Robotica (2025), **43**, pp. 2527–2545
doi:10.1017/S0263574725101823

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

A dynamic SLAM system with YOLOv7 segmentation and geometric constraints for indoor environments

Yewei Shen and Xinguang Zhang

School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai, China
Corresponding author: Xinguang Zhang; Email: jixieyuan123456@163.com

Received: 19 December 2024; **Revised:** 22 April 2025; **Accepted:** 20 May 2025; **First published online:** 30 June 2025

Keywords: visual SLAM; YOLOv7; semantic segmentation; optical flow tracking

Abstract

With the rapid advancements in robotics and autonomous driving, SLAM (simultaneous localization and mapping) has become a crucial technology for real-time localization and map creation, seeing widespread application across various domains. However, SLAM's performance in dynamic environments is often compromised due to the presence of moving objects, which can introduce errors and inconsistencies in localization and mapping. To overcome these challenges, this paper presents a visual SLAM system that employs dynamic feature point rejection. The system leverages a lightweight YOLOv7 model for detecting dynamic objects and performing semantic segmentation. Additionally, it incorporates optical flow tracking and multiview geometry techniques to identify and eliminate dynamic feature points. This approach effectively mitigates the impact of dynamic objects on the SLAM process, while maintaining the integrity of static feature points, ultimately enhancing the system's robustness and accuracy in dynamic environments. Finally, we evaluate our method on the TUM RGB-D dataset and in real-world scenarios. The experimental results demonstrate that our approach significantly reduces both the root mean square error (RMSE) and standard deviation (Std) compared to the ORB-SLAM2 algorithm.

1. Introduction

In recent years, intelligent robots have become increasingly prevalent in everyday life, transforming both work and living environments. As a prerequisite for autonomous robot localization and navigation, simultaneous localization and mapping (SLAM) technology has garnered significant attention [1]. The fundamental goal of SLAM is to use sensor data (e.g., cameras, LIDAR) to determine the robot's position in real time and construct a detailed map of its surroundings in an environment with no a priori map. SLAM can be divided into two primary categories based on sensor types: visual SLAM and laser SLAM. Visual SLAM, in particular, offers advantages such as lower cost, rich visual information, and ease of integration with other sensors compared to laser SLAM.

Classical visual SLAM consists of five primary modules: sensor data acquisition, front-end visual odometry, back-end (nonlinear) optimization, loop closure, and map construction. The algorithms for visual odometry are mainly categorized into feature-point and direct methods [2]. The feature-based approach, known for its stability and resistance to lighting variations and dynamic objects, is the dominant method. Among them, ORB-SLAM1-3 [3–5] is a widely recognized classic visual SLAM algorithm based on the feature-point method, which relies on extracting ORB (Oriented FAST and Rotated BRIEF) features [6] for matching and position estimation, thus enabling camera localization and map construction. However, this method is computationally intensive and prone to losing useful information. In contrast, the direct method estimates camera motion based on pixel brightness information, eliminating the need to compute keypoints and descriptors, thereby avoiding some limitations of the feature-point approach. Examples of direct-method visual SLAM algorithms include LSD-SLAM [7], SVO [8], and DTAM [9].

© The Author(s), 2025. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

However, traditional visual SLAM algorithms typically rely on static environment assumptions to ensure robustness [10]. With the diversification of application scenarios, particularly in dynamic environments, these algorithms face significant challenges. The movement of dynamic objects – such as pedestrians and vehicles – often interferes with the localization and mapping accuracy of the SLAM system. These dynamic objects cause mismatches in feature points, potentially leading to position drift and map inconsistency, thus undermining system stability and robustness. Moreover, as the positions of dynamic objects change across image frames, accurately distinguishing between static and dynamic feature points while maintaining real-time performance remains a major challenge in current visual SLAM research [11].

To contextualize our contributions, we highlight that while YOLO-based models have been applied in SLAM for object detection (e.g., [12] uses YOLOv5 with geometric constraints), integrating a segmentation head into a lightweight YOLOv7 framework specifically for dynamic SLAM is, to our knowledge, a novel contribution in its combined approach. Recent studies like [13] focus on lightweight detection using models like YOLO-Pose-Regression (YPR) but lack semantic segmentation, which is critical for precise object boundary delineation needed to accurately filter dynamic feature points. This omission can hinder accuracy in complex environments where dynamic objects partially occlude static backgrounds. Our approach uniquely combines YOLOv7's [14] efficiency (further enhanced with Ghost convolution) with a segmentation head for fine-grained dynamic object identification and pixel-level feature rejection. This addresses both accuracy and computational constraints in dynamic settings, offering an advantage over methods relying solely on detection bounding boxes or computationally heavier segmentation networks like Mask R-CNN used in DynaSLAM [15].

To tackle these challenges, this paper introduces a dynamic feature-point rejection approach. Our system integrates a lightweight YOLOv7 target detection model with a segmentation head, optical flow tracking, and multi-view geometry [16] to accurately detect dynamic objects – defined as moving physical entities in the environment. By screening and rejecting dynamic feature points at the pixel level using segmentation masks, it minimizes their impact on SLAM localization and mapping while preserving valuable static points often contained within dynamic object bounding boxes. Additionally, the system generates 3D point cloud maps using only static feature points. The main contributions of this paper are as follows:

1. A novel visual SLAM system integrating lightweight detection, segmentation, and geometric constraints that improves robustness and accuracy in dynamic environments compared to baseline methods.
2. A lightweight YOLOv7-based detection network, enhanced with a segmentation head and optimized using Ghost convolution and CBAM attention, for efficient and precise pixel-level delineation of dynamic objects in SLAM.
3. A leakage detection and compensation method using Lucas-Kanade optical flow and multi-view geometry to identify and mitigate the impact of missed or potentially dynamic feature points.

This paper is organized as follows: Section 2 reviews the current state of dynamic SLAM research and its challenges. Section 3 presents the system design and core algorithms of the proposed SLAM system. Section 4 covers experimental setup and result analysis. Finally, Section 5 concludes the paper and discusses future research directions.

2. Related works

The development of dynamic SLAM systems is a crucial area of research, driven by the challenges posed by dynamic objects in real-world environments [17]. Handling dynamic objects effectively and improving system robustness and stability in such environments have become central issues. This section reviews the primary methods and recent advancements in dynamic SLAM, focusing on techniques that

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

address dynamic object processing and enhance the robustness of SLAM systems in dynamic settings. Dynamic SLAM approaches can currently be classified into three main categories: geometric methods, deep learning-based methods, and hybrid approaches combining both.

2.1. Geometric methods

Geometric methods rely on the assumption that only static feature points comply with the geometric constraints (e.g., epipolar constraints) of SLAM algorithms, distinguishing dynamic objects from static backgrounds. KIM et al. [18] introduced a nonparametric background subtraction model that accounts for the camera's self-motion, computes a background model using depth information, and performs image motion estimation. Dai et al. [19] proposed a method leveraging geometric priors to separate static objects by identifying adjacency between map points and removing irrelevant edges. Wang et al. [20] developed a dynamic target recognition approach combining feature consistency detection and depth image clustering with RGB-D sensors. Xu et al. [21] used feature triangles to detect dynamic objects through changes between keyframes. While computationally efficient in some cases, these methods often depend on predefined thresholds or specific motion models, leading to false positives (static points misclassified as dynamic) or missed detections (dynamic points retained), especially in scenes with complex motions or subtle dynamics. These geometric techniques inspired our use of multi-view geometry as a complementary check in our system.

2.2. Deep learning methods

With rapid advancements in deep learning, numerous dynamic SLAM approaches have adopted these techniques to automatically recognize potentially dynamic objects based on semantic categories, avoiding some of the thresholding issues of purely geometric methods. Li et al. [22] utilized the SegNet semantic segmentation network [23] to distinguish dynamic objects (like people) from static backgrounds. VINCENT et al. [24] employed YOLACT [25] for real-time dynamic object instance segmentation, integrating an extended Kalman filter for tracking and categorizing objects as "moving" or "idle" to exclude them from visual odometry and RTAB-MAP [26] loop closure. However, relying solely on semantics can be problematic: potentially dynamic objects might be temporarily static, and removing all points associated with them can unnecessarily deplete features. Furthermore, deep learning methods require substantial labeled data for training and significant computational resources for inference [27, 28]. Simple networks may lack accuracy, while complex ones (like Mask R-CNN [29]) can compromise real-time performance, a critical requirement for SLAM. Our approach builds on these methods by employing a lightweight YOLOv7 model [14] with optimizations to balance accuracy and efficiency.

