Pattern Recognition 158 (2025) 111054

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

SLAM^2: Simultaneous Localization and Multimode Mapping for indoor dynamic environments☆

Zhihao Lin [a], Qi Zhang [b], Zhen Tian [a], Peizhuo Yu [a], Ziyang Ye [c], Hanyang Zhuang [d], Jianglin Lan [a,*]

[a] James Watt School of Engineering, University of Glasgow, Glasgow, G12 8QQ, United Kingdom
[b] Faculty of Science, University of Amsterdam, Amsterdam, 1098 XH, Netherlands
[c] School of Computer and Mathematical Sciences, The University of Adelaide, Adelaide, 5005, Australia
[d] The University of Michigan–Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, Shanghai, 200240, China

ARTICLE INFO

Dataset link: https://github.com/SEAL-UofG/SLAM2

Keywords:
Deep learning
Dynamic environment
Pose estimation
Semantic mapping
SLAM

ABSTRACT

Traditional visual Simultaneous Localization and Mapping (SLAM) methods based on point features are often limited by strong static assumptions and texture information, resulting in inaccurate camera pose estimation and object localization. To address these challenges, we present SLAM^2, a novel semantic RGB-D SLAM system that can obtain accurate estimation of the camera pose and the 6DOF pose of other objects, resulting in complete and clean static 3D model mapping in dynamic environments. Our system makes full use of the point, line, and plane features in space to enhance the camera pose estimation accuracy. It combines the traditional geometric method with a deep learning method to detect both known and unknown dynamic objects in the scene. Moreover, our system is designed with a three-mode mapping method, including dense, semi-dense, and sparse, where the mode can be selected according to the needs of different tasks. This makes our visual SLAM system applicable to diverse application areas. Evaluation in the TUM RGB-D and Bonn RGB-D datasets demonstrates that our SLAM system achieves the most advanced localization accuracy and the cleanest static 3D mapping of the scene in dynamic environments, compared to state-of-the-art methods. Specifically, our system achieves a root mean square error (RMSE) of 0.018 m in the highly dynamic TUM w/half sequence, outperforming ORB-SLAM3 (0.231 m) and DRG-SLAM (0.025 m). In the Bonn dataset, our system demonstrates superior performance in 14 out of 18 sequences, with an average RMSE reduction of 27.3% compared to the next best method.

1. Introduction

Simultaneous Localization and Mapping (SLAM) addresses the challenge of an agent determining its position while constructing a map of an unknown environment. Visual SLAM (vSLAM) systems, utilizing RGB-D cameras, have gained popularity due to their low cost, small size, and lightweight nature compared to other sensors like lidar. Due to this ascendant, a large volume of studies have been dedicated to visual SLAM (vSLAM) systems.

However, most traditional vSLAM systems are based on the assumption of static scenes, which is obviously not consistent with real-world scenarios. When dynamic objects appear in the scene, the vSLAM systems cannot recover the camera pose accurately. With the advancement of the SLAM technique, they are being increasingly used in a wide range of applications with complex dynamic environments, such as virtual reality (VR) [1], autonomous driving, and robotics [2]. This necessitates the understanding of dynamic objects in the environment.

Traditional vSLAM algorithms rely on feature point matching and epipolar constraints for camera pose estimation. Dynamic objects disrupt this process by affecting feature matching, thereby compromising pose recovery. The traditional geometric methods in the vSLAM systems, such as object detection-based methods, cannot completely segment the boundaries of moving objects. Deep learning-based semantic segmentation methods have been proposed to solve this problem. However, semantic segmentation can only process about 20 classes in the scene and cannot balance high precision and high real-time performance. In comparison, object detection-based methods can handle 80 classes in the scene but may cause insufficient data association, resulting in tracking failure [3]. This motivates us to integrate the traditional

☆ This work was supported in part by the China Scholarship Council Ph.D. Scholarship for 2023-2027 (No. 202206170011), in part by the Leverhulme Trust Early Career Fellowship (ECF-2021-517), and in part by the UK Royal Society International Exchanges Cost Share Programme (IEC \ NSFC \ 223228).

* Corresponding author.
E-mail address: jianglin.lan@glasgow.ac.uk (J. Lan).

https://doi.org/10.1016/j.patcog.2024.111054
Received 22 November 2023; Received in revised form 7 July 2024; Accepted 25 September 2024
Available online 28 September 2024

geometric methods with deep learning methods are integrated in the localization module of the vSLAM system to enhance the its ability in dynamic environments. The integration of these two methods enable the vSLAM system to detect both known and unknown dynamic objects. The detected dynamic points can be treated as outlier points to remove. This kind of processing eliminates the interference of dynamic noise on the recovery of camera pose and improves robustness of the vSLAM system in dynamic environments.

Most traditional SLAM algorithms only use the feature points to reconstruct sparse map point clouds. The map created in this way has very poor understandability and is not conducive to robots performing complex tasks. The dense point cloud map created by the traditional RGB-D visual SLAM systems is susceptible to interference from dynamic objects and limited to the RGB-D camera model. None of the above methods take advantage of the rich geometric structural information in real-world environments. We propose a novel multimode mapping method that fuses geometric structural and texture information to improve vSLAM localization performance and generate structural maps valuable for applications such as AR scene rendering.

On the other hand, dense point cloud maps that are non-structural can describe the actual texture of the real environment to meet the needs of robot navigation and autopilot [4]. The semantic information in the scene can be exploit to improve the 3D mapping of the environment. This can enhance the SLAM system for downstream tasks such as path planning, robot grasping, and obstacle avoidance [5]. The above motives us to propose a multimode mapping method, where different modes can be selected according to the needs of each task. We can make full use of the point, line, and surface information in space. By removing the interference of dynamic objects, we can construct a sparse point-line-plane 3D map that is more understandable and reusable. We can apply the dynamic object elimination and loop correction to the traditional RGB-D point cloud map to create a more accurate, stable, robust and dense 3D map.

With the development of autonomous robots, higher requirements have been placed on their SLAM systems to understand the surrounding objects to enhance robotic performance. Estimating the poses of other objects in the environment helps achieve complex robotic tasks. However, traditional SLAM algorithms only restore the camera pose without considering the poses of surrounding objects. This motivates us to estimate the 6DOF poses of the objects surrounding the camera to obtain a clearer semantic 3D map of the environment. This can widen the application of the vSLAM system to complex tasks, such as robot crawling and searching.

Considering the above, this paper presents SLAM^2, a novel semantic RGB-D vSLAM system, to solve the various problems in traditional SLAM systems. To reduce the sensitivity of point-based SLAM systems to texture information, we integrate the point features with the segmented instance plane features from PlaneRecNet [6] and the line features from the Line Segment Detector [7]. These features are passed into the state-of-the-art SLAM system ORB-SLAM3 to enhance the pose estimation.

To mitigate the effects of dynamic objects, we employ a two-step process: First, we use CUDA-accelerated YOLOv5 [8] to detect objects from 80 classes. Second, we apply LK optical flow with epipolar constraints to identify outlier points within detected boxes and determine object dynamics based on the percentage of outliers. We keep the static box points within the dynamic box and exclude the other points. We also exclude the features near the detected outliers in the background to deal with the unknown dynamic objects.

