point in points:
  Determine whether it is a dynamic feature point through extreme geometric constraints:
    if True:
      F.append(1)
    else:
      F.append(0)
Filter grey area feature points
for i from 0 to length(points) - 1:
  if S[i] != F[i] or S[i] == 0:
    Save this feature point
  end if
end for
```

the final gray area feature point set M. Algorithm 1 summarizes the specific steps for gray area feature point filtering.

3.5. Gray area feature point static probability calculation

Input the gray area feature point to be determined, calculate its static probability based on its motion and geometric relationship, and determine whether to retain the feature point.

First, calculate the first probability value based on motion estimation distance, calculate the movement distance Dis_a between the back projection point P_a and the corresponding mapping point x_a:

$$Dis_a = \sqrt{\left[\left(X_a^2 - X_a'^2\right) + \left(Y_a^2 - Y_a'^2\right) + \left(Z_a^2 - Z_a'^2\right)\right]} \tag{10}$$

$[X_a^{'}, Y_a^{'}, Z_a^{'}]^T$ is the 3D point coordinates of the map points x_a.

Based on Eq. (10), calculate the distances between the corresponding points in the reference frame and the key points on the map, obtain a series of distances, and then calculate the variance S_d and mean μ_d:

$$\mu_d = \sum_{a=1}^{N_a} Dis_a / N_a \tag{11}$$

$$S_d = \sqrt{\sum_{a=1}^{N_a} \left(Dis_a - \mu_d\right)^2 / N_a} \tag{12}$$

where N_a is the total number of points. Using the mean μ_d and variance S_d obtained from Eqs. (11) and (12), we can calculate the static observation weight W_a based on motion estimation for each feature

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

point in the current frame, as follows:

$$W_a = 1/\left(1 + \sqrt{-\beta\left[\left(Dis_a - \mu_d\right)/S_d\right]}\right) \tag{13}$$

where β is a constant used to adjust the sensitivity of the weight calculation formula.

Meanwhile, if the feature point is a static feature point, the number of times it is judged as a static feature point will be very large. Therefore, we further compute the observation count $V_{st}(p_a)$ for each feature point in the current frame using the reprojection error calculation method, with the initial value set to 0. Specifically, from the first frame to the current frame, utilizing the discriminative method outlined in Eq. (3), if a feature point p_a is observed in a frame and determined to be a static feature point through the reprojection error method, its count value is updated as follows:

$$V_{st}\left(p_a\right) = V_{st}\left(p_a\right) + 1 \tag{14}$$

If the feature point p_a is observed but is not classified as a static feature point, then the count value, $V_{st}(p_a)$ of the feature point p_a is updated as follows:

$$V_{st}\left(p_a\right) = V_{st}\left(p_a\right) - 1 \tag{15}$$

If the feature point p_a is not observed, no update $V_{st}(p_a)$. Then, the mean and standard deviation of the $V_{st}(p_a)$ for the current frame are calculated.

$$\mu_v = \sum_{a=1}^{N_v} V_{st}\left(p_a\right)/N_v \tag{16}$$

$$S_v = \sqrt{\sum_{a=1}^{N_v}\left(V_{st}\left(p_a\right) - \mu_v\right)^2/N_v} \tag{17}$$

where N_v is the number of feature points in the current frame.

By using the mean μ_v and standard deviation S_v of static observations, we can calculate the static observation weight of each feature point in the current frame, as follows:

$$W_{V_{st}(p_a)} = 1/\left(1 + \sqrt{-\beta_v\left[\left(V_{st}\left(p_a\right) - \mu_v\right)/S_v\right]}\right) \tag{18}$$

where β_v is a constant greater than 0.

The static weight of the static point p_a based on the reprojection error is represented as follows:

$$W_{bre} = W_a + \alpha_{st} W_{V_{st}(p_a)} \tag{19}$$

where α_{st} is a real number greater than 0.

Second, the second probability value is calculated based on the epipolar geometry constraints. According to Figure 3, if the feature point P is static, its projected point in the current frame should lie on the epipolar line L2. Conversely, if the point is moving, it will not be located on the epipolar line. A similar calculation is performed to determine the epipolar distance from the feature points in the current frame to the corresponding epipolar line:

$$H = \sqrt{\left(Au_2 + Bv_2 + C\right)^2/\left(A^2 + B^2\right)} \tag{20}$$

where H represents the distance from the feature point to the epipolar line L2 in the current frame, and A, B, C are the parameters of the epipolar line equation, u_2 and v_2 are the coordinates of the current feature point.

Then, calculate the mean μ_H and variance S_H, and then calculate a static weight:

$$\mu_H = \sum_{a=1}^{N_a} H/N_a \tag{21}$$

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

$$S_H = \sqrt{\sum_{a=1}^{N_a} (H - \mu_H)^2 / N_a} \tag{22}$$

$$W_b = 1/\left(1 + \sqrt{-\beta \left[(H - \mu_H)/S_H\right]}\right) \tag{23}$$

where N_a is the total number of points, β is a constant greater than 0.

Similarly, based on the discrimination results of polar geometric constraints, update the observation frequency $V'_{st}(p_a)$, and then calculate a probability value. The formula steps are as follows:

$$V'_{st}(p_a) = V'_{st}(p_a) + 1 \tag{24}$$

$$V'_{st}(p_a) = V'_{st}(p_a) + 1 \tag{25}$$

$$\mu_v = \sum_{a=1}^{N_v} V'_{st}(p_a) / N_v \tag{26}$$

$$S_v = \sqrt{\sum_{a=1}^{N_v} \left(V'_{st}(p_a) - \mu_v\right)^2 / N_v} \tag{27}$$

$$W_{V'_{st}(p_a)} = 1/\left(1 + \sqrt{-\beta \left[\left(V'_{st}(p_a) - \mu_v\right)/S_v\right]}\right) \tag{28}$$

where μ_v is the mean of the observation frequency for static points, S_v is the standard deviation of the observation frequency for static points, and N_v is the number of samples, β is a constant greater than 0.

The static weight representation W_{egc} of static points based on extreme geometric constraints is as follows:

$$W_{egc} = W_b + \beta_{st} W_{V'_{st}(p_a)} \tag{29}$$

Among them, β_{st} is a real number greater than 0.

Finally, by combining the results W_{bre} and W_{egc} obtained from Eqs. (18) and (28), the final static weights of the feature points are calculated and published as follows:

$$W_{st} = \varphi W_{bre} + \omega W_{egc} \tag{30}$$

where φ and ω are real number greater than 0.

Finally, the resulting static weight values are used to determine whether they need to be eliminated or not. By static probability, more static feature points are retained. The specific steps are as in Algorithm 2:

3.6. Fusion of weighted static probabilities and spatio-temporal constraints for dynamic position estimation

In VSLAM systems, pose estimation is one of the core aspects to achieve accurate localization and map building. However, moving objects in dynamic environments can have an impact on the reliability of the feature points, thus reducing the accuracy and robustness of the pose estimation. In order to improve the adaptability of pose estimation, this paper proposes a dynamic pose estimation method that incorporates weighted static probabilities and spatio-temporal constraints. The weighting weights are dynamically adjusted by integrating the motion velocity variations and spatial consistency characteristics of the feature points, which in turn optimizes the robustness of the pose estimation.