2.3. Methods combining geometry and deep learning

Recent studies increasingly integrate geometric and deep learning techniques to leverage their complementary strengths in dynamic SLAM. Yu et al. [30] proposed DS-SLAM, based on ORB-SLAM2 [4], combining semantic segmentation (SegNet) and optical flow to detect moving points within potentially dynamic object masks, aiming to reduce dynamic object interference effectively. Bescos et al.'s DynaSLAM [15] uses Mask R-CNN [29] for instance segmentation and multi-view geometry to check the consistency of points belonging to potentially dynamic objects, excelling in complex traffic scenes but at a higher computational cost. RDS-SLAM [31], built on ORB-SLAM3 [5], enhances recognition accuracy by modeling dynamic objects with movement probabilities and tracking keypoints probabilistically. Wu et al. [12] introduced YOLO-SLAM using YOLOv5 [32] for object detection and novel geometric constraints based on bounding boxes. Huang et al. [33] combined YOLOv5, Kalman filtering, and the Hungarian algorithm with multi-view geometry for improved mapping. More recently, Kan et al. [13] proposed YPR-SLAM using YOLO-Pose-Regression for lightweight human detection

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

combined with geometric constraints, while Zhong et al. [34] introduced DynaTM-SLAM focusing on fast filtering and object-based localization. These hybrid approaches represent the state-of-the-art, but often face trade-offs between the accuracy of dynamic point identification, computational efficiency, and robustness across diverse scenarios. Our system draws from these hybrid approaches by integrating lightweight detection and segmentation with geometric constraints for enhanced performance and efficiency in dynamic environments.

2.4. Limitations of existing dynamic SLAM methods

Although the methods discussed above have advanced dynamic SLAM, they exhibit several limitations that motivate our work:

- **Geometric methods** (e.g., [18], [21]) often rely on predefined thresholds or simple motion models, making them sensitive to parameter tuning and prone to failure in complex scenes with varied object motions, potentially causing false positives or missed detections.
- **Deep learning methods** that use only semantics (e.g., [22]) can incorrectly discard static objects belonging to dynamic classes (like a parked car or a sitting person). Methods relying solely on detection bounding boxes (e.g., [12]) may remove large background areas containing useful static features. High-performance segmentation networks (e.g., Mask R-CNN in [15]) improve precision but often impose significant computational overhead, hindering real-time operation [24].
- **Hybrid approaches** (e.g., [15], [30], [13]) attempt to balance these issues but still face challenges. For instance, DynaSLAM's Mask R-CNN increases computational load. DS-SLAM's optical flow can falter in low-texture regions or with fast motion. Lightweight detection-only methods [13] might lack the precision of segmentation for filtering points near object boundaries.
- **Overall Trade-off:** A persistent challenge across all categories is balancing computational efficiency (real-time performance) with the accuracy and robustness of dynamic object identification and feature rejection, especially in cluttered indoor environments with diverse object types and motions [10]. Many methods enhance accuracy at the cost of speed, or vice versa.

These gaps highlight the need for a dynamic SLAM system that integrates accurate, pixel-level dynamic feature rejection using efficient deep learning models combined with robust geometric verification, which is the focus of our proposed approach.

Despite advancements in dynamic SLAM, challenges remain in achieving real-time performance, optimizing network architectures, and ensuring accurate dynamic object recognition. Geometric methods' dependence on thresholds and limited adaptability further hinder their effectiveness across diverse scenarios. To address these shortcomings, we propose a semantic visual SLAM system tailored for dynamic environments, extending ORB-SLAM2 with a lightweight YOLOv7 model (enhanced with segmentation), optical flow, and multi-view geometry. This approach efficiently rejects dynamic feature points, enhancing both robustness and mapping accuracy, as detailed in the following sections.

3. System description

This section describes the overall framework of the proposed visual SLAM system for dynamic environments. The system design revolves around three key components. First, the system architecture is outlined, consisting of five parallel threads: object detection, dynamic feature point rejection, tracking, local mapping, and loop closure. The object detection thread is then discussed in detail, as it provides essential semantic information to support the other system modules in dynamic settings. Following this, a novel dynamic feature point filtering method is introduced. This method integrates semantic data with

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

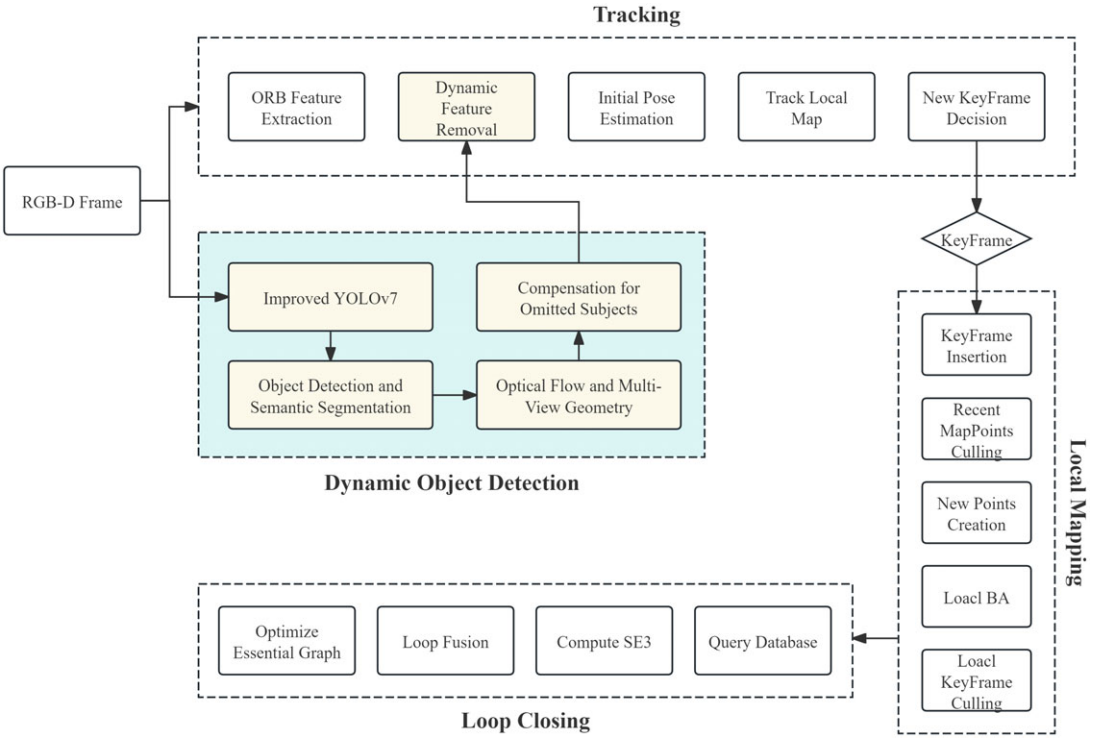


Figure 1. *The overall framework based on ORB-SLAM2 framework, colored parts are additions.*

geometric constraints to more accurately differentiate between dynamic and static feature points, thereby enhancing the system's stability and robustness in dynamic environments.

3.1. Overall system framework

The proposed Dynamic Visual SLAM system builds upon the traditional ORB-SLAM2 framework, which is effective in static environments. ORB-SLAM2 operates with three main parallel threads: tracking, local mapping, and loop closure, ensuring global consistency in both the camera trajectory and the map. Our proposed SLAM system introduces two additional threads: object detection and dynamic feature point rejection. These new threads are designed to address errors caused by dynamic objects, ultimately enhancing the accuracy of position estimation.

In our system, the object detection thread uses the lightweight YOLOv7 model with an added segmentation head to detect dynamic objects, generating bounding boxes and pixel-level masks. These are typically used to identify potentially dynamic objects, such as humans, particularly in indoor environments based on semantic priors. The bounding boxes help broadly locate regions potentially containing dynamic features, while the masks provide precise outlines of the dynamic objects. Next, ORB feature points are extracted from each image frame, including both static and dynamic points. The dynamic feature point rejection thread combines the bounding box and mask information, uses optical flow tracking to handle potential leakage/fast motion, and employs multi-view geometry as a final check to effectively separate dynamic from static feature points within the masks. Only the static feature points are retained, which are then passed to the tracking, local mapping, and loop closure threads. This process aims to preserve more static background points compared to bounding-box-based removal, while remaining consistent with the core ORB-SLAM2 pipeline for map building and optimization using reliable static features. The overall system framework is shown in Figure 1, with the specific dynamic feature point rejection process depicted in Figure 2.