Our mapping module offers three modes: dense point cloud mapping, semi-dense point-line-plane 3D reconstruction, and sparse point-line-plane 3D reconstruction. These modes provide simultaneous representation of realistic textures and structural geometry. We use an object detection-based 3D semantic information rendering approach to build 3D semantic models of static objects and estimate their positions for optimal relocation. In dense point cloud mapping, we use the detected edge combined with a continuous-time filter to remove the noise blocks left by the moving objects. In point-line-plane 3D reconstruction, we describe the scene with 3D points, lines, planes, and estimated 3D objects. A brief overview of our proposed research is provided in Fig. 1. The three figures on the left-hand side of Fig. 1 show the feature extraction process, where Fig. 1(a) is the original RGB image and Fig. 1(b) is the segmented plane obtained through PlaneRecNet. Fig. 1(c) shows the extracted points, lines, and detected boxes, where the pink dots and lines indicate those moving points while the green dots and lines indicate those static ones. Our approach preserves more static keypoints, compared to a direct elimination of the entire bounding box. It can be seen that most dynamic points are on the moving human body, while the others in the human bounding box are judged as static. The three figures on the right-hand side of Fig. 1 show our different mapping modes, all of which reconstruct complete and clean static 3D models in the dynamic environment.

The contributions of this paper are summarized as follows:

- We present SLAM^2, a semantic RGB-D vSLAM system based on the point, line, and plane features in the scene. The system has advanced camera pose estimation accuracy and can generate static semantic 3D point cloud maps, and 3D point-line-plane maps for indoor dynamic environments.
- We propose a method for semantic 3D point-line-plane reconstruction by using an object detection technique to estimate the precise location of objects and create 3D models to render mapping with semantic information.
- We develop a multimode mapping method, including dense, semi-dense, and sparse, where individual modes can be selected to meet the needs of different application scenarios.

2. Related works

2.1. vSLAM based on point, line, and plane features

Feature-based vSLAM systems like ORB-SLAM have high requirements for texture richness in the environment. Lines and planes as geometric primitives can efficiently extract the structural information in indoor scenes and they can also be used in SLAM systems along with points. He et al. [9] combine the pre-integrated IMU error with the re-projection errors of points and lines to build a monocular visual-inertial odometry system PL-VIO, by using the Plücker coordinates and orthogonal representations for lines.

Planes from depth images can resist some measurement noise. Zhang et al. [10] generate vertical planes from edge lines using an RGB-D camera and use plane structure constraints for factor graph optimization over points and planes. Arndt et al. [11] enforce the points to lie exactly on their corresponding planes, forming a joint optimization of the planes, points in planes, and regular 3D points.

Some researchers estimate plane normals from RGB images using CNNs to avoid depth image dependence and get better estimates.

Li et al. [12] use a CNN to predict plane normals per input RGB image in real-time. Then they compute the translation with points and lines and the rotation with lines and planes to reduce drifts in the monocular SLAM. Yunus et al. Yang et al. [13] propose Pop-up SLAM, which uses a CNN to segment the ground plane and process features of planes and points in the SLAM system.

2.2. vSLAM in dynamic scenes

Geometry-based methods can improve the performance of vSLAM in dynamic environments. Sun et al. [14] use a re-projection error map of dense optical flow to obtain foreground models that are moving and perform online motion removal. Cheng et al. [15] use sparse LK (Lucas–Kanade) optical flow and fundamental matrices to check if the key points in the scene are dynamic. Du et al. [16] innovatively use

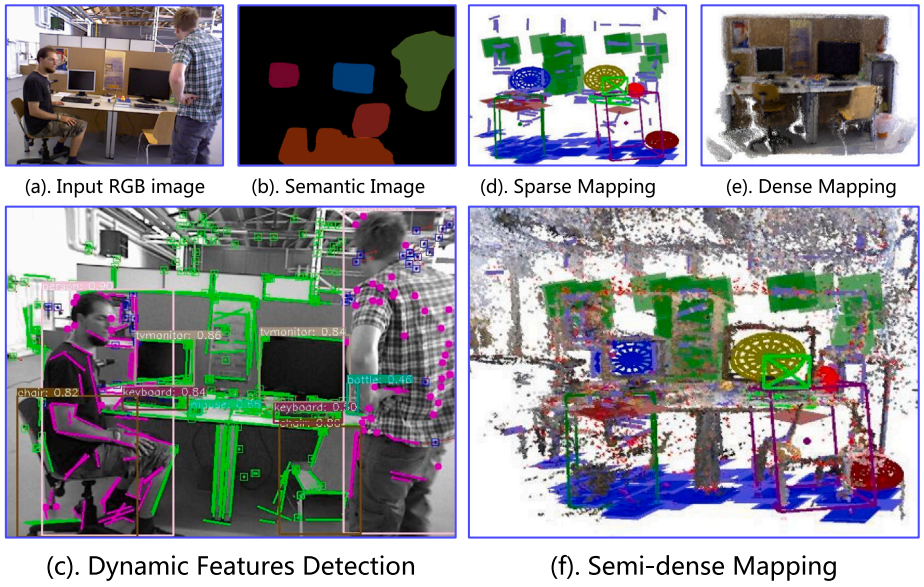

Fig. 1. Overview of our SLAM^2 system. (a) Input RGB image. (b) Semantic segmentation plane image. (c) Feature extraction and dynamic object detection: pink dots and lines indicate dynamic features, green dots and lines indicate static features. (d–f) Our mapping module's three reconstruction modes: (d) sparse, (e) dense, and (f) semi-dense. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

conditional random fields based on historical observations to identify dynamic keypoints.

In contrast, deep learning-based semantic segmentation and object detection algorithms can provide complete, but slightly noisy, object boundaries through bounding boxes and masks. Yu et al. [17] propose DS-SLAM using lightweight SegNet [18] to segment people in scenes and combine sparse LK optical flow with epipolar constraints to determine if people are in motion. Fan et al. [19] refine object masks from depth images that are segmented by BlitzNet [20] and classify objects into three categories based on semantic prior motion information for processing. They also construct epipolar constraints from matched points in the background regions for static point identification.

Hu et al. [21] use multi-view geometry, instead of epipolar constraints, along with Deeplabv3+ masks to discard dynamic points. Liu et al. [22] update the moving probabilities using results from the Bayesian filters and semantic segmentation, by regarding the points above a probability threshold as dynamic. Due to using semantic segmentation, these methods can only process objects from 20 classes or predefined as potentially dynamic. Wang et al. [23] proposed a method using both handcrafted and learned features for point cloud reconstruction. However, their approach requires preprocessing to label static and dynamic parts, which is impractical for real-world applications and challenging for large datasets. In contrast, our method, utilizing an object detection module and optical flow consistency check, can map in various modes without such preprocessing.

Some researchers have introduced lines and planes into SLAM for dynamic environments. Yuan et al. [24] obtain object masks using Mask R-CNN [25], then use the predefined non-priority dynamic object point-line feature matches to compute the fundamental matrices, and finally update the motion probabilities with Bayes' theory to remove the outliers. Wang et al. [26] extend DS-SLAM [17] to dynamic points, lines, and planes. They also use the multi-view geometry to handle the undetected moving objects in semantic segmentation.

Table 1 summarizes the key approaches, datasets, and limitations of recent works in dynamic SLAM. This comparison highlights the evolution of techniques from purely geometry-based methods to deep learning-enhanced approaches, as well as the persistent challenges in handling diverse dynamic objects and computational efficiency.