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

Algorithm 2 Grey area feature point static probability calculation

Input: Grey area feature point set M, thresholds σ
Output: Static feature point set S_{static}

```
Initialize sets P_s = [ ], S_static = [ ]
for each point m ∈ M:
    Calculate motion-based distance d_m
    Compute motion variance μ_d, S_d
    Compute motion-based probability W_a according to Eq. (13);
    Statistically observe state parameters V_st(p_a) and conduct statistical analysis W_V_st(p_a)
    Compute probability W_bre according to Eq. (19)
    Calculate geometric-based distance H_m
    Compute geometric variance μ_H, S_H
    Compute geometric-based probability W_b
    Statistically observe state parameters V_st'(p_a) and conduct statistical analysis W_V_st'(p_a)
    Compute probability W_egc according to Eq. (29)
    Combine static probability according to Eq. (30)
    if W_st > σ :
        S_static.append(m)
    end if
end for
```

Specifically, the dynamic weight model based on static probability and velocity change is first established, and the combined static probability W_{st} and dynamic velocity information $\|\Delta v\|$ are weighted to establish dynamic weights $W_{st}^{'}$; second, time continuity and spatial smoothness constraints are introduced in the pose estimation. Finally, the improved objective function is constructed to minimize the weighted sum of the reprojection error and the spatio-temporal consistency error, and the optimal position matrix is solved iteratively using the Gauss-Newton method. Detailed steps are as follows:

Static probabilities are calculated for all feature points in the current frame, and the static probabilities of the feature points are weighted as weights, while the motion information (velocity changes of the feature points) is combined with the static probability weights to dynamically adjust the weight allocation. The adaptive weighting formula for incorporating speed changes is:

$$W_{st}^{'} = \alpha W_{st} + \beta \left[1 - (\|\Delta v\| \,/\, \|\Delta v\|_{max})\right] \tag{31}$$

where α *and* β denote the adaptive parameters that determine the relative weights of static probabilities and dynamic information. To ensure the optimality of the parameter settings, Bayesian optimization is used to automatically adjust α *and* β, which enables the dynamic weights to adaptively optimize the weight assignment of feature points in different scenarios,$\|\Delta v\|_{max}$ is the maximum value of the velocity change. Δv represents the inter-frame velocity change at the feature point, calculated as:

$$\|\Delta v\| = \left\| p_i^t - p_i^{t-1} \right\| / \Delta t \tag{32}$$

where p_i^t and p_i^{t-1} denote the positions of feature points in neighboring frames and Δt denotes the time interval between neighboring frames.

In dynamic environments, motion interference can lead to inconsistencies in feature point matching, which negatively impacts pose estimation accuracy. To address this issue, we introduce a spatio-temporal consistency constraint that minimizes the spatial error between feature points in consecutive frames, thereby enhancing the robustness of pose estimation. By computing the spatial consistency error between feature points in consecutive frames, we incorporate a spatio-temporal smoothing constraint $\|\Delta p_i\|$ into the objective function, optimizing the pose estimation results and solving for the optimal pose matrix

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

T_{cw}. The formula is as follows:

$$T_{cw} = \underset{R,t}{arg\,min} \sum_{i=1}^{n} \left(W_{st}^{'} \cdot \| KTP_i \|_2^2 + \gamma \, \| \Delta P_i \|^2 \right) \tag{33}$$

$$\| \Delta P_i \| = \left\| p_i^t - T_{cw}^{t-1} p_i^{t-1} \right\| \tag{34}$$

where K is the internal parameter matrix of the camera, P_i denotes the 3D map point corresponding to p_i of the feature point in the current frame,$W_{st}^{'}$ is the static probability weight of the feature point, n is the number of valid static map points in the current frame, p_i^t refers to the projection position of the feature point in the current frame, T_{cw}^{t-1} is the bit-pose transformation matrix of the previous frame, γ is the spatio-temporal smoothing coefficient, which is used for adjusting the influence of the reprojection error and the spatio-temporal consistency error, T_{cw} is the to-be-solved bit-position transformation matrix to be solved. And the optimal transformation, T_{cw} can be expressed as:

$$T_{cw} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{35}$$

where R is the rotation matrix of the current frame and t is the translation vector.

The pose matrix T_{cw} is optimized by Gauss-Newton method by taking the previous frame pose T_{cw}^{t-1} as the initial value of the current frame pose and constructing the residual function r_i to represent the residual value at each point:

$$r_i = \sqrt{W_{st}^{'}} \cdot \| KTP_i - p_i \| + \sqrt{\gamma} \cdot \| \Delta P_i \| \tag{36}$$

The Jacobi matrix A is constructed by taking the derivatives of the rotation R and translation t, respectively, where the derivative of the rotation matrix R is shown in Eq. (37), and the derivative with respect to the translation vector t is shown in Eq. (38), and ultimately, the Jacobi matrix A obtained by combining the rotational and translational derivatives can be expressed as Eq. (40):

$$\partial r_i / \partial R = K \cdot (\partial P_{cam} / \partial R) \tag{37}$$

$$\partial r_i / \partial t = K \cdot (\partial P_{cam} / \partial t) \tag{38}$$

$$P_{cam} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \tag{39}$$

$$J_i = \begin{bmatrix} \partial r_{i,x} / \partial R & \partial r_{i,x} / \partial t \\ \partial r_{i,y} / \partial R & \partial r_{i,y} / \partial t \end{bmatrix} = \begin{bmatrix} K_{1x} \cdot [P_{cam}]_\times & K_{1x} \\ K_{1y} \cdot [P_{cam}]_\times & K_{1y} \end{bmatrix} \tag{40}$$

where Jacobian matrix J_i for the i-th feature point represents, $\partial r_{i,x} / \partial R$ and $\partial r_{i,y} / \partial R$ as the derivatives of the residuals in the x/y directions with respect to the rotation matrix; $\partial r_{i,x} / \partial t$ and $\partial r_{i,y} / \partial t$ are the derivatives of the residuals in the x/y directions with respect to the translation vector. K_{1x} and K_{1y} represent the parameters of the corresponding rows in the camera intrinsic matrix. $[P_{cam}]_\times$ represents the skew-symmetric matrix of the vector P_{cam}.

Combined with the spatio-temporal consistency constraints, we further update the Jacobi matrix so as to construct a complete Jacobi matrix containing both the reprojection error and the spatio-temporal consistency error, J. The derivative of the spatio-temporal consistency error term $\| \Delta P_i \|$ is:

$$\partial \, \| \Delta P_i \| \, / \partial T_{cw} = \Delta P_i / \, \| \Delta P_i \| \tag{41}$$

Here, ΔP_i represents the positional difference of the feature point between two consecutive frames, and $\| \Delta P_i \|$ represents the magnitude of the change.

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

The corresponding Jacobi matrix $J_{\Delta P}$ is:

$$J_{\Delta P} = \left[\Delta P_x / \|\Delta P\| \quad \Delta P_y / \|\Delta P\| \quad \Delta P_z / \|\Delta P\| \right] \tag{42}$$

where ΔP represents the position change vector of a feature point between two consecutive frames, $\|\Delta P\|$ represents the magnitude of the vector ΔP, and ΔP_x,ΔP_y,ΔP_z are the components of ΔP along the x,y,z directions, respectively.