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

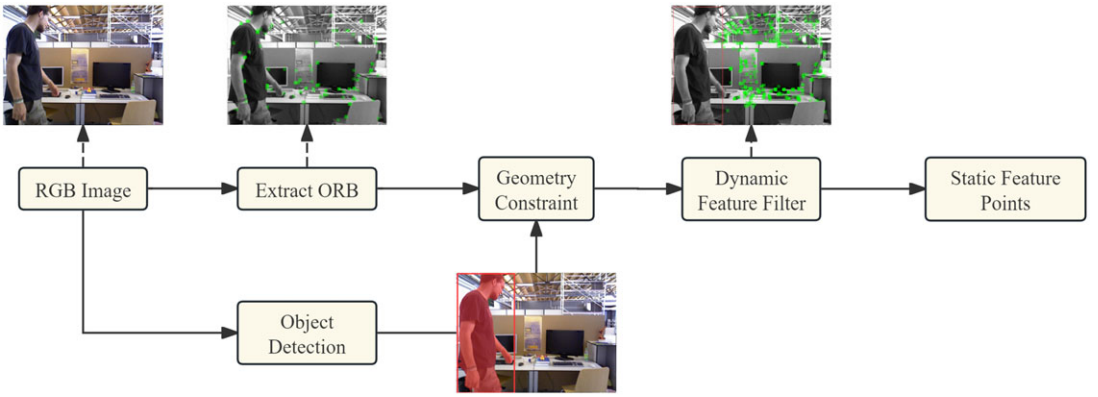


Figure 2. *The process of removing dynamic feature points.*

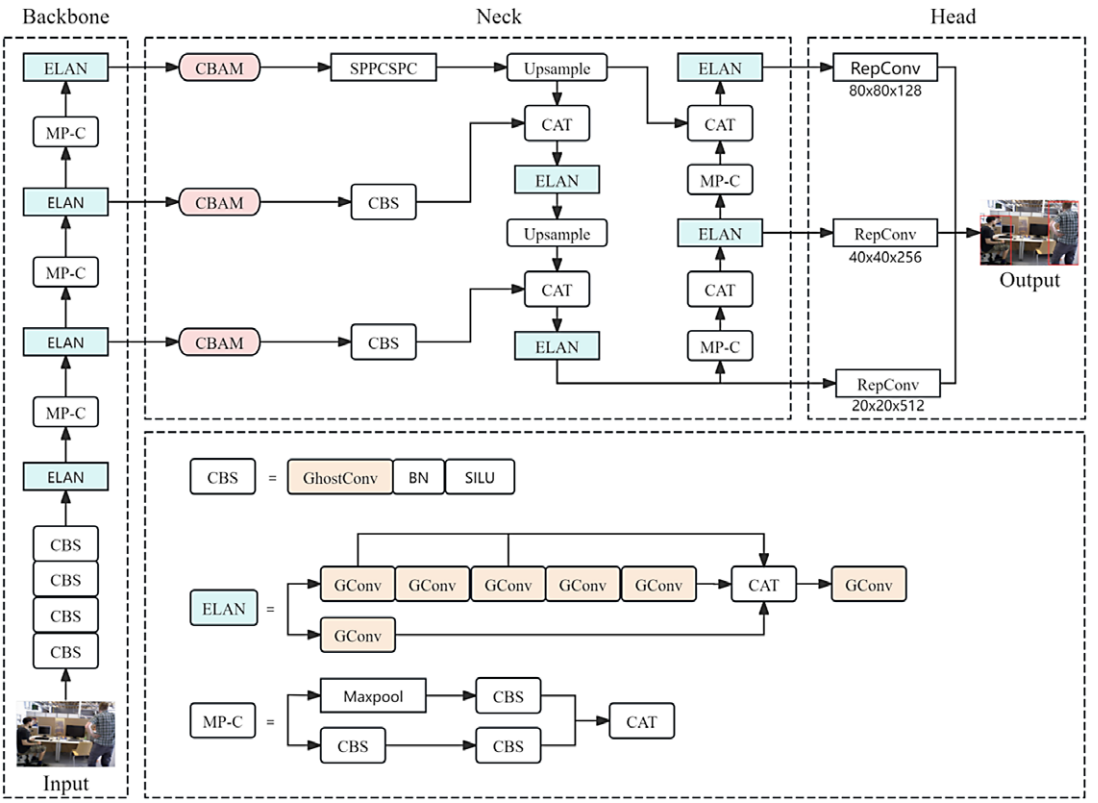


Figure 3. *Improved network framework of YOLOv7, where the colored modules are improved modules.*

3.2. Lightweight target detection network

To meet the real-time demands in resource-limited environments, this paper proposes a lightweight YOLOv7 [14] architecture aimed at removing feature points from dynamic objects (e.g., pedestrians, vehicles, etc.). YOLOv7 was chosen as the base model due to its state-of-the-art performance in balancing detection speed and accuracy at the time of development [35]. While other fast detectors like YOLOv5 [32] or SSD [36] exist, YOLOv7 provided a strong foundation for integrating the required segmentation head and optimizations. The architecture, enhanced with a segmentation head, is shown in Figure 3. The original structure of YOLOv7 consists of three major modules: Backbone, Neck, and Head.

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

To enhance real-time performance, this paper replaces the traditional convolution with Ghost convolution [37] in the backbone network, an existing technique also used in SLAM contexts [36] to reduce computational requirements. The number of parameters and the computational requirements of Ghost convolution are significantly reduced compared with the traditional convolution. Firstly, the comparison of the number of parameters P_{conv} for traditional convolution and P_{Ghost} for Ghost convolution is as follows:

$$P_{conv} = C \times C' \times k \times k \tag{1}$$

$$P_{Ghost} = C \times m \times k \times k + m \times n \times d \times d \tag{2}$$

where C and C' represent the input and output feature map channels, k is the kernel size, m is the number of channels of the initial feature map generated by standard convolution in GhostConv, and n and d refer to additional feature maps and kernel sizes for the cheap linear operations. In Ghost convolution, the computational requirements are reduced by generating fewer intrinsic feature maps (m) and augmenting them with cheaper operations. The ratio R_{cg} of the number of parameters for conventional convolution and Ghost convolution is:

$$R_{cg} = \frac{C \times C' \times k \times k}{C \times m \times k \times k + m \times n \times d \times d} \approx \frac{C'}{m} \tag{3}$$

This shows that when C' is large and m is small (typically $m \approx C'/2$), the number of parameters of Ghost convolution is significantly lower than that of normal convolution, thus reducing the overall number of parameters of the network model. In addition, the computational effort (*FLOPs*) of Ghost convolution is lower than that of the conventional convolution, with the computational effort being respectively:

$$G_{conv} = C \times C' \times k \times k \times H' \times W' \tag{4}$$

$$G_{Ghost} = (C \times m \times k \times k + m \times n \times d \times d) \times H' \times W' \tag{5}$$

where H' and W' are the height and width of the output feature map. The ratio R_{g} of normal convolution and Ghost convolution computation is approximately:

$$R_g = \frac{G_{conv}}{G_{Ghost}} = \frac{C \times C' \times k \times k \times H' \times W'}{(C \times m \times k \times k + m \times n \times d \times d) \times H' \times W'} \approx \frac{C'}{m} \tag{6}$$

This confirms that Ghost convolution reduces both the number of parameters and computational effort, contributing to balancing real-time performance and detection accuracy in our system. To address potential accuracy loss from using lightweight convolutions, we incorporate the existing CBAM attention mechanism [37], a widely used module that enhances feature representation by focusing on important channel and spatial features, as seen in prior computer vision and SLAM works.

CBAM combines channel attention with spatial attention to improve feature detection by the following steps. First, applying the channel attention module to the input feature map F to obtain the channel attention map M_c, computed as:

$$M_c = \sigma(FC(AP(F)) + FC(MP(F))) \tag{7}$$

where AP and MP denote average pooling and max pooling operations, respectively, FC is the fully connected layer, and σ is the Sigmoid activation function. Then, using the spatial attention module to generate the spatial attention map N by global average pooling:

$$N = \sigma(FC(AL(F))) \tag{8}$$

where AL is average pooling along the channel dimension. Finally, multiplying the channel and spatial attention maps element-by-element to obtain the final feature map $F_a = M \times N$.

To assess the impact of these modifications (GhostConv "G" and CBAM "C") on the YOLOv7 model's performance and complexity, we conducted an ablation study. The experiments were performed using the COCO 2017 dataset (118,287 training images, 5,000 validation images) within the PyTorch 1.9.0 framework on an NVIDIA RTX 3090 GPU. We trained the models for 100 epochs with a batch

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

Table I. *Ablation study results on COCO 2017 Subset. G denotes GhostConv replacement, C denotes CBAM integration. Standard training settings were used. P, R, mAP measured at IoU threshold 0.5.*

Model	P/%	R/%	mAP@0.5/%	P_a/M
YOLOv7	94.12	92.15	78.45	40.10
YOLOv7+G	90.45	88.90	70.15	32.58
YOLOv7+C	95.78	94.25	82.10	41.23
YOLOv7+G+C	95.12	93.85	**83.15**	**34.75**

size of 16 and an input image size of 640×640, employing standard hyperparameters typical for object detection tasks. We evaluated Precision (P), Recall (R), mean Average Precision (mAP@0.5), and the number of parameters (P_a). The results, shown in Table I, demonstrate that combining GhostConv and CBAM (YOLOv7+G+C) achieves a good balance, significantly reducing parameters compared to the original YOLOv7 while maintaining high mAP, even slightly improving it over the baseline, indicating the effectiveness of the attention mechanism in compensating for the lightweight convolution.

3.3. Dynamic feature removal

In feature-based SLAM systems, the quality of the feature points is crucial for system stability and performance. To address this, this paper presents a robust feature point selection strategy that integrates semantic a priori with geometric information. The semantic a priori helps differentiate between static and dynamic objects, while the geometric constraints ensure the feature points are evenly distributed in space and possess strong geometric consistency.