2.3. Our novelties against related works

Most existing 3D reconstruction works focus on building static environments with 3D points, lines and planes, or only 3D points. Most dynamic vSLAM systems focus on improving the accuracy of camera localization. Differently, our work can simultaneously reconstruct scene structure and texture in detail, obtaining clear and accurate 3D point, line, and plane maps, and 3D point clouds even in dynamic environments. Furthermore, we estimate the 6DOF poses of the objects in the scene for clear semantic 3D maps.

When in the 3D point cloud reconstruction mode, our work is close to the Blitz-SLAM [19]. But our work has several notable advantages over theirs: (i) Their method for removing the residual 3D points from moving objects is less accurate than ours. (ii) They regard the keypoints in human masks as outliers and discard them directly, leading to insufficient association of the valid data. (iii) Our work has richer semantic information than theirs, which only segments 20 classes. (iv) Our method can estimate the 6DOF poses of the objects, but their work cannot.

For the 3D point, line and plane reconstruction, our system is close to PLP-SLAM [27] and DRG-SLAM [28]. PLP-SLAM has a similar 3D plane reconstruction process as ours but it cannot be robustly run in dynamic scenes and lacks semantic information. DRG-SLAM focuses only on using plane features from the Agglomerative Hierarchical Clustering method [29] to improve localization in dynamic vSLAM, without rendering semantic information in static mapping. Although semantic information is used in DRG-SLAM, it is only for segmenting the objects in scenes. Differently, in our work, the plane features extracted by deep learning include some semantics such as walls and floors, not restricted by the Manhattan world hypothesis. Our object detection also provides richer semantic information. Moreover, we use the percentage threshold of dynamic feature points within the object detection box and eliminate dynamic features of unknown objects.

3. SLAM^2 system design

The workflow of our proposed SLAM^2 is illustrated in Fig. 2.

Our SLAM system extracts the key points in the scene based on the point feature algorithm in ORB-SLAM3 [30], constructs the line features based on the Line Segment Detector [7], and constructs plane

Table 1
Summary of Related Work in Dynamic SLAM.

Author (Year)	Approach	Dataset	Limitations
Yu et al. (2018) [17]	Optical Flow, Semantic Segmentation	TUM RGB-D	Handles only human dynamics
Sun et al. (2018) [14]	Dense Optical Flow	TUM RGB-D	Small movements only
Cheng et al. (2019) [15]	Optical Flow, Fundamental Matrices	TUM RGB-D	Issues with slow objects
Du et al. (2020) [16]	Conditional Random Fields	TUM RGB-D, Bonn RGB-D	Low accuracy
Liu et al. (2021) [22]	Bayesian Filters, Semantic Segmentation	TUM RGB-D	20 object classes
Fan et al. (2022) [19]	Semantic Segmentation, Depth Mask	TUM RGB-D	Predefined dynamic objects
Hu et al. (2022) [21]	Multi-view Geometry, Semantic Segmentation	TUM RGB-D	Predefined object classes
Wang et al. (2022) [26]	Point-line-plane, Semantic Segmentation, Epipolar Constraints	TUM RGB-D, Bonn RGB-D	20 object classes
Yuan et al. (2023) [24]	Point-line, Semantic Segmentation, Epipolar Constraints, Bayes' Theory	KITTI	Predefined non-priority objects
Wang et al. (2024) [23]	Handcrafted, Learned Features for Point Cloud	TUM RGB-D	Requires preprocessing dynamic parts

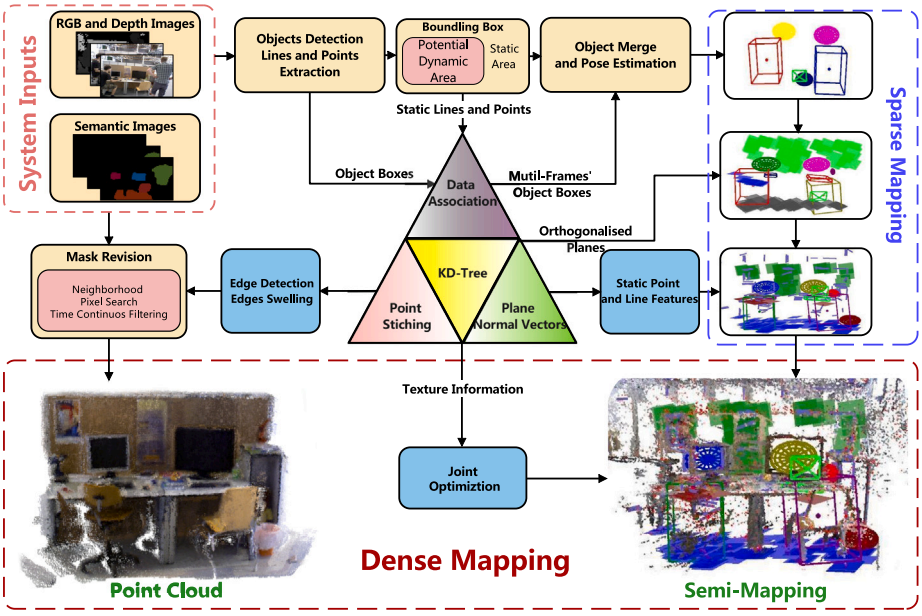

Fig. 2. The workflow of our SLAM^2 system. After RGB and Depth images are input into our system, points, lines and object bounding boxes are extracted. The input semantic images contain plane features of the scene. Next, the system delineates the potentially dynamic area in the boxes judged as dynamic and removes the points and lines inside the area. Meanwhile, object boxes are associated across frames to estimate the corresponding 6DOF poses and merge their 3D model. The static points and lines associated across frames and combined objects 3D models are used for sparse mapping. In dense reconstruction, we use the extracted planes to revise the dense point cloud and apply multiple methods to remove the 3D noisy block generated by moving objects, producing a clean static mapping. In semi-dense reconstruction, we add certain textures to the sparse 3D reconstruction to achieve better reconstruction results.

features by augmenting PLP-SLAM [31] with PlaneRecNet [6] to detect semantic planes.

Then, the system uses YOLOv5 [8] to detect the objects in the scene and combine it with the epipolar constraints of LK optical flow to judge those dynamic objects and reject their corresponding features. We propose a new dynamic feature rejection algorithm that can adapt the threshold value to different object boxes and can handle undetected objects. Finally, we use the obtained static point and line features for 3D reconstruction of the scene in different modes. We use the object boxes associated with multiple frames to compute the 6DOF pose of each object and build 3D models of those static objects. These models, together with the extracted 3D plane and line features, form a sparse reconstruction of the scene. We construct the scene using the static dense point cloud and use Edge Detection, Edges Swelling and Time Continuous Filtering to remove the residual shadows left by moving objects. This allows us to construct a clean 3D static point cloud map in a dynamic scene. We also design a semi-dense reconstruction method by adding texture to the sparse reconstruction for more versatile uses.

3.1. Line features-based SLAM

Our proposed SLAM system integrates line features to enhance its robustness in dynamic scenes and facilitate the treatment of dynamic objects in the map.

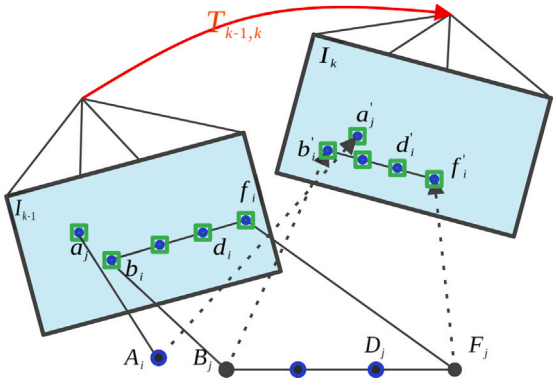

Fig. 3. Line feature matching process in two frames. A definitive match is established based on the similarity of the matched lines' cosine angles and the degree of overlap.