The complete Jacobi matrix J combining the reprojection error and the spatio-temporal consistency error is denoted as:

$$J = \begin{bmatrix} \partial r_{i,x}/\partial R & \partial r_{i,x}/\partial t \\ \partial r_{i,y}/\partial R & \partial r_{i,y}/\partial t \\ \partial \|\Delta P_i\| /\partial R & \partial \|\Delta P_i\| /\partial t \end{bmatrix} \tag{43}$$

where $r_{i,x}$ and $r_{i,y}$ represent the reprojection errors of the i-th feature point in the x and y directions of the image plane, respectively. The $\partial r_{i,x}/\partial R$ and $\partial r_{i,x}/\partial t$,$\partial r_{i,y}/\partial R$, and $\partial r_{i,y}/\partial t$ represent the derivatives of the residuals in the x and y directions with respect to the camera pose rotation R and translation t. $\|\Delta P_i\|$ is the spatio-temporal position difference of the i-th feature point between consecutive frames. $\partial\|\Delta P_i\|/\partial R$ and $\partial\|\Delta P_i\|/\partial t$ represent the derivatives of the corresponding error with respect to rotation and translation, used to constrain the feature point's smoothness and matching stability.

Then, iterations are performed by Gaussian Newton method to solve for the parameter increments δ :

$$\delta = -\left(J^T J\right)^{-1} J^T r \tag{44}$$

The J is the Jacobian matrix, J^T is the transpose of the Jacobian matrix J, and r is the residual vector, representing the reprojection error and spatio-temporal consistency error under the current estimated pose.

Update pose matrix:

$$T_{cw} \leftarrow T_{cw} \cdot exp\,(\delta) \tag{45}$$

where $T_{cw}$is the pose transformation matrix of the current frame, and $exp(\delta)$ is the exponential map of the increment δ. By iterating to the maximum number of times, the optimal position matrix between the current frame and the map points is finally found, thus realizing the accurate matching and pose estimation of inter-frame features.

4. Experimental results

To evaluate the accuracy of our system, we conducted experiments using the publicly available TUM RGB-D dataset [41]. The TUM dataset was created by the Technical University of Munich and captures data using a Kinect sensor at a rate of 30 Hz, with an image resolution of 640 × 480. Simultaneously, a high-precision motion capture system, VICON, equipped with an inertial measurement system, was used to obtain camera position and orientation data, which can be approximated as the true position data of the RGB-D camera.

This paper focuses on experiments using four high-dynamic scene sequences and one low-dynamic sequence from the TUM RGB-D dataset. In the high-dynamic sequence, two people walk in front of or around a table, while in the low-dynamic sequence, two people sit in chairs, engaging in conversation and making slight gestures. For each type of dataset series, the camera motion was also categorized into four states: static (where the camera remains stationary), xyz (where the camera moves along the spatial X–Y–Z axes), rpy (where the camera rotates with roll, pitch, and yaw angles), and hemispherical (where the camera moves along a trajectory of a hemisphere with a 1 m diameter). Figure 5 shows the effect of the implementation of the algorithm and comparison.

Experiments were conducted on a computer system equipped with an Intel i5 CPU, Nvidia GeForce RTX 4060 Ti, 32 GB of RAM, and running the Ubuntu 18.04 operating system.

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

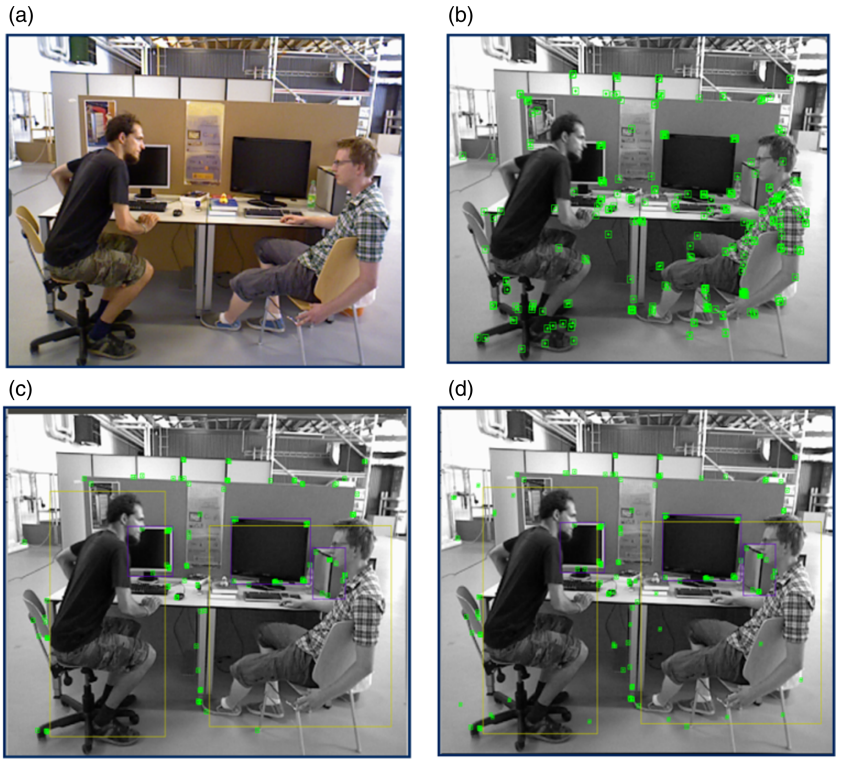

Figure 5. *Visual comparison display of algorithm effects. (a) Real scenes. (b) ORB-SLAM2. (c) ORB-SLAM2 under YOLO. (d) GAF-SLAM.*

4.1. Performance evaluation of TUM RGB-D dataset

Absolute trajectory error (ATE) and relative pose error (RPE) are commonly used metrics for evaluating the localization accuracy of VSLAM systems. ATE measures the overall discrepancy between the estimated trajectory and the ground-truth trajectory, while RPE focuses on assessing rotational and translational drift between consecutive frames. To assess the performance improvement of our proposed GAF-SLAM over ORB-SLAM2 and ORB-SLAM3, we conducted comparative experiments on the TUM RGB-D dataset. The results report the root mean square error (RMSE), mean error (Mean), and standard deviation (Std) for both ATE and RPE.

As shown in Tables I, II, III, and Figure 6, GAF-SLAM achieves significantly better accuracy and robustness than ORB-SLAM2 and ORB-SLAM3, particularly under scenarios with substantial dynamic interference. Even in low-dynamic settings – such as when a person remains nearly stationary in a chair – our method demonstrates noticeable improvements. As illustrated in Figure 7, where the estimated trajectory is shown in black, the ground-truth trajectory in blue, and the deviation in red, GAF-SLAM consistently aligns well with the real trajectory, confirming its strong adaptability to highly dynamic environments.