In contrast to object detection, semantic segmentation offers precise pixel-level masks, allowing for a clearer distinction between feature points of dynamic and static objects. Object detection typically uses bounding boxes, which, although simpler, may include feature points from both dynamic objects and static ones, like furniture. This rough way of bounding box delineation leads to errors in feature point selection. Some studies such as Liu [38] and Xiao [39] chose to directly remove all feature points within the bounding box, but this approach significantly reduces the number of static feature points, especially when the bounding box occupies a large area, which may lead to the failure of the SLAM system.

To maintain as many static feature points as possible, this paper enhances the lightweight YOLOv7 model by incorporating semantic segmentation. This upgrade allows the model to not only detect dynamic objects but also generate pixel-level masks for them. By removing dynamic pixels while preserving static feature points within the dynamic object's region, the system ensures that sufficient static points are retained for initial map estimation, even if a dynamic object occupies a large portion of the scene. This method greatly improves the robustness and accuracy of the geometrically constrained computations in dynamic SLAM scenarios. Figure 4 illustrates the performance of the YOLOv7 model with the added segmentation feature.

However, since the segmentation accuracy of Ghost-YOLOv7 is found to be insufficient in the experimental process, there is a leakage detection of dynamic objects, which needs to be further improved. The leakage detection compensation strategy designed in this paper is mainly divided into two parts: the Lucas-Kanada optical flow tracking module and the mapping compensation part based on the constant velocity model. The Lucas-Kanada optical flow tracking module, on the one hand, tracks the dynamic features between frames, avoids the dynamic features from participating in the calculation of the basis matrix, and ensures the screening accuracy of the limit constraints. On the other hand, it serves as a supplement to the leakage detection compensation module of the constant velocity model. Complementary. In the following, the leakage compensation module based on the constant velocity model is emphasized.

Inspired by SG-SLAM [40], the constant velocity model between neighboring frames is considered, the segmentation result of the previous frame is used to predict the segmentation result of the current frame, and a mask closure operation is used to recover the leakage-detection dynamic object masks,

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press



Figure 4. *Results of dynamic mask segmentation.*

so as to improve the system stability. The algorithm mainly consists of leakage detection judgment, edge point projection, and mask recovery. Let the current frame index be k, and the previous and upper frames be $k-1$ and $k-2$, respectively. The transformation matrices from the upper frame to the previous frame and from the previous to the current frame are $T_{k-1,k-2}$ and $T_{k,k-1}$, respectively. According to the constant velocity model of ORB-SLAM2 [4], since the time intervals between adjacent frames are short, the pose transformations are approximately equal: $T_{k,k-1} \approx T_{k-1,k-2}$. If the number of dynamic objects detected in the current frame is fewer than those in the previous three frames, the dynamic objects in the current frame are considered missed, and mask edge projection is performed accordingly.

First, the mask edge points of the missed dynamic objects in the previous frame are calculated and projected to the world coordinate system (coordinate origin is O_w) by Eq. (9) to obtain $P_i = [X_i, Y_i, Z_i, 1]^T$.

$$P_i = \left(T_{cw}^k\right)^{-1} \begin{bmatrix} (u - c_u) \times d/f_u \\ (v - c_v) \times d/f_v \\ d \\ 1 \end{bmatrix} \tag{9}$$

where T_{cw}^k denotes the bitmap of the k-th frame image, (u, v) denotes the feature point, (c_u, c_v) is the optical center of the camera, (f_u, f_v) is the focal length of the camera, d denotes the depth of the feature point (u,v), and $P_i \in \mathbb{R}^4$ denotes the chi-square coordinates of the feature point in the world coordinate system.

According to the ORB-SLAM2 [4] constant velocity model, the k-th frame bit pose can be calculated as:

$$T_{cw}^k = T_{k,k-1} T_{cw}^{k-1} = T_{k-1,k-2} T_{cw}^{k-1} \tag{10}$$

With Eq. (10), the edge points of the ignored object in the world coordinate system are projected into the current frame (p_i denotes the pixel coordinates of the edge point projected to the current frame).

$$p_i = \frac{1}{d} K T_{cw}^k P_i \tag{11}$$

where K is the current frame's internal reference matrix, and mask restoration is performed after obtaining the outline of the missed object in the current frame. Mask expansion is first implemented to increase the foreground size in the image and eliminate unreliable feature points. Next, mask filling is performed to ensure the continuity and integrity of the recovered leakage object mask. Finally, mask erosion is performed to correct the possible over-expansion caused by the expansion and to ensure that the mask better fits the contours of the actual object. In the subsequent dynamic feature rejection part, the system will combine the Lucas-Kanada method with the results of the leakage compensation module to reject

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

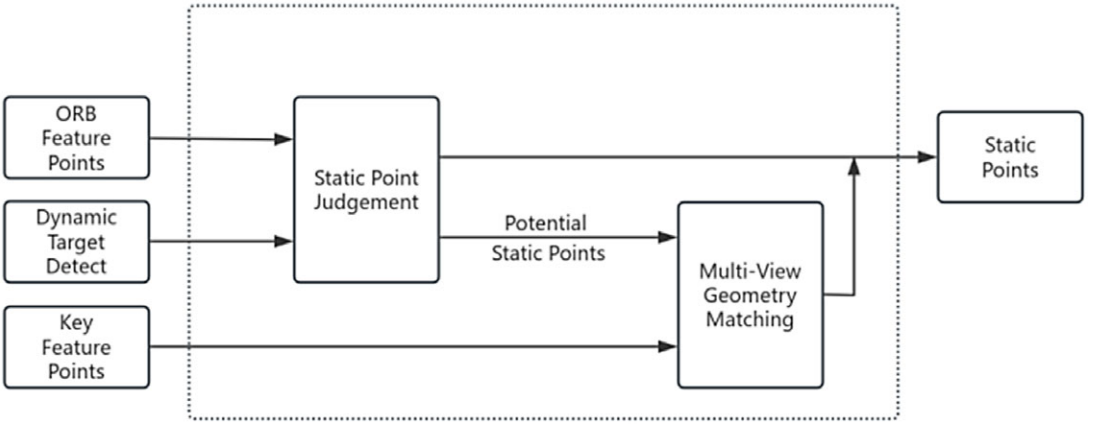


Figure 5. *Multiview geometry preserve static points:This module is executed when the percentage of dynamic target frames in the image exceeds a threshold.*

the feature points belonging to the dynamic mask as needed to ensure the integrity of the leakage mask and the accuracy of the object edges.

When using semantic information to segment dynamic objects in the environment, it is necessary to detect and segment dynamic objects in conjunction with a priori information in advance. Therefore, semantic segmentation detection can only eliminate objects with a priori information that are highly dynamic, and semantic segmentation cannot accurately segment objects with low and potential dynamics. In real-world scenarios, it is often encountered that the motion of dynamic objects with static a priori information in the semantic library has an impact on the SLAM system's mapping and localization. Therefore, we add a multi-view geometry module to the system, as shown in Figure 5.

To enhance the system's efficiency, we avoid applying multi-view geometric constraints to every frame [33]. The ratio T between the image frame area s_1 and the mask area s_2 is defined as $T = s_1/s_2$. If $T < \theta$ (with $\theta = 0.6$), all feature points on the mask are discarded, assuming there are sufficient static feature points for matching. On the other hand, when $T > \theta$, dynamic feature points are filtered using multi-view geometry, ensuring both accuracy and real-time performance.

For filtering dynamic and static feature points, the similarity between the current frame and keyframes is calculated. The three keyframes most similar to the current one are selected based on a distance metric, defined as:

$$v_{Dist} = 0.7 \times v_{Trans} + 0.3 \times v_{Rot} \tag{12}$$

where v_{Trans} is the normalized translation distance between two frames, and v_{Rot} represents the normalized Euler angle rotation difference. The coefficients 0.7 and 0.3 are the weight factors for translation and rotation, respectively. The keyframes are then sorted based on v_{Dist}, and the three closest ones are chosen as the reference system. Next, depth information from the selected reference frames is used to compute the 3D coordinates of feature points, which are then transformed into the world coordinate system. Feature points in the current frame CF are kept if their view direction α is within 30° of the reference frame KF view. If the angle α exceeds 30° (as shown in Figure 6(a)), the point is considered dynamic and discarded. The remaining mapped points are projected back into the current frame's coordinate system. Points with depth values Z' greater than zero and less than the projected depth Z_{proj} are considered potentially dynamic, as they may be occluded by moving objects. The depth difference for these points is then calculated. If the depth difference is smaller than a predefined threshold, the variance of the depth in the region is checked. As shown in Figure 6(b), if the depth variance is below the threshold, the region is considered static; otherwise, the points are classified as dynamic.