3.1.1. Line features matching

Our system employs a robust line feature-matching methodology for stereo camera configurations. After identifying line features from the stereo camera images with the same grid ID, we compare the line descriptors and select the descriptor with the shortest distance for matching. The process is completed by verifying the overlap threshold

and examining the differences in the cosine angles of direction vectors between matched lines, as illustrated in Fig. 3.

3.1.2. Line feature attributes updating algorithm

We initialize the depth values of line endpoints, the representation of line segments as 3D vectors, and the disparity to remove redundant line features and maintain stable tracking.

We perform bi-directional cosine similarity matching for line-matching pairs and omit those line segments with low similarity. Homogeneous coordinate vectors are formed for the start and end points of each line segment. The line segments with an overlap greater than 0.75 are considered to be stable. We apply a disparity threshold to filter matched line segments and discard those with small disparity values. The depth value D is estimated as $D = \frac{K}{D_v}$, where K is the ratio between the left and right images' focal lengths, and D_v is the disparity value.

In this paper, we minimize the re-projection errors of the point and line features across all camera frames simultaneously and use the plane normal vector and the yaw angle of the object to eliminate redundant line features. This is beneficial to retaining the edge of the object in the real world. Eliminating the redundant line features can also reduce the computational complexity of the nonlinear optimization and improve the positioning accuracy in the SLAM system.

3.2. Constructing planes on the map

The input semantic planar mask image guides the selection of map points to form a planar representation, with a distance threshold determining which points fit the plane. The plane π is represented by a normal vector $\mathbf{n}$ and a distance d to the world origin as follows:

$$\mathbf{n} \cdot (\mathbf{x} - \mathbf{p}) = 0, \ \boldsymbol{\pi} = (\mathbf{n}^{\mathbf{T}}, \mathbf{d}) \tag{1}$$

where $\mathbf{x}$ and $\mathbf{p}$ are two different points on the plane.

The plane fitting problem is formulated as the following optimal binary labeling problem:

$$E(\Theta) = \sum_i \|\Theta_i\| + \lambda \cdot \sum_{(j,k)\in M} \delta(\Theta_j \neq \Theta_k) \tag{2}$$

where Θ_i is the assignment of plane models to 3D point i and $\|\Theta_i\|$ is the geometric error measure indicating the distance between the 3D point i and the plane. M is the neighborhood graph constructed using the Fast Approximate Nearest Neighbors algorithm. λ is an empirically chosen scalar to balance the two terms.

In this paper, the value of $\|\Theta_i\|$ is computed as follows:

$$\|\Theta_i\| = \begin{cases} 0, & \text{when either } (\Theta_i = 1 \wedge \text{dist}(i, \mathbf{n}, d) < q) \\ & \text{or } (\Theta_i = 0 \wedge \text{dist}(i, \mathbf{n}, d) \geq q) \text{ is true.} \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

where $\text{dist}(i, \mathbf{n}, d)$ represents the distance between the 3D point i to the plane defined by $\mathbf{n}$ and d.

We utilize high-quality map points processed in the previous step to create a simulated map. These points are projected onto the map and selected if they are close to planes identified from the semantic images. If the distance between the map points and these planes falls below a specified threshold, these points are chosen to construct planes in different colors to represent the objects in the environment.

3.2.1. Dynamic features culling

The process of plane construction involves selecting two map points that are sufficiently close to a plane and calculating their vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. Next, the normal vector of the plane is determined by crossing the previously calculated vectors:

$$\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2 \tag{4}$$

A point $\mathbf{p}$ and the normal vector are then selected, and a specific distance d is chosen as the side of square planes. The plane is then constructed following (4). The square plane generation procedure is illustrated in Fig. 4.

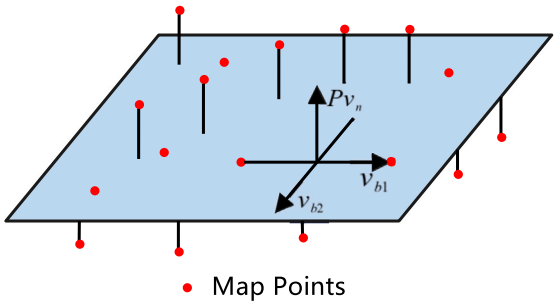

Fig. 4. Illustration of the plane construction.

Algorithm 1 Dynamic points culling algorithm

Input: Bounding boxes B_n, key points for the current frame P_n, dynamic key points for the current frame DP_n.
Output: Static key points for the current frame S_n.

$S_n \leftarrow P_n$.
for each bounding box $B_n(i)$ in B_n **do**
  $bool_a \leftarrow 0$, count $\leftarrow 0$.
  for each dynamic key point (du_n, dv_n) in DP_n **do**
    if (du_n, dv_n) in $DB_n(i)$ **then**
      $bool_a \leftarrow 1$, count $\leftarrow$ count $+ 1$.
    end if
  end for
  if count $> 0.4|P_n|$ **then**
    $S_n \leftarrow S_n \setminus B_n(i)$.
  else
    for each key point (u_n, v_n) in P_n **do**
      if $(u_n, v_n) \in B_n(i)$ **and** $bool_a = 1$ **and** $\sqrt{(u_n - du_n)^2 + (v_n - dv_n)^2} \leq 15$ **then**
        $S_n \setminus (u_n, v_n)$.
      end if
    end for
  end if
end for

3.3. Dynamic features removal

We adopt multiple strategies to reduce the effect of dynamic features on the SLAM system's pose estimation.

Firstly, object detection is used to identify the potential dynamic objects. Then, the LK optical flow algorithm, combined with epipolar constraints, is used to detect the geometric outliers in the current frame. Finally, a dynamic feature removal algorithm is used to filter out those features that may lead to inaccurate estimations.

3.3.1. Dynamic features detection

Our method detects the dynamic points and lines in an input image. It refines the Harris corner matching using the LK optical flow pyramid, and discard the matches near the pixel edge or with high disparity. We classify the points exceeding a predefined distance from their corresponding epipolar line as outliers. A RANSAC algorithm [32] identifies the fundamental matrix with the maximum inliers, forming the basis for computing the polar line.

Let the matched points in the preceding and current frames be p_1 and p_2, respectively, and their homogeneous coordinate forms be P_1 and P_2. Then the following vectors can be obtained:

$$p_1 = [u_1, v_1], \ p_2 = [u_2, v_2], \ P_1 = [u_1, v_1, 1], \ P_2 = [u_2, v_2, 1]. \tag{5}$$

The corresponding epipolar line L_1 is calculated as:

$$L_1 = [X, Y, Z] = FP_1 = F[u_1, v_1, 1] \tag{6}$$

We compute the distance between the matched point and its corresponding epipolar line using

$$D = \frac{|P_2^T F P_1|}{\sqrt{\|X^2\| + \|Y^2\|}} \tag{7}$$

If D exceeds a preset threshold, the feature point is considered as an outlier. The corresponding outlier feature lines and planes are also detected.