As shown in Table IV, to further evaluate the effectiveness of the proposed algorithm, GAF-SLAM is compared with several state-of-the-art dynamic visual SLAM systems, including Dyna-SLAM, YOLO-SLAM, SG-SLAM, and MPOC-SLAM. Since some of these methods are not open-sourced or reproducible under a unified hardware platform, the reported results are obtained from the corresponding original publications. Although differences in hardware settings may introduce certain deviations in performance metrics, this comparison is intended to provide a general reference for accuracy trends across different methods. As shown in the results, GAF-SLAM consistently demonstrates superior performance among all evaluated approaches, achieving the lowest trajectory error in highly dynamic sequences such

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

Table I. *Results of the metric absolute trajectory error (ATE)[M].*

Sequences	ORB-SLAM2			ORB-SLAM3			Ours		
	RMSE	**Mean**	**SD**	**RMSE**	**Mean**	**SD**	**RMSE**	**Mean**	**SD**
Fr3_s_static	0.0087	0.0078	0.0039	0.0157	0.0132	0.0045	0.0049	0.0041	0.0029
Fr3_w_half	0.5979	0.5864	0.2665	0.2351	0.2198	0.1705	0.0205	0.0186	0.0143
Fr3_w_xyz	0.6691	0.6332	0.2916	0.2881	0.2521	0.1910	0.0151	0.0135	0.0101
Fr3_w_static	0.5016	0.4112	0.1930	0.2812	0.2188	0.1130	0.0054	0.0042	0.0022
Fr3_w_rpy	0.8191	0.7245	0.4283	0.6041	0.5161	0.4003	0.0139	0.0135	0.0137

Table II. *Results of the metric relative translation error (RTE)[M/S].*

Sequences	ORB-SLAM2			ORB-SLAM3			Ours		
	RMSE	**Mean**	**SD**	**RMSE**	**Mean**	**SD**	**RMSE**	**Mean**	**SD**
Fr3_s_static	0.0093	0.0082	0.0044	0.0191	0.0184	0.0039	0.0045	0.0040	0.0032
Fr3_w_half	0.4136	0.2950	0.3483	0.0380	0.0246	0.0306	0.0272	0.0245	0.0144
Fr3_w_xyz	0.3729	0.2738	0.2393	0.0386	0.0284	0.0263	0.0152	0.0139	0.0105
Fr3_w_static	0.2259	0.1512	0.2151	0.0276	0.0196	0.0213	0.0083	0.0076	0.0035
Fr3_w_rpy	0.4439	0.2745	0.2923	0.2409	0.1894	0.2278	0.0298	0.0241	0.0155

Table III. *Results of the metric relative rotation error (RRE)[DEG/S].*

Sequences	ORB-SLAM2			ORB-SLAM3			Ours		
	RMSE	**Mean**	**SD**	**RMSE**	**Mean**	**SD**	**RMSE**	**Mean**	**SD**
Fr3_s_static	0.2899	0.2606	0.1271	0.4795	0.3845	0.1797	0.1541	0.1312	0.0821
Fr3_w_half	7.9219	4.4695	6.5406	6.7369	4.5141	5.5049	0.4211	0.3412	0.2341
Fr3_w_xyz	7.1415	5.6403	4.3804	6.7826	4.8465	5.5083	0.4127	0.2712	0.2751
Fr3_w_static	3.8068	1.6993	3.4065	3.5224	1.5224	3.3677	0.1662	0.1436	0.0844
Fr3_w_rpy	7.9249	4.4695	6.5406	8.8879	5.5125	4.5746	0.3935	0.3454	0.2141

as "rpy" and "static," and maintaining competitive accuracy in the other scenarios. Specifically, the bolded data in the table represent the best performance achieved in each sequence.

4.2. Performance evaluation of Bonn RGB-D dataset

The Bonn RGB-D Dynamic dataset [42], released by Bonn University in 2019, aims to evaluate the performance of RGB-D SLAM and contains 24 dynamic sequences. In order to test the generalization ability of the dynamic feature rejection algorithm, we conducted further experiments on this dataset, and selected seven representative sequences for analysis, including three datasets of the “crowd” series, two datasets of the “person” series, and two datasets of the “synchronous” series and two datasets of “synchronous” series. The dataset of the “crowd” sequence shows three people moving freely in a room; the “person” sequence mainly shows the camera following the walker; and the “synchronous” sequence shows several people repeatedly jumping in the same direction.

To systematically evaluate the localization accuracy and robustness of GAF-SLAM, we conducted comparative experiments with two classical SLAM frameworks: ORB-SLAM2 and ORB-SLAM3. For SG-SLAM, the results were directly cited from the original publication. As summarized in Table V, GAF-SLAM achieves the best performance across all seven test sequences, significantly outperforming the compared methods. These results strongly demonstrate the robustness, localization accuracy, and generalization capability of GAF-SLAM in complex dynamic environments.