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

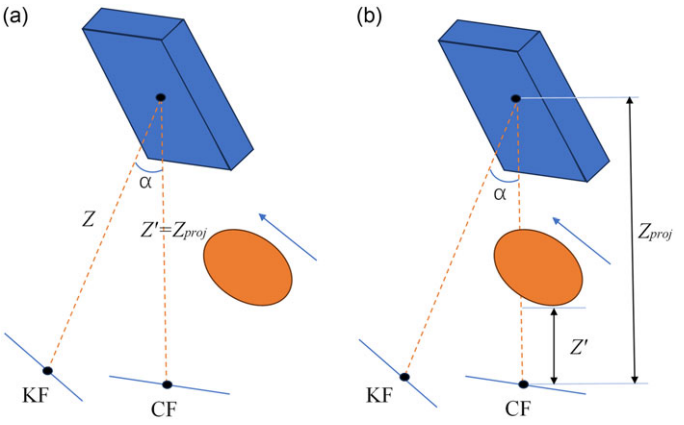


Figure 6. *Multi-view geometric dynamic feature judgment principle.*

4. Experimental

To assess the effectiveness of the proposed dynamic SLAM framework, we conducted experiments using the TUM RGB-D dataset, which includes a variety of indoor scene sequences with corresponding RGB-D images and real camera motion trajectories. The dynamic object sequences in this dataset are categorized into two groups: low-dynamic and high-dynamic scenes. The high-dynamic sequences feature four types of camera motions: (1) *w/half*, where the camera follows a hemispherical path; (2) *w/rpy*, where the camera rotates around the xyz axes; (3) *w/static*, where the camera remains largely stationary with minimal movement; and (4) *w/xyz*, where the camera moves along the xyz axes. In these sequences, dynamic objects occupy a significant portion of the field of view, which complicates position estimation and presents substantial challenges for SLAM systems. The experiments were conducted on a personal computer with an Intel Core i7-8750H processor, 16 GB of RAM, and an NVIDIA GTX 1060 graphics card. These sequences were specifically chosen to evaluate the performance of the SLAM system in environments with varying levels of dynamic activity.

4.1. Experiments in the TUM dataset

The dynamic feature detection method proposed in this paper is built upon the front-end of ORB-SLAM2 [4]. To evaluate its performance, we compare the trajectory and accuracy with ORB-SLAM2 [4], DynaSLAM [15], and DS-SLAM [30], which are established dynamic SLAM approaches. DynaSLAM uses deep learning and multi-view geometry, while DS-SLAM combines semantic segmentation and optical flow for dynamic object handling. Additionally, we reference recent methods like [13] and [34] to broaden the comparison. The accuracy of the estimated trajectories in dynamic settings is assessed using the EVO evaluation tool. Specifically, we use two key metrics: absolute trajectory error (ATE) and relative positional error (RPE) for quantitative analysis. Tables II, III, IV summarize the results, showing the ATE, relative translational error, and relative rotational error, respectively. We utilize root mean square error (RMSE) and standard deviation (SD) to characterize the system's accuracy and stability. Our method demonstrates notable improvements over ORB-SLAM2 [4] in all dynamic sequences, as evidenced by the last two columns of Tables II, III, IV, which highlight the enhancement in performance under varying levels of dynamics.

During the tracking phase, we incorporate a dynamic feature thresholding model into the SLAM front-end to remove dynamic feature points. Experimental results show that ATE significantly outperforms ORB-SLAM2 [4] in four highly dynamic scenarios. The optimized root mean square error (RMSE) reaches more than 96% at peak and at least 89% compared to ORB-SLAM2. Compared with other dynamic SLAM frameworks, our algorithm's ATE outperforms DynaSLAM [15] for *w/half* and *w/xyz* sequences (e.g., 0.0264 m vs. 0.0279 m in *w/half*), and achieves competitive ATE results in *w/rpy*

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

Table II. *Absolute trajectory error (ATE RMSE & SD, in meters) comparison on TUM RGB-D dynamic sequences.*

	ORBSLAM2 [4]		DynaSLAM [15]		DS-SLAM [30]		OurSLAM		Improvement (vs ORBSLAM2)	
Datasets	**RMSE**	**SD**	**RMSE**	**SD**	**RMSE**	**SD**	**RMSE**	**SD**	**RMSE(%)**	**SD(%)**
walking_half	0.7213	0.3719	0.0279	0.0144	0.0303	0.0159	**0.0264**	**0.0123**	96.34	96.69
walking_rpy	0.9094	0.4364	**0.0414**	**0.0201**	0.4442	0.2350	0.0517	0.0268	94.31	93.86
walking_static	0.1062	0.0677	**0.0075**	**0.0033**	0.0081	0.0036	0.0111	0.0055	89.55	91.88
walking_xyz	0.5950	0.2409	0.0364	0.0175	0.0347	0.0161	**0.0301**	**0.0160**	94.94	93.36
sitting_static	0.0089	0.0042	0.0087	0.0044	**0.0065**	**0.0033**	0.0073	0.0037	17.98	11.90

Table III. *Relative pose error (RPE) – translation (RMSE & SD, in meters) comparison.*

	ORBSLAM2 [4]		DynaSLAM [15]		DS-SLAM [30]		OurSLAM		Improvement (vs ORBSLAM2)	
Datasets	**RMSE**	**SD**	**RMSE**	**SD**	**RMSE**	**SD**	**RMSE**	**SD**	**RMSE(%)**	**SD(%)**
walking_half	0.4501	0.3386	0.0302	**0.0146**	**0.0297**	0.0152	0.0308	0.0149	93.16	95.60
walking_rpy	0.3789	0.2518	**0.0531**	**0.0269**	0.1503	0.1168	0.0661	0.0375	82.58	85.11
walking_static	0.0676	0.0570	0.0118	0.0056	0.0112	0.0058	**0.0100**	**0.0049**	85.21	91.40
walking_xyz	0.3613	0.2431	0.0293	0.0148	0.0333	0.0229	**0.0261**	**0.0139**	91.89	93.91
sitting_static	0.0099	0.0048	**0.0084**	0.0043	0.0095	0.0046	0.0086	**0.0040**	13.13	16.67

Table IV. *Relative pose error (RPE) – rotation (RMSE & SD, in degrees) comparison.*

	ORBSLAM2 [4]		DynaSLAM [15]		DS-SLAM [30]		OurSLAM		Improvement (vs ORBSLAM2)	
Datasets	**RMSE**	**SD**	**RMSE**	**SD**	**RMSE**	**SD**	**RMSE**	**SD**	**RMSE(%)**	**SD(%)**
walking_half	9.4547	7.0276	**0.7874**	**0.3541**	0.8142	0.4101	0.8584	0.4025	91.13	94.27
walking_rpy	7.3284	4.8906	**0.9657**	**0.5264**	3.0042	2.3065	1.1329	0.6417	84.54	86.88
walking_static	1.1645	0.9591	0.3206	0.1487	0.2890	0.1382	**0.2633**	**0.1193**	77.39	87.57
walking_xyz	6.8331	4.5150	0.6511	**0.3826**	0.8266	0.5826	**0.6337**	0.3954	90.73	91.24
sitting_static	0.3021	0.1323	0.2754	0.1211	**0.2735**	0.1215	0.2828	**0.1197**	6.38	9.52

and *w/static* sequences, though DynaSLAM performs slightly better (e.g., 0.0414 m vs. 0.0517 m in *w/rpy*). Against DS-SLAM, our method shows superior ATE in high-dynamic sequences (e.g., 0.0159 m vs. 0.0264 m in *w/half*), but DS-SLAM excels in low-dynamic *s/static* (0.0065 m vs. 0.0073 m).

In the low-dynamic environment, the *s/static* sequence is selected for testing, featuring two people sitting at a table with slight body movement, alongside moving pedestrians. This scenario demands precise detection of subtle motion features. Compared with ORB-SLAM2 [4], our algorithm achieves better accuracy (ATE 0.0073 m vs. 0.0089 m), while DynaSLAM's ATE (0.0087 m) is closer to ORB-SLAM2's, and DS-SLAM performs best (0.0065 m). Our method not only performs robustly in highly dynamic scenes but also effectively detects real moving objects in low-dynamic scenes, ensuring sufficient static feature points are extracted. DynaSLAM excels in high-dynamic sequences but struggles with subtle dynamics, whereas our method maintains robustness across both conditions.