After detecting the dynamic feature points, we enhance them with the semantic information within the object boxes. We also propose a local filtering algorithm to retain the key data and reduce the errors. Those points within a 15-pixel radius of a dynamic key point are removed, and boxes with over 40% dynamic points are excluded entirely. For those points outside the semantic boxes, we directly remove the dynamic points detected in the image to eliminate unstable data associations. This enhances our SLAM system's ability in handling unknown objects. The corresponding dynamic feature lines and planes are also removed. The overall algorithm is illustrated in Algorithm 1.

3.4. 6 DoF pose estimation

In this section, we use the Intersection over Union (IoU) Calculation to match the same object across different frames and estimate a cuboid representation of the object in 3D space along with its center location. We then match the detected lines to the cuboid edges across frames to estimate the rotation angle of the object. Finally, we utilize a t-test to merge redundant detections corresponding to the same physical object.

3.4.1. Intersection over union (IoU) calculation

Given the current bounding box B_{curr} and the predicted bounding box B_{pred}, the Intersection over Union (IoU) is calculated as follows:

$$\text{IoU} = \frac{\text{area}(B_{curr} \cap B_{pred})}{\text{area}(B_{curr} \cup B_{pred})} \tag{8}$$

The predicted bounding box B_{pred} is computed based on the bounding boxes of the object in the last two consecutive frames.

3.4.2. Object cuboid estimation

We represent the transformation of 3D point coordinates from the world coordinates X_w to camera coordinates X_c and finally to image coordinates x as

$$X_c = RX_w + t, \; x = PX_c = KRX_w + Kt. \tag{9}$$

In the process of mapping objects within an environment, we first compute the mean 3D position of all points belonging to a certain object as:

$$\text{center} = \frac{1}{N}\sum_{i=1}^{N} pos_i \tag{10}$$

We then calculate the standard deviation in each direction d (which is x, y, or z) using:

$$\text{std}_d = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(pos_{i,d} - \text{mean}_d)^2} \tag{11}$$

Next, we calculate the size of the object by identifying the maximum and minimum values in each direction and calculating the difference. Specifically, the size in direction d is given by

$$\text{size}_d = \max(pos_d) - \min(pos_d) \tag{12}$$

Finally, we update the object's pose by encapsulating the center location and dimensions (length, width, height) into a cuboid. The quadratic surface is generated in the same way.

Algorithm 2 Object Estimation

function TransformToCamera(X_w, R, t, K)
  $X_c \leftarrow RX_w + t$
  $x \leftarrow KRX_w + Kt$
  return x ▷
Inputs: world coordinate X_w, rotation matrix R, translation vector t, and intrinsic matrix K. Outputs: camera coordinate x.
end function
function ComputeCenter(points)
  $N \leftarrow$ length(points)
  return $\frac{1}{N}\sum_{i=1}^{N}$ points$_i$ ▷ Inputs: a set of points. Outputs: the center of the points.
end function
function ComputeStd(points, mean$_d$)
  $N \leftarrow$ length(points)
  return $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{points}_{i,d} - \text{mean}_d)^2}$ ▷ Inputs: a set of points and their mean. Outputs: the standard deviation of the points.
end function
function ComputeSize(points$_d$)
  return max(points$_d$) − min(points$_d$) ▷ Inputs: a set of points in a dimension. Outputs: the size of the points in that dimension.
end function

3.4.3. Object orientation estimation

The function initializes with zero yaw and iteratively computes a rotation matrix, R, based on the yaw value. The corner of the cuboid, P_{WF}, is transformed to the object frame (OF) as P_{OF}, rotated, and projected into the image frame. During each iteration, the function calculates the angle difference between the detected lines in the scene and cuboid edges. If the difference is within a threshold, they are considered parallel. This process is repeated for 30 yaw angles.

The error function and transformation are defined as

$$e(\phi) = \|\beta(\tilde{Y}(\phi)) - \beta(\tilde{Y}_{\text{ref}})\|^2, \quad \tilde{Y}(\phi) = P \cdot S_d \cdot (Q(\phi)X + u) \tag{13}$$

The parameter ϕ is being adjusted to minimize the error. The function β extracts relevant features or properties. $\tilde{Y}$ is the transformed version of the original data X, determined by ϕ, and $\tilde{Y}_{\text{ref}}$ is the reference or desired outcome to achieve by adjusting ϕ. The projection matrix P and scaling factor S_d are used in the transformation of X. The rotation matrix $Q(\phi)$ depends on ϕ, while u is the translation vector used in the transformation of X.

The score for each yaw sample is computed by

$$\text{score} = \frac{numMax}{nAllLineNum} \times (1 - \alpha \times errorYaw), \tag{14}$$

where $\alpha \in (0, 1)$. The yaw orientation is updated using the highest-scoring yaw sample. The overall algorithm is illustrated in Algorithm 2.

3.4.4. Repeat object processing

The feature points are checked for redundancy against existing map points. The difference between map points from the last frame and the current frame is given by:

$$D = \sum_{i=1}^{n} |PosLast_i - PosCurr_i| \tag{15}$$

where $PosLast_i$ and $PosCurr_i$ are the ith elements of the position vectors in the last and current frame, respectively, with n as their dimension. If $D = 0$, the two points are identical.

To handle misidentifications, a two-sample t-test is used to decide whether to merge objects, based on their historical centers. The t-statistics for objects A_1 and A_2 are given by

$$t = \frac{M(A_1) - M(A_2)}{SE} \sim t(|A_1| + |A_2| - 2), \tag{16}$$

$$SE = \sqrt{\frac{\sigma_1^2}{|A_1|} + \frac{\sigma_2^2}{|A_2|}}, \quad \sigma_i = \sqrt{\frac{(|A_i|-1)\sigma_i^2}{|A_1|+|A_2|-2}}, \quad i \in \{1,2\}, \tag{17}$$

where SE is the standard error and σ_i is the standard deviation of A_i. Objects are not merged if t falls in the t-distribution's rejection region with $df = |A_1| + |A_2| - 2$.

3.5. Generation of point cloud

Point cloud generation is critical for map construction and camera pose estimation. Our SLAM system constructs point clouds from keyframes, each of which contains an RGB image and a depth image. The construction begins with applying a median blur to the depth image for noise reduction.

To accurately identify and process the dynamic objects in depth images, we first apply the Sobel operator (a discrete differential operator) to highlight the high spatial frequency regions of the image, i.e. edges. Next, we extend and connect the depth edges within the dynamic object detection frame to draw the edges of the dynamic object. To maintain edge continuity, the flood fill processing is applied from the dynamic edge to the center of the detection frame to generate a binary mask. Subsequently, an edge dilation operation is performed on the edges of the binary mask to eliminate residual point clouds produced by motion blur and noise. Finally, we obtain a binary image mask that culls the point clouds generated by dynamic objects. We also use a time-based filtering algorithm to identify the static pixels. When the pixels in the RGB-D image are outside the dynamic object detection box for three consecutive frames, we consider these pixels to be static and use them to generate point clouds. Through the above steps, we can either compensate for the low accuracy of the object detection algorithm, or conclude that the object detection frame cannot completely contain the point cloud afterimages due to motion blur.

We then iterate over each pixel, checking the validity of the depth and dynamic classification. For those valid pixels, the 3D coordinates are computed using the depth and the camera's intrinsic parameters as follows:

$$x = (u - c_x) \cdot \frac{d}{f_x}, \quad y = (v - c_y) \cdot \frac{d}{f_y}, \quad z = d \tag{18}$$

where u, v are the pixel's image coordinates, c_x, c_y are the camera's principal points, f_x, f_y are the camera's focal lengths, and d is the depth. The point cloud is then transformed to the world coordinates using the camera pose.