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

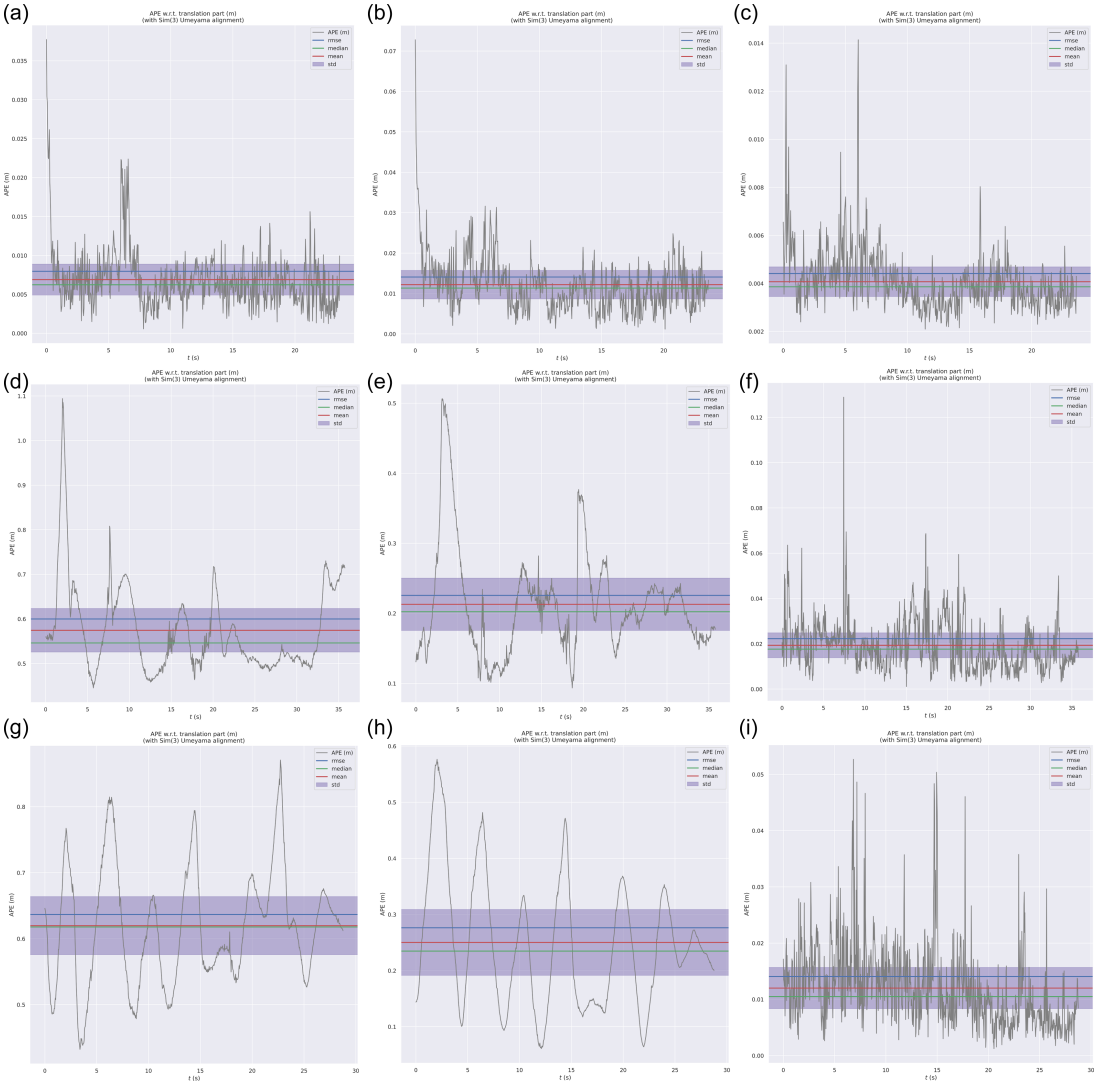

Figure 6. *ATE results of ORB-SLAM2, ORB-SLAM3, and our proposed system across five dynamic scene sequences. (a–c) Fr3_s_static; (d–f) Fr3_w_half; (g–i) Fr3_w_xyz; (j–l) Fr3_w_static; (m–o) Fr3_w_rpy.*

4.3. Ablation experiment

To validate the effectiveness of each module, we performed ablation experiments. As shown in Table VI, the experimental results of ATE for different modules show that each module plays an important role in improving the system performance. In the experiments, we set up three different configurations to gradually evaluate the effect of the module: First, Ours(Y) uses a YOLO-Point network instead of the traditional ORB feature extraction method to perform deep learning feature extraction only at the front-end. Second, based on YOLO-Point, the static probability calculation of gray area feature points is further introduced, and the static weights are incorporated into the bit-pose estimation optimization, but not combined with the optimized static weights and spatio-temporal constraints. Finally, ours denotes our complete method, which combines YOLO-Point feature extraction with static probability computation, screening static feature points in the detection frame by reprojection error and polar

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

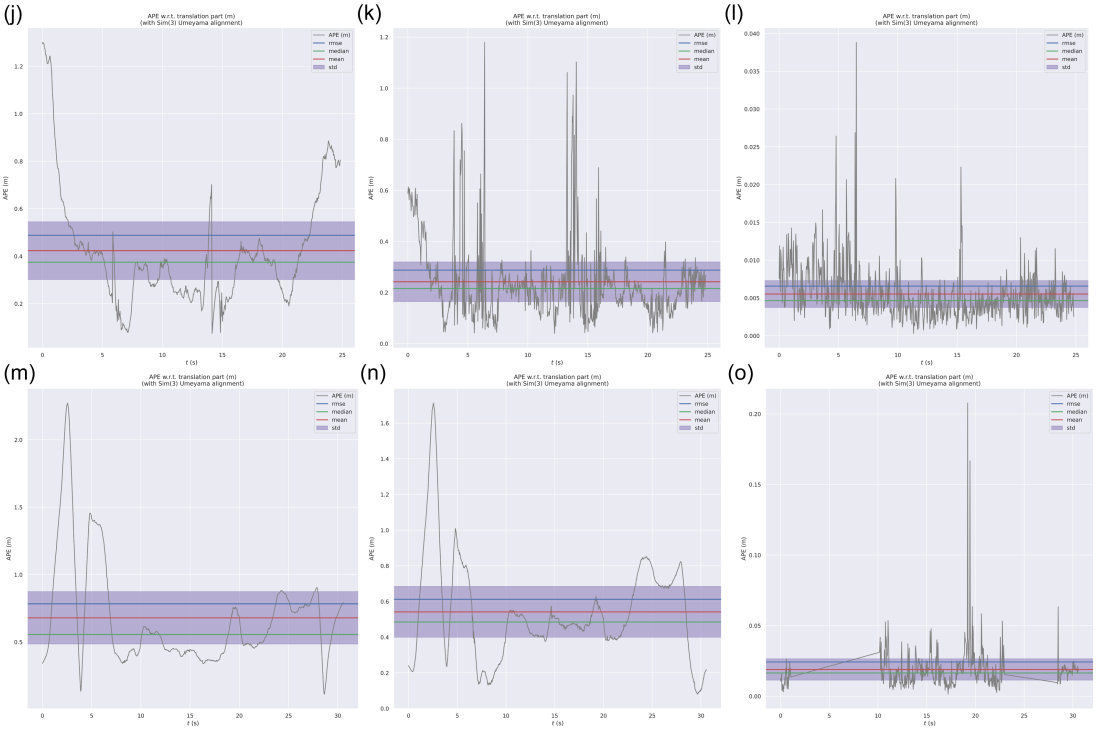

Figure 6. *Continued*

geometry constraints, while excluding dynamic feature points, and ultimately combining static weights with spatio-temporal constraints for optimized bit-pose estimation.

The experimental results show that the addition of each module significantly reduces the ATE value, indicating that feature screening and static probability computation effectively improve the localization accuracy of the system in dynamic environments. In particular, the complete method (Ours) has the most superior performance, which is able to maximize the rejection of dynamic feature interference while preserving static features, thus achieving accurate bit-pose estimation in highly dynamic scenes. Overall, the results of the ablation experiments validate the effectiveness of the individual modules and confirm the advantages of our proposed method for processing dynamic features in dynamic SLAM systems.

4.4. Time analysis

Real-time is also one of the important evaluation metrics for SLAM systems; therefore, we tested the time consumption of the system and compared it with five other algorithms as shown in Table VII. Dyna-SLAM uses Mask-R-CNN for pixel-level semantic segmentation, so its average time cost per frame processed is very high, and YOLO-SLAM, SG-SLAM, and MPOC-SLAM meet the real-time requirements while improving the accuracy. And GAF-SLAM consumes only 52.19 ms, so it can meet the real-time requirements of mobile robots while improving the localization accuracy.

In addition, the run-time performance of individual modules within the GAF-SLAM framework was evaluated, as summarized in Table VIII. Specifically, Module A represents the YOLO-Point module, which simultaneously generates keypoints and object detection bounding boxes. Module B denotes the gray area feature recognition module, while Module C is responsible for static probability estimation. Module D performs dynamic point removal and pose estimation, and Module E manages keyframe insertion and local mapping. The results confirm that GAF-SLAM satisfies real-time processing requirements, even in dynamic environments characterized by dense motion interference.

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

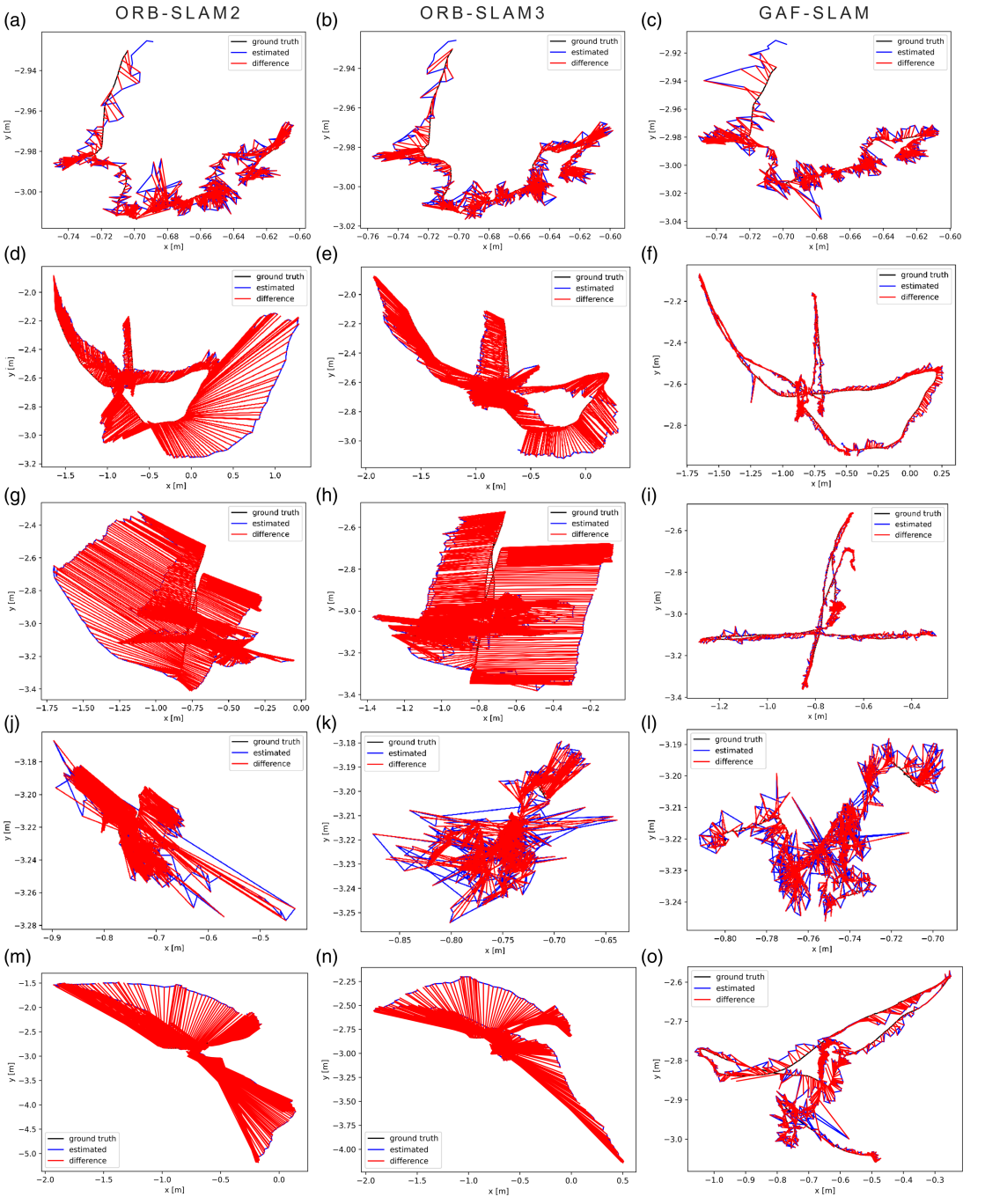

Figure 7. *Visualization of the differences between the estimated and ground-truth trajectories for ORB-SLAM2, ORB-SLAM3, and our proposed system across five dynamic scene sequences. (a–c) Fr3_s_static; (d–f) Fr3_w_half; (g–i) Fr3_w_xyz; (j–l) Fr3_w_static; (m–o) Fr3_w_rpy.*