To assess computational efficiency, we compared the average processing time per frame of our method (OurSLAM) with DynaSLAM [15] and ORB-SLAM2 on the TUM RGB-D dataset. The results are shown in Table V. OurSLAM reduces processing time by approximately 68% to 84% compared to DynaSLAM across all sequences. For instance, in the *w/half* sequence, OurSLAM achieves 0.3448 s per frame compared to DynaSLAM's 1.1400 s, a reduction of about 69.8%. This efficiency stems from

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

Table V. *Computational time comparison (Average seconds per frame on TUM RGB-D).*

Sequence	OurSLAM	DynaSLAM [15]	ORBSLAM2 [4]
walking_xyz	0.2423	1.1270	0.0354
walking_half	0.3448	1.1400	0.0376
walking_static	0.1818	0.5620	0.0345
walking_rpy	0.3444	1.0954	0.0327
sitting_static	0.1772	0.5501	0.0264

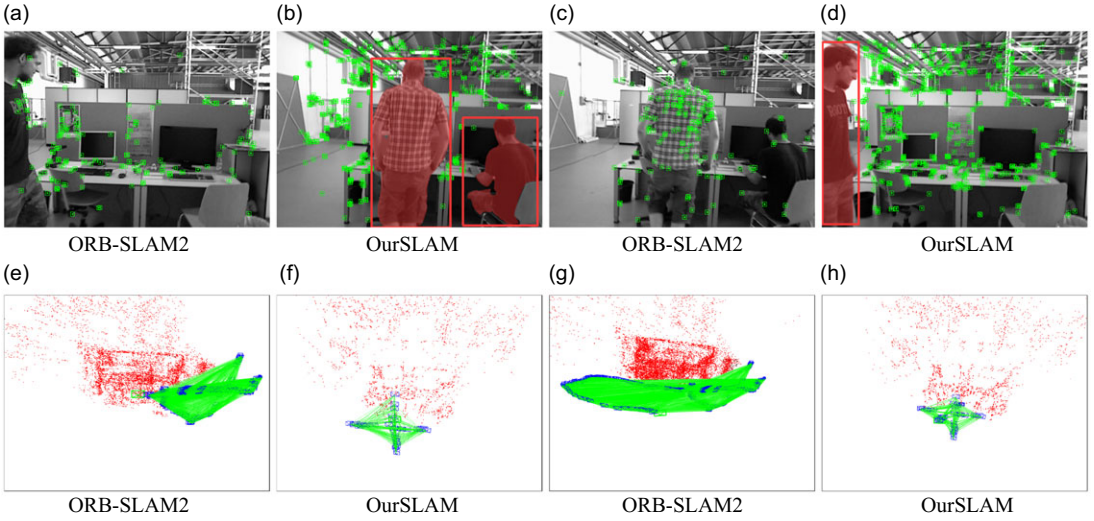


Figure 7. *Dynamic feature point removal and mapping on TUM "walking_xyz." (a,c) ORB-SLAM2 retains features on the person. (b,d) OurSLAM removes features within the person mask. (e,g) ORB-SLAM2 map/trajectory shows significant errors due to dynamic points. (f,h) OurSLAM produces a cleaner map and more accurate trajectory.*

the lightweight YOLOv7 model and optimized geometric constraints, making OurSLAM significantly more suitable for real-time applications than DynaSLAM. However, ORB-SLAM2 remains the fastest, averaging around 0.03 s per frame (e.g., 0.0376 s in *w/half*), though it sacrifices accuracy in dynamic environments.

In highly dynamic sequences such as *w/half* (abbreviated as *w/half*) and *w/xyz* (abbreviated as *w/xyz*), OurSLAM's efficiency contributes to a lower absolute trajectory error (ATE). For example, in *w/half*, OurSLAM achieves an ATE of 0.0264 m compared to DynaSLAM's 0.0279 m, thanks to rapid dynamic feature rejection using YOLOv7, outperforming DynaSLAM's slower Mask R-CNN-based approach. In *w/xyz*, OurSLAM reduces ATE by 17% (0.0301 m vs. 0.0364 m) with faster processing (0.2423 s vs. 1.1270 s per frame), enabling accurate pedestrian removal, as shown in Figure 7. Conversely, in sequences with rotational motion or minimal dynamics, such as *w/rpy* (abbreviated as *w/rpy*) and *w/static* (abbreviated as *w/static*), DynaSLAM exhibits better ATE (e.g., 0.0414 m vs. 0.0517 m in *w/rpy*), likely due to its segmentation sensitivity, though at a higher computational cost. Figure 7 illustrates this with the *w/xyz* dataset: ORB-SLAM2 retains pedestrian features (Figure 7(a), (c)), causing trajectory errors (Figure 7(e), (g)), while OurSLAM removes them (Figure 7(b), (d)), yielding accurate maps and trajectories (Figure 7(f), (h)). Figure 8 further shows reduced ATE and relative positional error (RPE) in *w/xyz* and *w/half* compared to ORB-SLAM2, validating our approach's effectiveness. OurSLAM thus balances robust performance and efficiency, excelling in real-time high-dynamic scenarios.

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

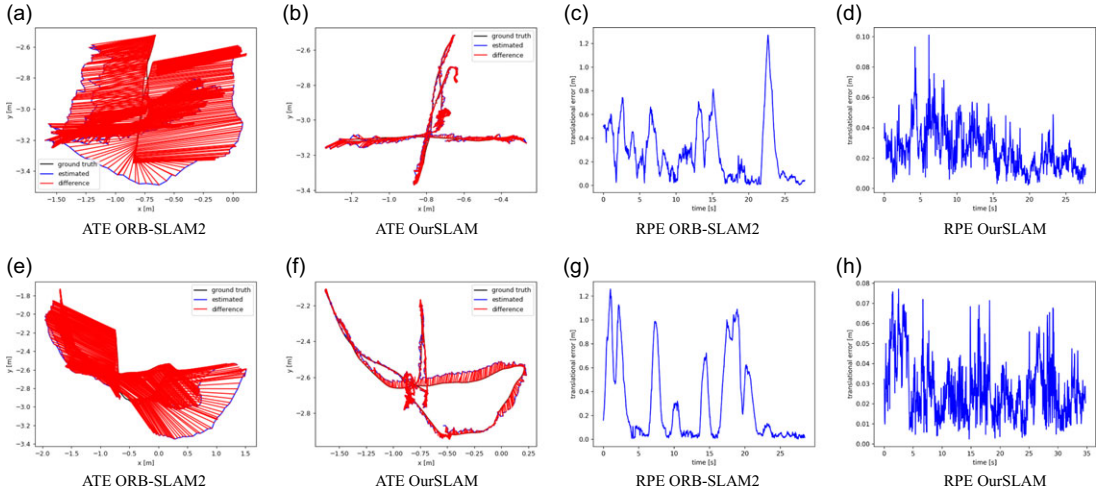


Figure 8. *Trajectory error plots (ATE and RPE trans.) comparing ORB-SLAM2 and OurSLAM on "walking_xyz" (top row) and "walking_half" (bottom row) sequences. Error plots show significantly lower errors for OurSLAM.*

Although the TUM RGB-D dataset primarily features human dynamic objects, our YOLOv7 model, trained on COCO, can detect diverse objects like cars. While direct testing on non-human objects is limited by the dataset, OurSLAM's generalizable rejection mechanism suggests potential effectiveness, unlike DynaSLAM's more specific Mask R-CNN approach. Future work will explore this capability on datasets like KITTI.

4.2. Dense point cloud diagram

In order to validate the map construction effect and accuracy of the proposed SLAM system in this paper, dense point cloud maps under the TUM dataset are constructed in this paper using the proposed dynamic environment SLAM system. In this paper, two scenarios, static and dynamic dataset under TUM dataset, are used to verify the mapping effect of the proposed SLAM system, and the same type of ORB-SLAM2 [4] and DynaSLAM [15] are used to construct the dense point cloud maps under the same dataset, and the results of this paper's SLAM system are verified to meet the requirements by analyzing and comparing the results. By analyzing and comparing the mapping results, it is verified that the proposed SLAM system in this paper meets the requirements.

Figure 9 shows the mapping results on the low-dynamic "sitting_static" sequence. ORB-SLAM2 (Figure 9(a)) fails to remove the dynamic objects (the seated person highlighted in the red box) and integrates them into the map, creating inaccuracies and ghosting artifacts. In contrast, OurSLAM (Figure 9(b)) successfully identifies the person using semantic segmentation and verifies their points are inconsistent or belong to a dynamic category, removing them from the final map while preserving the static background (table, monitor, walls), resulting in a cleaner and more accurate representation of the static environment.

Figure 10 compares mapping results on the highly dynamic "walking_xyz" sequence. DynaSLAM (Figure 10(a)), while attempting to remove the walking person, exhibits noticeable ghosting artifacts (remnants of the dynamic object in the map, highlighted in the red box). This might be due to slight inaccuracies in pose estimation caused by its higher computational requirements (see Table V) potentially leading to delays in processing frames or minor inconsistencies in the segmentation and inpainting process over time. OurSLAM (Figure 10(b)), benefiting from its efficient dynamic feature rejection, completely removes the highly dynamic person, leaving almost no ghosting artifacts and providing

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

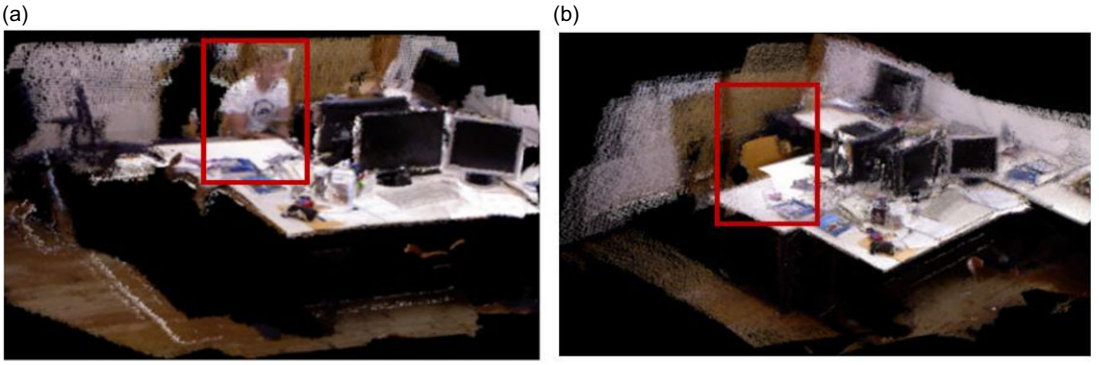


Figure 9. *Comparison of dense map building results in a low dynamic environment ("sitting_static").*