In loop closure detection, the point clouds' poses are updated to reflect the adjustments in the keyframe poses. The transformation is given by

$$P_{\text{new}} = T_{\text{adjust}} \cdot T_{KF}^{-1} \cdot P_{KF}, \tag{19}$$

where T_{adjust} is the adjustment transformation, T_{KF} is the keyframe's pose transformation, and P_{KF} is the original points cloud without adjustment.

Finally, we apply outlier removal and voxel grid filtering to reduce the point count. Through the above process, the algorithm we proposed can very well remove the afterimages generated by dynamic objects in the point cloud map, as already shown in Section 4.3.

4. Experimental evaluation

The experiments are conducted on a Linux computer equipped with the Ubuntu 18.04.6 LTS OS, a 12th generation 16-thread Intel® Core™ i5-12600KF CPU, an NVIDIA GeForce RTX 3070Ti GPU, and 16 GB of RAM. For real-time object detection, we employ the YOLOv5-s model. This model is specifically chosen for its balance between accuracy and computational efficiency.

We evaluate the efficacy of our SLAM system on two RGB-D indoor dynamic datasets: the TUM dataset [33] and the Bonn dataset [34].

TUM dataset. The TUM RGB-D dataset is rich in dynamic indoor settings, comprising image sequences with various line and plane densities, human movements, and dynamic objects. These sequences are instrumental in evaluating SLAM systems under different challenging conditions. We select specific sequences, including four walking sequences (`w/half`, `w/rpy`, `w/static`, `w/xyz`), to evaluate our system in a highly dynamic environment. These four sequences represent the different motion modes of the camera. We use the `fr3/long_office_household sequence` and `w/xyz` to evaluate the mapping capability of our system in static and dynamic environments, respectively. This can demonstrate that our work is capable of building clean point cloud maps and recovering scene structures robustly in dynamic environments.

Bonn dataset. The RGB-D dataset from the University of Bonn augments the TUM dataset with 24 new dynamic sequences across various scenes. This expands the limited TUM sequences and enables a more comprehensive evaluation of the SLAM systems. After removing the repetitive and featureless sequences, We conduct experiments on 18 RGB-D sequences. These cover diverse scenarios — people randomly moving in a space, tracking a walking person, a struck/falling balloon, moving/placing boxes, and handling a large obstructing box. The dataset provides a thorough tested for the SLAM system proposed here across crowded spaces, tracked motions, free dynamics, and obstructed views.

Parameter selection and generalizability. To demonstrate the robust performance of the proposed system in various scenarios, we validate our proposed system on the TUM and Bonn datasets. These datasets are widely recognized in the SLAM community and offer a comprehensive range of challenging scenarios, allowing for meaningful comparisons with state-of-the-art methods. These datasets collectively offer a wide range of dynamic environments, from subtle movements to highly dynamic scenes with multiple moving objects. Table 2 provides a detailed overview of the sequences from both the TUM and Bonn datasets used in our experiments. This table highlights the diverse range of dynamic scenarios covered, including varying durations and types of dynamic elements present in each sequence. The consistency of our system's performance across these varied sequences suggests that our chosen parameters are not overly sensitive to specific scene characteristics. For example, the threshold for classifying dynamic objects (40% of feature points within a bounding box) and the radius for local filtering (15 pixels) demonstrated consistent effectiveness across sequences with varying levels of dynamicity and object sizes. This robustness indicates that our method is likely to generalize well to other similar indoor dynamic environments without requiring significant parameter tuning. While we have not extensively tested on datasets beyond TUM and Bonn, the diverse nature of these benchmark datasets provides a strong foundation for evaluating our system's performance. Future work could potentially explore the system's performance on additional datasets to further validate its generalizability. However, the current results on these standard benchmarks provide compelling evidence of our system's robustness and applicability to a wide range of dynamic indoor scenarios.

To offer an in-depth, quantitative, and analytical insight into the experimental results, we utilize the metric of Absolute Trajectory Error (ATE). The ATE reflects the overall consistency of the estimated trajectory and is formulated as

$$A_t = E_t^{-1} S G_t, \tag{20}$$

where S is a similarity transformation aligning the scales of the estimated and ground truth trajectories.

Table 2
Summary of datasets used in experiments.

Dataset	Sequence	Description	Duration	Dynamic Elements
TUM	w/half	Half-sphere orbiting	37.15 s	People gesticulating
	w/rpy	Axis rotation	27.48 s	People gesticulating
	w/static	Static camera	24.83 s	People walking
	w/xyz	Linear translation	42.53 s	People translating
Bonn	balloon	Struck/falling balloon	14.65 s	Moving balloon
	balloon_tracking	Struck/falling balloon	19.74 s	Moving balloon and people
	crowd	People randomly moving	27.31 s	Multiple people
	moving_no_box	Moving objects	26.15 s	Various objects
	placing_no_box	Moving objects	24.13 s	Moving box and people
	person_tracking	Moving people	19.47 s	Multiple people
	synchronous	Moving people	11.13 s	Multiple people

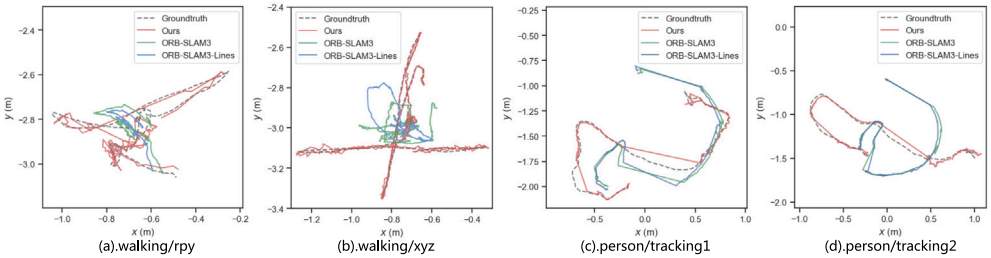

Fig. 5. The contrast of trajectories obtained from ORB-SLAM3, ORB-SLAM3 plus lines features, and our system against the real ground truth trajectories of TUM and Bonn datasets.

Table 3
Comparison between our SLAM system and the state-of-the-art dynamic SLAM systems.

Sequence	t.ATE/m							
	O3 RMSE (S.D.)	O3L RMSE (S.D.)	DeepLab RMSE (S.D.)	SD RMSE (S.D.)	Blitz RMSE (S.D.)	Planar RMSE (S.D.)	DRG RMSE (S.D.)	Ours RMSE (S.D.)
w/half	0.231 (0.008)	0.209 (0.095)	0.027 (0.012)	0.019 (0.010)	0.025 (0.012)	0.325 (–)	0.025 (–)	**0.018 (0.009)**
w/rpy	0.160 (0.073)	0.158 (0.077)	**0.031 (0.018)**	0.053 (0.031)	0.035 (0.022)	0.553 (–)	0.385 (–)	0.033 (0.018)
w/static	0.024 (0.012)	0.020 (0.011)	0.006 (0.002)	**0.005 (0.002)**	0.010 (0.005)	0.293 (–)	0.007 (–)	**0.005 (0.002)**
w/xyz	0.275 (0.145)	0.276 (0.119)	0.013 (0.006)	0.013 (0.008)	0.015 (0.007)	0.276 (–)	0.018 (–)	**0.012 (0.006)**

The best results of RMSE and S.D. are highlighted in bold.