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

Table IV. *Comparison of the absolute trajectory error (ATE)[M]*

Sequences	Dyna-SLAM		YOLO-SLAM		SG-SLAM		MPOC-SLAM		Ours	
	RMSE	SD	RMSE	SD	RMSE	SD	RMSE	SD	RMSE	SD
Fr3_s_static	0.0067	0.0028	0.0066	0.0033	0.0060	**0.0029**	0.0055	0.0031	**0.0049**	**0.0029**
Fr3_w_half	0.0296	0.0157	0.0283	0.0138	0.0268	0.0134	0.0222	**0.0114**	**0.0205**	0.0143
Fr3_w_xyz	0.0164	0.0086	0.0146	0.0070	0.0152	0.0075	**0.0137**	**0.0068**	0.0151	0.0101
Fr3_w_static	0.0068	0.0032	0.0073	0.0035	0.0073	0.0034	0.0055	0.0024	**0.0054**	**0.0022**
Fr3_w_rpy	0.0354	0.0190	0.2164	0.1001	0.0324	0.0187	0.0265	0.0145	**0.0139**	**0.0137**

Table V. *Comparison of the absolute trajectory error (ATE) [M].*

Sequences	ORB-SLAM2		ORB-SLAM3		SG-SLAM		GAF-SLAM	
	RMSE	SD	RMSE	SD	RMSE	SD	RMSE	SD
Crowd	0.8632	0.5918	0.5691	0.4216	0.0234	0.0143	**0.0135**	**0.0139**
Crowd2	1.3573	0.6207	1.2412	0.5445	0.0584	0.0406	**0.0247**	**0.0318**
Crowd3	1.0772	0.3823	0.9394	0.4012	0.0319	0.0219	**0.0135**	**0.0219**
Person_tracking	0.7959	0.3617	0.4012	0.2891	0.0400	0.0139	**0.0385**	**0.0126**
Person_tracking2	1.0679	0.8732	0.7071	0.5118	0.0376	0.0154	**0.0365**	**0.0144**
Synchronous	1.1411	0.5703	0.7761	0.4622	0.3229	0.1824	**0.0165**	**0.0184**
Synchronous2	1.4069	0.4864	1.0361	0.5193	0.0164	0.0126	**0.0064**	**0.0115**

Table VI. *Results of metric ATE.*

Sequences	Ours(Y)			Ours(Y + P)			Ours		
	RMSE	Mean	S.D.	RMSE	Mean	S.D.	RMSE	Mean	S.D.
Fr3_s_static	0.0076	0.0062	0.0039	0.0073	0.0059	0.0035	**0.0049**	**0.0041**	**0.0029**
Fr3_w_half	0.0280	0.0219	0.0149	0.0279	0.0201	0.0144	**0.0205**	**0.0186**	**0.0143**
Fr3_w_xyz	0.0172	0.0124	0.0087	0.0160	0.0094	0.0081	**0.0151**	**0.0135**	**0.0101**
Fr3_w_static	0.0075	0.0058	0.0048	0.0071	0.0052	0.0030	**0.0054**	**0.0042**	**0.0022**
Fr3_w_rpy	0.0367	0.0315	0.0257	0.0209	0.0196	0.0138	**0.0139**	**0.0135**	**0.0137**

Table VII. *Time evaluation.*

Systems	Tims (ms)	Hardware platform
ORB-SLAM2	25.58	i5 CPU, 32 GB RAM, Nvidia GeForce RTX 4060Ti
ORB-SLAM3	27.41	i5 CPU, 32 GB RAM, Nvidia GeForce RTX 4060Ti
GAF-SLAM(Ours)	52.19	i5 CPU, 32 GB RAM, Nvidia GeForce RTX 4060Ti
Dyna-SLAM	235.98(at least)	Nvidia Tesla M40 GPU
YOLO-SLAM	696.09	I5-4288U CPU
SG-SLAM	65.71	Nvidia Jetson AGX Xavier Developer Kit
MPOC-SLAM	99.03	i7-10870H CPU, RTX3060 Laptop GPU

4.5. Real environment experiment

To further evaluate the practicality of our system, we conducted experiments using a monocular camera in a real-world environment. During the experiments, we performed camera panning and rotation while asking the person in front of the camera to perform actions such as standing up, walking around the chair, and leaving the camera's field of view.

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

Table VIII. *The average run time of different modules.*

Moudle	A	B	C	D	E	Total
Time (ms)	19.41	6.12	6.47	10.83	9.36	52.19

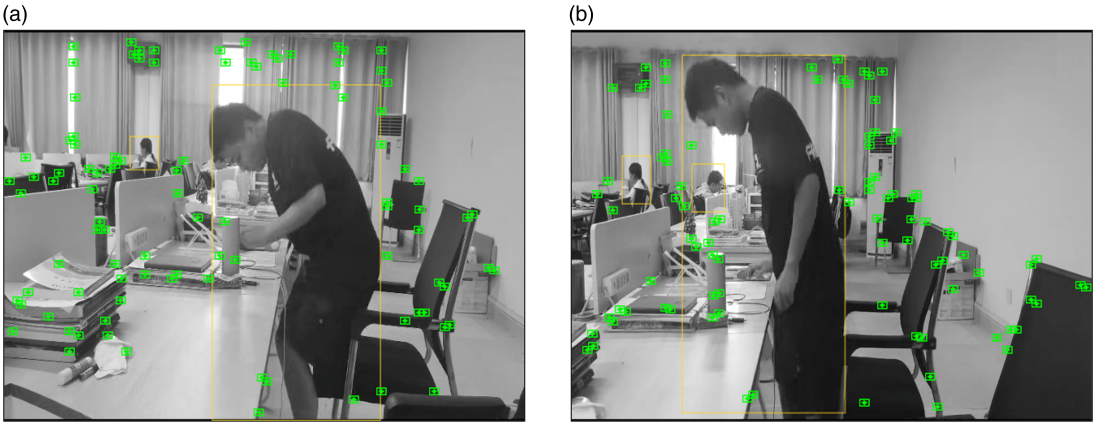

Figure 8. *The effect of preserving static feature points in real scenes.*

Figure 8 demonstrates the effectiveness of our algorithm in removing dynamic feature points while retaining more static feature points in real-world scenarios. It is evident that we successfully preserved the static feature points within the feature frame. The experimental results indicate that the presence of dynamic objects leads to significant differences in the estimated trajectory lengths. As shown in Figure 9, we compared the trajectory estimation results of the GAF-SLAM system with those of ORB-SLAM2. When the surrounding environment is static, both systems perform well. However, when dynamic objects are present and moving in the environment (highlighted areas in Figure 9), ORB-SLAM2 experiences considerable jitter, while our system maintains consistency with the ground-truth trajectory and remains unaffected by the dynamic objects.