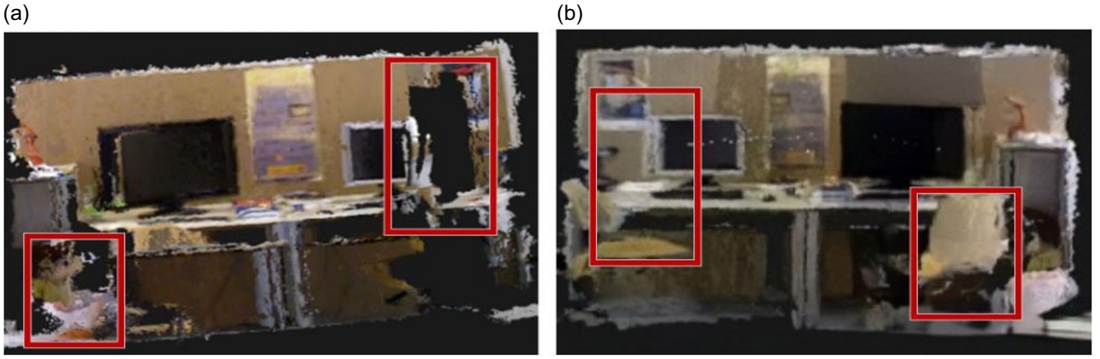


Figure 10. *Comparison of dense map building results in a highly dynamic environment ("walking_xyz").*

a clear map of the static background structure. This demonstrates the effectiveness of our integrated approach in handling highly dynamic scenes for accurate mapping.

4.3. Experimentation in real environments

To qualitatively assess the proposed SLAM algorithm's performance in a real-world indoor setting beyond benchmark datasets, we conducted two experiments using an Intel RealSense D435i RGB-D camera connected to the laptop specified earlier. These experiments aim to visually demonstrate the system's capability in handling dynamic objects, including those with static semantic priors that start moving. Quantitative evaluation with ground truth is challenging in unstructured real-time scenarios, so this section focuses on qualitative observations.

First, the camera was held moving through a lab environment while a person walked back and forth across the field of view, representing a typical high-dynamic scenario. Second, we tested the system's ability to handle objects with static priors that become dynamic. In this test, the camera moved while a person initially stood still holding a water cup (both potentially detectable) and then the person walked across the scene carrying the cup.

As shown in Figure 11(b), during the first experiment, the proposed SLAM algorithm successfully tracked static features in the background environment while effectively ignoring or rejecting features associated with the walking person. The visualization shows a stable set of features tracked on the static parts of the scene, demonstrating the system's ability to filter out the high-dynamic human motion in real time.

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

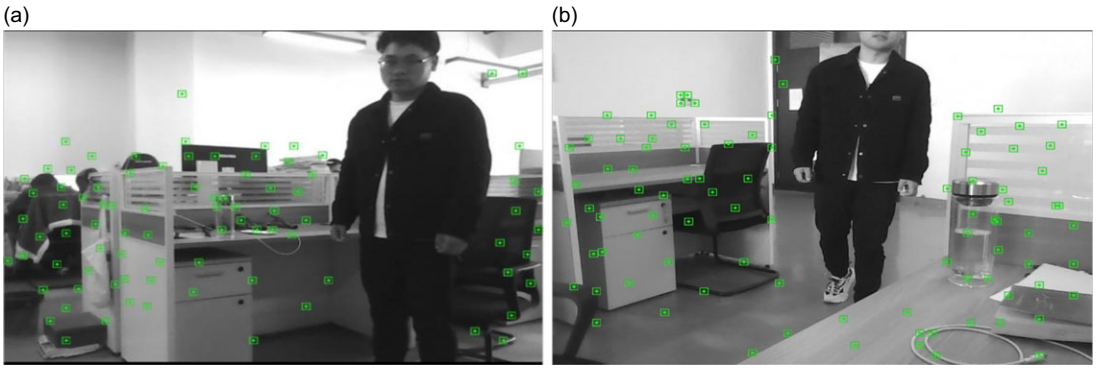


Figure 11. *Dynamic feature point rejection in a real environment. (a) View from the camera with a person walking. (b) OurSLAM running, showing tracked features (green dots). Features on the walking person (within the detected mask, not shown explicitly) are correctly excluded from tracking.*

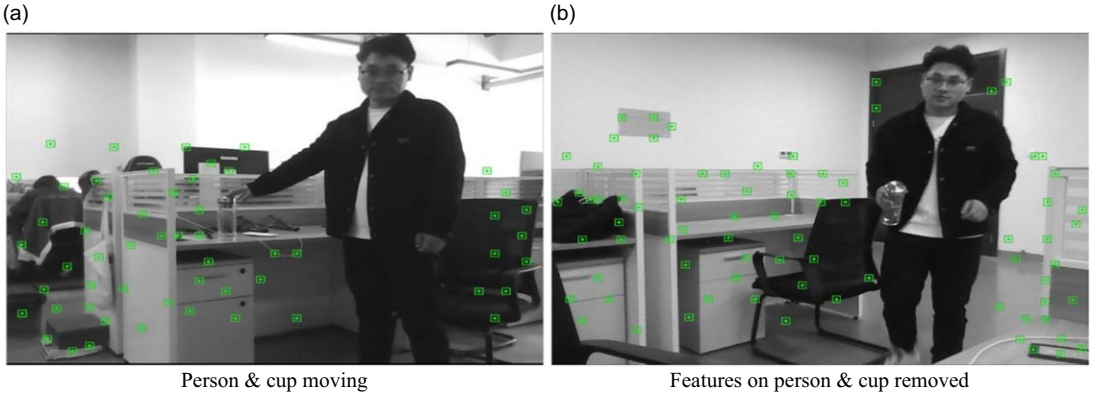


Figure 12. *Handling dynamic objects with static priors. (a) Scene where a person walks while holding a cup. (b) OurSLAM correctly identifies both the person and the cup as dynamic (due to association with the moving person and geometric inconsistency) and excludes their features from tracking.*

In the second experiment (Figure 12), when the person holding the cup was stationary, features on both might initially be tracked (similar to Figure 11(b) but including the cup). However, once the person started walking, the system detected the motion inconsistency of features on both the person and the cup they were carrying. Figure 12(b) illustrates that features on the moving cup were successfully eliminated along with features on the person. This indicates that the system does not rely solely on semantic priors ("cup" is usually static) but uses the geometric and motion consistency checks (optical flow, multi-view geometry) to correctly identify and reject features on objects that become dynamic through interaction.

These qualitative results suggest that the proposed SLAM system functions effectively in real indoor environments, robustly handling both common dynamic objects like people and objects that unexpectedly become dynamic, meeting the practical requirements for dynamic feature removal.

5. Summary and outlook

In this work, we presented a visual SLAM system designed to improve robustness and accuracy in dynamic environments. The core contribution is an effective dynamic feature rejection strategy that integrates a lightweight YOLOv7 network (enhanced with a segmentation head and optimized with

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

Ghost convolution and CBAM) with optical flow tracking and multi-view geometric constraints. This combination allows for efficient, pixel-level identification and removal of dynamic feature points while maximizing the retention of useful static features, even those located near or within the bounding regions of dynamic objects. Our experiments on the TUM RGB-D dataset demonstrate that the proposed system (OurSLAM) significantly outperforms the baseline ORB-SLAM2 and achieves competitive, sometimes superior, accuracy compared to state-of-the-art dynamic SLAM methods like DynaSLAM and DS-SLAM, particularly in highly dynamic scenes. Crucially, OurSLAM achieves this performance with substantially lower computational requirements compared to methods using heavier segmentation networks like Mask R-CNN, making it more suitable for real-time robotics applications. Qualitative tests in real-world scenarios further validated its effectiveness in handling various dynamic situations.

However, while the method demonstrates significant improvements, limitations remain. Performance can still degrade in scenarios with extreme dynamics, severe occlusions, or very low-texture environments where feature tracking (both ORB and optical flow) becomes unreliable. The accuracy is also dependent on the performance of the underlying YOLOv7 model and the quality of the segmentation masks. Future work will focus on several areas: (1) Refining the dynamic feature rejection logic, perhaps by incorporating more sophisticated motion models or probabilistic filtering. (2) Enhancing robustness to challenging conditions like poor illumination or texture. (3) Conducting extensive evaluations on more diverse datasets (e.g., KITTI [41]) featuring a wider range of dynamic objects like vehicles. (4) Exploring further optimizations for deployment on computationally constrained robotic platforms. (5) Releasing the source code to facilitate further research and community adoption.