4.1. Experiments on trajectory performance

We evaluate our SLAM system through comparison with the state-of-the-art dynamic SLAM systems. The benchmarks include ORB-SLAM3 (O3) [30], O3L which is an adaptation of ORB-SLAM3 with extra line features without addressing dynamic objects, the DeepLab [21], the SD system [35], Blitz [19], Planar [36] which integrates point-line-plane features, and DRG [26] which is the most advanced point-line-plane feature integration system in dynamic environments. Our focus is on the global robustness and overall performance, measured by the root-mean-square error (RMSE) and standard deviation (S.D.) of the ATE.

Note: For ORB-SLAM3 (O3), O3L, and our proposed system, we conducted evaluations using the same hardware setup and dataset sequences. The results for algorithms such as Planar, DRG, etc., are taken from their respective publications, as their source code are not available.

The results of the TUM high dynamic dataset are reported in Table 3. The results reveal that our SLAM system outperforms other SLAM systems and has good robustness in high dynamic scenes, even though it is slightly less effective than DeepLab in the `w/rpy` sequence.

The results in Table 3 reveal that our SLAM^2 outperforms other SLAM systems, demonstrating robustness in high dynamic scenes, even though it is slightly less effective than DeepLab in the `w/rpy` sequence. Fig. 5 shows that our dynamic removal algorithm improves the accuracy of O3 and O3L.

To further demonstrate the robustness of our system, we compare it with the latest SLAM systems as shown in Table 4, including DS-SLAM [17] (referred to as DS), Dyna-SLAM [37] (referred to as Dyna), DGS-SLAM [38] (referred to as DGS), Planner, and DRG, on the Bonn dataset. While our system struggles in the crowd, `placing_no_box` and `removing_no_box` series, it achieves state-of-the-art performance in all other sequences. In the `crowd2` sequence, our system performs slightly worse than DGS, since DGS's coarse tracking based on residual motion models can handle moving objects better,

Table 4
Comparison between our SLAM system and the state-of-the-art SLAM systems.

Sequence	t.ATE/m							
	O3	O3L	DS	Dyna	DGS	Planar	DRG	Ours
balloon	0.051	0.061	0.035	0.031	0.023	0.145	0.026	**0.019**
balloon2	0.141	0.075	0.037	0.026	0.025	0.162	0.023	**0.020**
balloon_tracking	0.081	0.035	0.032	0.042	0.032	0.087	0.035	**0.026**
balloon_tracking2	0.087	0.031	0.062	0.031	0.028	0.151	**0.027**	**0.027**
crowd	0.474	0.366	0.343	0.017	0.018	0.590	0.188	**0.016**
crowd2	0.583	0.178	0.282	0.027	**0.023**	0.718	0.146	0.037
crowd3	0.358	0.165	0.246	**0.023**	0.024	0.576	0.064	0.024
moving_no_box	0.160	0.187	0.287	0.029	0.018	0.129	0.067	**0.014**
moving_no_box2	0.128	0.096	0.057	0.030	0.028	0.166	0.027	**0.019**
removing_no_box	0.094	0.014	0.019	0.017	0.015	0.055	0.021	**0.014**
removing_no_box2	0.072	0.022	0.023	0.023	0.020	0.072	**0.017**	0.019
placing_no_box	0.717	0.167	0.287	0.125	**0.016**	0.268	**0.016**	0.018
placing_no_box2	0.061	0.035	0.071	0.022	0.028	0.140	**0.015**	0.019
placing_no_box3	0.174	0.122	0.124	0.065	0.034	0.205	0.033	**0.032**
person_tracking	0.590	0.307	0.030	0.039	0.061	0.224	0.033	**0.024**
person_tracking2	0.756	0.561	0.096	**0.030**	0.048	0.607	0.042	**0.030**
synchronous	0.712	0.206	0.108	0.011	0.040	0.597	0.037	**0.005**
synchronous2	0.824	0.223	0.101	0.008	**0.006**	0.785	0.010	0.007

The best results of RMSE are highlighted in bold and the second-best are underlined.

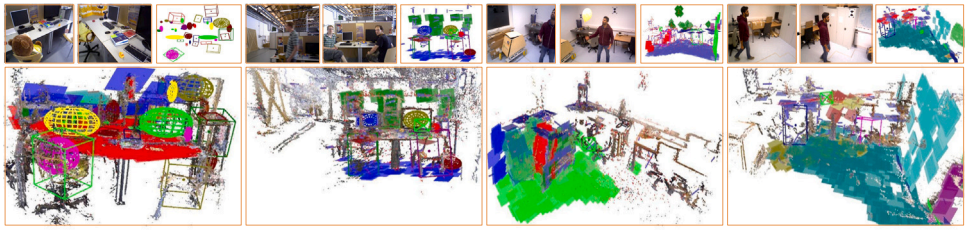

Fig. 6. Experimental results of semi-dense map construction using our proposed method. (a) `fr3/long_office_household` and (b) `fr3/w/xyz` sequences from the TUM dataset; (c) `balloon2` and (d) `person_tracking2` sequences from the Bonn dataset. These images demonstrate various stages of the mapping process, including lightweight reconstruction and 6DOF pose estimation. The results highlight our method's robust handling of dynamic objects (such as moving people and balloons) while accurately mapping static elements in diverse indoor environments. The consistency across different datasets showcases the adaptability of our approach to various dynamic scenarios.

when people occupy relatively more of the view and move faster as in the `crowd2` sequence. However, in sequences with fewer people, DGS's strategy of partitioning some points into unknown subsets results in less stable data associations. Additionally, in such cases, DGS relies solely on initial pose estimates to detect and filter dynamics of clusters, making the overall method susceptible to noise in initial pose estimation. Furthermore, semantic segmentation can assist both Dyna and DGS in acquiring more precise boundaries for segmented moving objects. This facilitates robust performance from these two approaches on the `crowd2` sequence, where people extensively occupy the field of view, as well as the `crowd3` sequence, which contains many small moving objects such as phones and laptops.

The sequence length of `removing_no_box2` is twice that of `removing_no_box`, with longer static scenes and richer planar information. Compared to our method, DRG lacks consistency in judging whether different category objects are dynamic and cannot detect the box in the scene (it can only detect 20 classes), easily misjudging dynamic boxes as static. Therefore, DRG performs worse than our work on the `removing_no_box` sequence where the proportion of dynamic duration is long, while it is slightly better than our system on the `removing_no_box2` sequence where the proportion is small. For the `placing_no_box` and other sequences where our system does not perform the best, the gap between our work and other works is relatively small.

This is particularly notable in high dynamic sequences, where our performance holds a significant advantage. Note that we did not perform additional experiments when we could not construct a plane. This is reasonable because the plane construction process does not involve modifying or culling map points, so it has no impact on trajectory performance.

4.2. Experiments on semi-dense map construction

These experiments demonstrate the effectiveness of our proposed innovative semi-dense map construction method in dynamic environments. Our method can attenuate the disturbance of motion noise to the system. We show the results of four sets of images in Fig. 6, including `fr3/long_office_ household` and `fr3/w/xyz` from the TUM dataset, and `balloon2` and `person _tracking2` from the Bonn dataset. As shown in Fig. 6, each set consists of smaller images depicting different scenes in the dataset, and a lightweight semi-dense reconstruction with lines, planes, and a 6DOF pose estimate of the object. The larger image beneath provides a textured semi-dense reconstruction demonstrating the removal of dynamic objects and accurately locating static objects.