5. Conclusion and future work

In this paper, we introduce GAF-SLAM, an optimization method for visual SLAM systems in dynamic environments. Based on the ORB-SLAM2 framework, we realize the effective fusion of dynamic object detection, feature point screening and weighted pose estimation by integrating the YOLO-Point deep learning network with the static probability computing framework. Specifically, we propose a method for selecting gray area feature points based on reprojection errors and polar geometric constraints, which enables the system to retain potential static points within the dynamic detection region. In addition, we develop a new static probability calculation method for gray area feature points to further improve the determination accuracy of static feature points through the static probability scoring mechanism in order to enhance the retention of static information. Finally, the proposed weighted static probability and spatio-temporal constraint pose estimation algorithm effectively reduces the interference of dynamic points on pose estimation, thus significantly improving the posing accuracy and robustness of the system. Experimental results based on the TUM RGB-D dataset and the Bonn RGB-D Dynamic dataset show that our method is significantly more accurate in highly dynamic scenes.

Despite these encouraging results, one of the current limitations of our method lies in its reliance on a static probability model that depends heavily on multi-frame visual consistency. In environments with drastic illumination changes – such as sudden light switching, dynamic lighting conditions, or natural light interference – the same spatial location may exhibit significant appearance variations across frames.

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

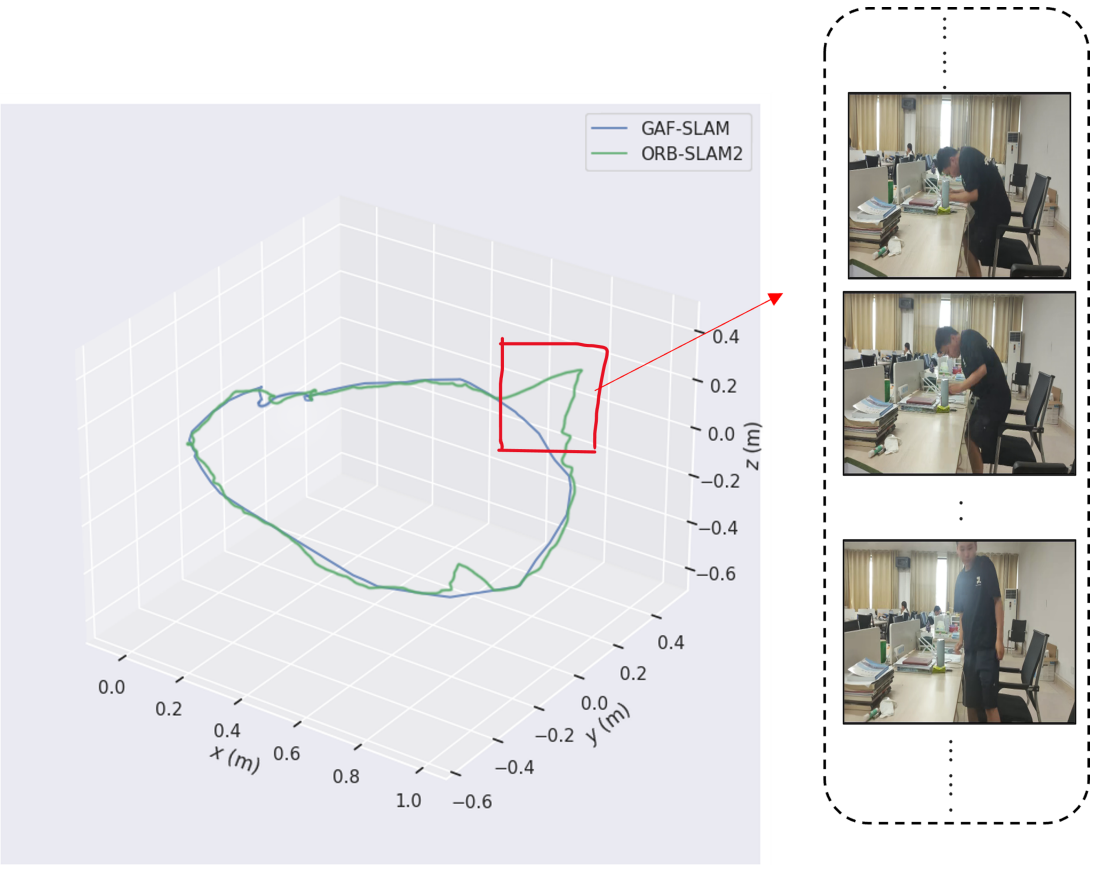

Figure 9. *The contrast of trajectories obtained from ORB-SLAM2 and our system in real environment.*

This can lead to the misclassification of dynamic regions as static, ultimately introducing noise into the map and degrading SLAM performance. To address this issue, future work will explore illumination-disentangled modeling techniques inspired by neural radiance fields (NeRF). By decoupling structural and appearance information, especially under varying lighting conditions, we aim to achieve more robust static-dynamic region separation and improve the reliability of feature selection in complex and unconstrained environments. In addition, we plan to extend the system to accommodate nonrigid motion patterns and improve the generalization ability of GAF-SLAM in more realistic and diverse scenarios.

Author contributions. Huilin Liu and Lunqi Yu conceived the method, built the framework, and conducted the theoretical study. Lunqi Yu designed and performed the experiments and wrote the article. Huilin Liu and Shenghui Zhao analyzed experimental data and assisted in revising the paper.

Financial support. This work was supported by the Scientific Research Foundation for High-level Talents of Anhui University of Science and Technology [grant number 2024yjrc52], the open Foundation of Anhui Engineering Research Center of Intelligent Perception and Elderly Care [grant number 2022OPB01], the National Natural Science Foundation of China [grant number 52374155] and the Anhui Provincial Natural Science Foundation [grant number 2308085MF218].

Competing interests. The authors declare no competing interests exist.

Ethical approval. None.

References

[1] H. M. Cho, H. Jo and E. Kim, "SP-SLAM: Surfel-point simultaneous localization and mapping," *IEEE/ASME Trans. Mechatron.* **27**(5), 2568–25792022.

[2] C. Li, X. Zhang, H. Gao, R. Wang and Y, "Fang "Bridging the gap between visual servoing and visual SLAM: A novel integrated interactive framework," *IEEE Trans. Autom. Sci. Eng.* **19**(3), 2245–2255 (2022).

[3] F. Nie, W. Zhang, Y. Wang, Y. Shi and Q. Huang, "A forest 3-D lidar SLAM system for rubber-tapping robot based on trunk center atlas," *IEEE/ASME Trans. Mechatron.* **27**(5), 2623–2633 (2022). doi: 10.1109/TMECH.2021.3120407.