Author contributions. All authors contributed to the study's conception and design. Yewei Shen conducted data gathering, implementation, experiments, and wrote the article. Xinguang Zhang supervised the project and reviewed the manuscript.

Financial support. This research received no specific grant from any funding agency, commercial, or not-for-profit sectors.

Competing interests. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable. This study used publicly available datasets (TUM RGB-D) and conducted experiments in a lab environment with consenting participants.

References

[1] G. Bresson, Z. Alsayed, L. Yu and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Trans. Intell. Veh.* **2**(3), 194–220 (2017).

[2] Z. Chen, W. Sheng, G. Yang, Z. Su and B. Liang, "Comparison and Analysis of Feature Method and Direct Method in Visual Slam Technology for Social Robots," **In:** *2018 13th World Congress on Intelligent Control and Automation (WCICA) 2018 Jul 4*, IEEE (2018) pp. 413–417.

[3] R. Mur-Artal, J. M. Montiel and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.* **31**(5), 1147–1163 (2015).

[4] 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).

[5] C. Campos, R. Elvira, J. J. Rodríguez, J. 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).

[6] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," **In:** *2011 International conference on computer vision 2011 Nov 6*, Ieee (2011) pp.2564–2571.

[7] J. Engel, T. Schöps and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," **In:** *European conference on computer vision 2014 Sep 6*, Cham, Springer International Publishing (2014) pp. 834–849.

[8] C. Forster, M. Pizzoli and D. Scaramuzza, "SVO: Fast Semi-direct Monocular Visual Odometry," **In:** *2014 IEEE international conference on robotics and automation (ICRA) 2014 May 31*, IEEE (2014) pp. 15–22.

[9] R. A. Newcombe, S. J. Lovegrove and A. J. Davison, "DTAM: Dense Tracking and Mappping In Real-Time," **In:** *2011 international conference on computer vision 2011 Nov 6*, IEEE (2011) pp. 2320–2327.

[10] A. Macario Barros, M. Michel, Y. Moline, G. Corre and F. Carrel, "A comprehensive survey of visual slam algorithms," *Robotics* **11**(1), 24 (2022).

[11] M. R. Saputra, A. Markham and N. Trigoni, "Visual SLAM and structure from motion in dynamic environments: A survey," *ACM Computing Surveys (CSUR)* **51**(2), 1–36 (2018).

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

[12] 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**(8), 6011–6026 (2022).
[13] X. Kan, G. Shi, X. Yang and X. Hu, "YPR-SLAM: A SLAM system combining object detection and geometric constraints for dynamic scenes," *Sensors (Basel, Switzerland)* **24**(20), 6576 (2024). doi: 10.3390/s24206576.
[14] C. Y. Wang, A. Bochkovskiy and H. Y. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-art For Real-time Object Detectors," **In:** *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition 2023*, IEEE (2023) pp. 7464–7475.
[15] B. Bescos, J. M. Fácil, J. Civera and N. J. DynaSLAM, "Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.* **3**(4), 4076–4083 (2018).
[16] P. Sturm, "Multi-View Geometry for General Camera Models," In. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) 2005 Jun 20*, IEEE (2005) vol. 1, pp. 206–212.
[17] B. Al-Tawil, T. Hempel, A. Abdelrahman and A. Al-Hamadi, "A review of visual SLAM for robotics: Evolution, properties, and future applications," *Front. Robot. AI* **11**, 1347985 (2024). doi: 10.3389/frobt.2024.1347985.
[18] D. H. Kim and J. H. Kim, "Effective background model-based RGB-D dense visual odometry in a dynamic environment," *IEEE Trans. Robot.* **32**(6), 1565–1573 (2016).
[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. Mach. Intell.* **44**(1), 373–389 (2020).
[20] 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.-BASEL* **11**(10), 1143 (2019).
[21] G. Xu, Z. Yu, G. Xing, X. Zhang and F. Pan, "Visual odometry algorithm based on geometric prior for dynamic environments," *Int. J. Adv. Manuf. Tech.* **122**(1), 235–242 (2022).
[22] F. Li, W. Chen, W. Xu, L. Huang, D. Li, S. Cai, M. Yang, X. Xiong, Y. Liu and W. Li, "A mobile robot visual SLAM system with enhanced semantics segmentation," *IEEE Access* **29**, 25442–25458 (2020).
[23] V. Badrinarayanan, A. Kendall and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(12), 2481–2495 (2017).
[24] J. Vincent, M. Labbé, J. S. Lauzon, F. Grondin, P. M. Comtois-Rivet and F. Michaud, "Dynamic Object Tracking and Masking for Visual SLAM," **In:** *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2020 Oct 24*, IEEE (2020) pp. 4974–4979.
[25] D. Bolya, C. Zhou, F. Xiao and Y. J. Lee, "Yolact: Real-Time Instance Segmentation," **In:** *Proceedings of the IEEE/CVF international conference on computer vision 2019,* IEEE (2019) pp. 9157–9166.
[26] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *J. Field. Robot.* **36**(2), 416–446 (2019).
[27] L. Guo, Y. Lei, N. Li, T. Yan and N. Li, "Machinery health indicator construction based on convolutional neural networks considering trend burr," *Neurocomputing* **31**, 142–150 (2018).
[28] Y. Tan, L. Guo, H. Gao and L. Zhang, "Deep coupled joint distribution adaptation network: A method for intelligent fault diagnosis between artificial and real damages," *IEEE T. Instrum. Meas.* **9**, 1–2 (2020).
[29] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask r-cnn," **In:** *Proceedings of the IEEE international conference on computer vision 2017,* IEEE (2017) pp. 2961–2969.
[30] 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:** *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS) 2018 Oct 1*, IEEE (2018) pp. 1168–1174.
[31] Y. Liu and J. Miura, "RDS-SLAM: Real-time dynamic SLAM using semantic segmentation methods," *IEEE Access* **9**, 23772–23785 (2021).
[32] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, Y. Kwon, K. Michael, C. Liu, V. Abhiram and T. Skalski, "ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support," [Software], Zenodo, Oct 12, 2021. doi: 10.5281/zenodo.5563715.
[33] J. Huang, Q. Xia and L. Kang, "TKG-SLAM: Dynamic SLAM Based on Target Tracking And Multi-view Geometry," **In:** *2024 3rd International Conference on Innovations and Development of Information Technologies and Robotics (IDITR) 2024 May 23*, IEEE (2024) pp. 105–111.
[34] M. Zhong, C. Hong, Z. Jia, C. Wang and Z. Wang, "DynaTM-SLAM: Fast filtering of dynamic feature points and object-based localization in dynamic indoor environments," *Robot. Auton. Syst.* **174**, 104634 (2024). doi: 10.1016/j.robot.2024.104634.
[35] O. E. Olorunshola, M. E. Irhebhude and A. E. Evwiekpaefe, "A comparative study of YOLOv5 and YOLOv7 object detection algorithms," *J. Comput. Soc. Inform.* **2**(1), 1–2 (2023).
[36] I. Winata and J. Oh, "Lightweight extraction and segmentation with ghost convolutional and attention module integration for visual SLAM," *Int. J. Cont., Autom. Syst.* **22**(12), 3595–3605 (2024).
[37] S. Woo, J. Park, J. Y. Lee and I. S. Kweon, "Cbam: Convolutional Block Attention Module," **In:** *Proceedings of the European conference on computer vision (ECCV) 2018*, Cham, Springer International Publishing (2018) pp. 3–19.
[38] G. Liu, W. Zeng, B. Feng and F. Xu, "DMS-SLAM: A general visual SLAM system for dynamic scenes with multiple sensors," *Sensors* **19**(17), 3714 (2019).
[39] L. Xiao, J. Wang, X. Qiu, Z. Rong and Z. X. Dynamic-SLAM, "Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robot. Auton. Syst.* **117**, 1–16 (2019).
[40] 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**, 1–2 (2022).

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press

[41] A. Geiger, P. Lenz and R. Urtasun, "Are We Ready For Autonomous Driving? The Kitti Vision Benchmark Suite," **In:** *2012 IEEE conference on computer vision and pattern recognition 2012 Jun 16*, IEEE (2012) pp. 3354–3361.
[42] J. Sturm, N. Engelhard, F. Kerl, D. Cremers and W. Burgard, "A Benchmark for the Evaluation of RGB-D SLAM Systems," **In:** *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems 2012 Oct 7*, IEEE (2012) pp. 573–580.
[43] Grupp M, "evo: Python package for the evaluation of odometry and slam," [Software], GitHub, 2017. Available at: https://github.com/MichaelGrupp/evo.

Cite this article: Y. Shen and X. Zhang (2025). "A dynamic SLAM system with YOLOv7 segmentation and geometric constraints for indoor environments", Robotica **43**, 2527–2545. https://doi.org/10.1017/S0263574725101823

https://doi.org/10.1017/S0263574725101823 Published online by Cambridge University Press