[4] J. Cai, F. Yan, Y. Shi, M. Zhang and L. Guo, "Autonomous robot navigation based on a hierarchical cognitive model," *Robotica* **41**(2), 690–712 (2023). doi: 10.1017/S0263574722001539.

[5] C. Mulubika and K. Schreve, "An approach towards mobile robot recovery due to vision sensor failure in vSLAM systems using ROS," *Robotica* **1-23** (2025). doi: 10.1017/S0263574724002145. Published online.

[6] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans Robot.* **33**(5), 1255–1262 (2017). doi: 10.1109/TRO.2017.2705103.

[7] 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.* **37**(6), 1874–1890 (2021). doi: 10.1109/TRO.2021.3075644.

[8] J. Engel, T. Schöps and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," **In:** *Computer Vision – ECCV 2014*, (2014) pp. 834–84910.1007/978-3-319-10604-5_53.

[9] J. Engel, V. Koltun and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal.* **40**(3), 611–625 (2018). doi: 10.1109/TPAMI.2017.2658577.

[10] C. Yu, Z. Liu, X. J. Liu, F. Xie, Y. Yang, Q. Wei and Q. Fei. DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments. **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018) pp. 1168–1174.

[11] B. Bescos, J. M. Fácil, J. Civera and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.* **3**(4), 4076–4083 (2018). doi: 10.1109/LRA.2018.2860039.

[12] Z. Xing, X. Zhu and D. Dong, "DE-SLAM: SLAM for highly dynamic environment," *J. Field Robot.* **39**, 528–542 (2022).

[13] A. Backhaus, T. Luettel and H. J. Wuensche, "YOLOPoint joint keypoint and object detection. arXiv:abs/2402.03989 (2024).

[14] A. J. Davison, I. D. Reid, N. D. Molton and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal.* **29**(6), 1052–1067 (2007). doi: 10.1109/TPAMI.2007.1049.

[15] X. Liu, S. Wen and H. Zhang, "A real-time stereo visual-inertial SLAM system based on point-and-line features," *IEEE Trans. Veh. Technol.* **72**(5), 5747–5758 (2023). doi: 10.1109/TVT.2022.3233721.

[16] X. Liu, S. Wen, J. Zhao, T. Z. Qiu and H. Zhang, "Edge-assisted multi-robot visual-inertial SLAM with efficient communication," *IEEE Trans. Autom. Sci. Eng.* **22**, 2186–2198 (2025). doi: 10.1109/TASE.2024.3376427.

[17] D. Zou and P. Tan, "CoSLAM: Collaborative visual SLAM in dynamic environments," *IEEE Trans. Pattern Anal.* **35**(2), 354–366 (2013). doi: 10.1109/TPAMI.2012.104.

[18] R. Wang, W. Wan, Y. Wang and K. Di, "A new RGB-D SLAM method with moving object detection for dynamic indoor scenes," *Remote Sens.* **11**, 1143 (2019).

[19] W. Dai, Y. Zhang, P. Li, Z. Fang and S. Scherer, "RGB-D SLAM in dynamic environments using point correlations," *IEEE Trans. Pattern Anal.* **44**(1), 373–389 (2022). doi: 10.1109/TPAMI.2020.3010942.

[20] 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.* **72**, 1–11 (2023). doi: 10.1109/TIM.2023.3280533.

[21] M. Hahsler, M. Piekenbrock and D. Doran, "Dbscan: Fast density-based clustering with R," *J. Stat. Softw.* **91**(1), 1–30 (2019).

[22] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," *IEEE Trans. Pattern Anal.* **42**(2), 386–397 (2020). doi: 10.1109/TPAMI.2018.2844175.

[23] X. Yang, Z. Yuan, D. Zhu, C. Chi, K. Li and C. Liao, "Robust and efficient RGB-D SLAM in dynamic environments," *IEEE Trans. Multimedia* **23**, 4208–4219 (2021). doi: 10.1109/TMM.2020.3038323.

[24] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal.* **39**(6), 1137–1149 (2017). doi: 10.1109/TPAMI.2016.2577031.

[25] C. M. Yu, K. C. Chen, C. T. Chang and Y. W. Ti, "SegNet: A network for detecting deepfake facial videos," *Multimedia Syst.* **28**, 793–814 (2022).

[26] S. Wen, X. Li, X. Liu, J. Li, S. Tao, Y. Long and T. Qiu, "Dynamic SLAM: A visual SLAM in outdoor dynamic scenes," *IEEE Trans. Instrum. Meas.* **72**, 1–11 (2023). doi: 10.1109/TIM.2023.3317378.

[27] J. He, M. Li, Y. Wang and H. Wang, "OVD-SLAM: An online visual SLAM for dynamic environments," *IEEE Sens. J.* **23**(12), 13210–13219 (2023).

[28] 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.* **34**, 6011–6026 (2022).

[29] 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 Sens J.* **23**(21), 26279–26291 (2023).

[30] 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.* **72**, 1–12 (2023). doi: 10.1109/TIM.2022.3228006.

[31] Q. Sun, W. Liu, J. Zou, Z. Xu and Y. Li, "GGC-SLAM: A VSLAM system based on predicted static probability of feature points in dynamic environments," *SIViP* **18**, 7053–7064 (2024).

https://doi.org/10.1017/S0263574725102701 Published online by Cambridge University Press

[32] Q. U. Islam, H. Ibrahim, P. K. Chin, K. Lim, M. Z. Abdullah and F. Khozaei, "Advancing real-world visual SLAM: Integrating adaptive segmentation with dynamic object detection for enhanced environmental perception," *Expert Syst Appl* **255**(Part A), 124474 (2024).
[33] F. Fu, J. Yang, J. Ma and J. Zhang, "Dynamic visual SLAM based on probability screening and weighting for deep features," *Measurement* **236**, 115127 (2024).
[34] S. Wu, X. Zhang, S. Zhang, Z. Song, R. Wang and J. Yuan, "MPOC-SLAM: An RGB-D SLAM system with motion probability and object category in high dynamic environments," *IEEE/ASME Transactions on Mechatronics* (2024). doi: 10.1109/TMECH.2024.3410508.
[35] D. DeTone, T. Malisiewicz and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. **In:** *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2018) pp. 224–236.
[36] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," **In:** *Computer Vision – ECCV* (Springer, Zurich, 2014) pp. 740–755.
[37] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.* **32**(11), 1231–1237 (2013).
[38] X. Shi, D. Li, P. Zhao, Q. Tian, Y. Tian, Q. Long, C. Zhu, J. Song, F. Qiao, L. Song and Y. Guo. Are We Ready for Service Robots? The OpenLORIS-Scene Datasets for Lifelong SLAM. **In:** *2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France* (2020) pp. 3139–3145.
[39] R. Munoz-Salinas and R. Medina-Carnicer, "UcoSLAM: Simultaneous localization and mapping by fusion of keypoints and squared planar markers," *Pattern Recogn.* **101**, 107193 (2020).
[40] S. Baker and I. Matthews, "Lucas-kanade 20 Years on: A unifying framework," *Int. J. Comput. Vis.* **56**, 221–255 (2004).
[41] 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* (2012) pp. 573–580.
[42] E. Palazzolo, J. Behley, P. Lottes, P. Giguere and C. Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. **In:** *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019) pp. 7855–7862.

Cite this article: H. Liu, L. Yu and S. Zhao, "Integrating gray area feature points and static probability to enhance the performance of indoor dynamic visual simultaneous localization and mapping", Robotica. https://doi.org/10.1017/S0263574725102701