First, we traverse all the keyframes to obtain all the key points used to generate the planar map. An object detection algorithm detects whether the coordinates of these key points in the image coordinate system are within the dynamic object detection box or judged as dynamic points outside the box; if so, they are removed. This is beneficial for experimenting with the balloon sequence in the Bonn dataset for unknown potential dynamic objects. By extracting the plane mask of the balloon, the key points in the mask that do not satisfy the constraints are eliminated and do not participate in the plane reconstruction. This process builds sparse semi-dense maps with geometric information, proving the robustness of our algorithm to known and unknown dynamic object interference.

In additional experiments, such as the `fr3/long_office_household` and `fr3/w/xyz` sequences in the TUM dataset, our algorithm can accurately estimate the 6DOF pose and map static objects like books, computers, etc. The method reconstructs more geometric structures from the original scene by employing point-line-plane representation for semi-dense mapping, showing strong robustness to dynamic noise(humans, chairs in w/xyz sequence).

The experiments in this subsection highlight the adaptability of our method, allowing lightweight point-line-plane reconstruction and post-processing probabilistic mapping. The ability to accurately locate static objects, whilst excluding dynamic objects, emphasizes the precision and texture clarity of the generated maps.

The contribution of our method to semi-dense visual SLAM in dynamic environments includes the extraction of richer geometric information and the potential to choose different mapping modes according to varying task requirements.

4.3. Experiments on point cloud map construction

In our experiments, we aim to validate the efficacy of our proposed algorithm for point cloud reconstruction and also test it on long sequences and closed-loop environments. Fig. 7 illustrates the results from the static sequence of the Bonn dataset. The top row consists of representative images from the dataset, while the following three rows depict the results of our system. These results demonstrate our system's ability to construct a detailed and undistorted point cloud image, in the presence of a closed loop. This accuracy is further enhanced by updating the pose of the point cloud based on the adjusted camera pose during loop closure.

In a high dynamic sequence `fr3/w/xyz` in TUM dataset, moving people often cause boundary information leakage(Floating point clouds appearing in the map). Our system, however, manages to eliminate this effect, leaving virtually no residual information about persons in the point cloud map. On the maps obtained by Dyna and DS, two chairs appear multiple times due to their frequent repositioning by individuals. But our map only retains the most recent chair information before human interaction. As a result, the two chairs are present only once in our point cloud map. The fixed objects like computer monitors remain unaffected by these changes. Fig. 8 illustrates three different perspectives of the point cloud maps acquired by four SLAM systems in the dynamic sequence `fr3/w/xyz`. In the first column, the objects appear entirely distorted for DS. Although DS is capable of reducing some dynamic noise, it fails to exclude the information of the two individuals. The second and third columns reveal that Dyna has almost eliminated the human information, though their contours

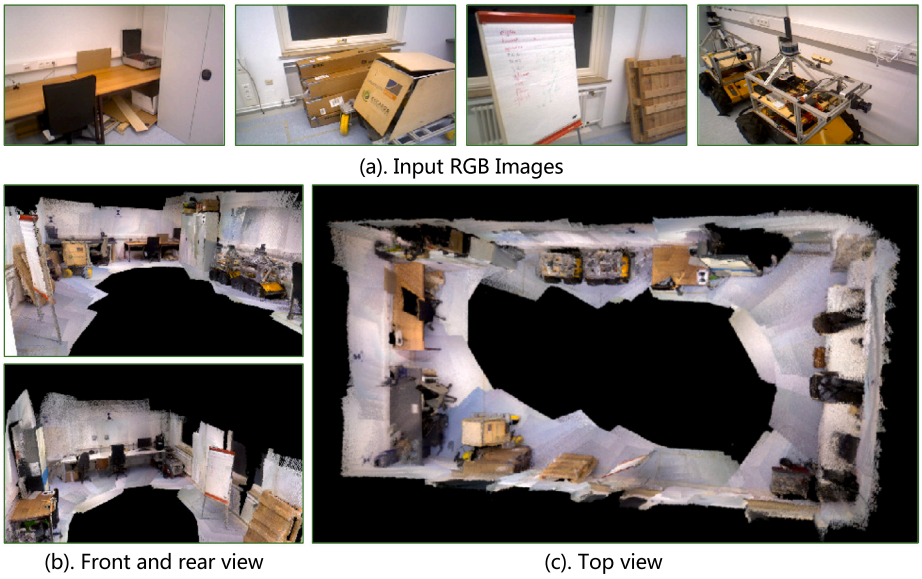

Fig. 7. Static point cloud reconstruction on Bonn dataset using our algorithm. (a) Input images from the dataset. (b) Front and rear views of the reconstructed point cloud. (c) Top view of the reconstructed point cloud, showcasing the spatial layout of the scene.

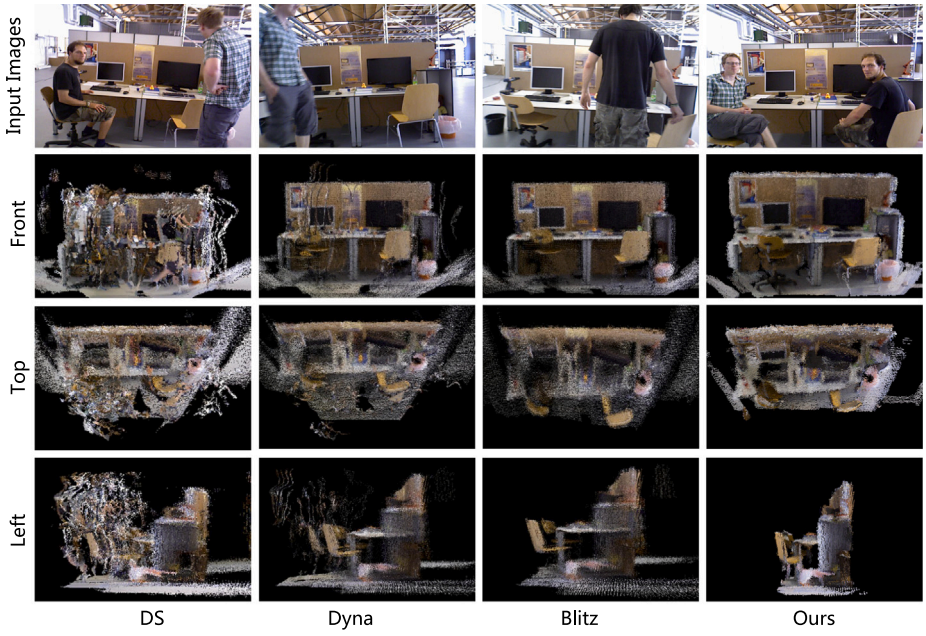

Fig. 8. Comparison of point cloud reconstructions on the TUM dataset. First row: Input images. Columns from left to right show results from: DS-SLAM, Dyna-SLAM, Blitz-SLAM, and our proposed method. Each column demonstrates the performance of different SLAM systems in handling dynamic scenes and reconstructing static environments.

are still discernible from noise blocks. This issue is likely due to a too-small human body region in the semantic segmentation stage and close proximity of a person's leg to the table, leading to the failure of subsequent geometric methods. Blitz, as seen in the third column, does remove noise blocks formed by the individuals, but it lacks clarity in the details such as chairs, desk items, and trash cans. In summary, our system's global point cloud map in the high dynamic sequences outperforms the other three SLAM systems, providing a more precise and robust representation.

Furthermore, we conduct experiments in the highly dynamic environment of the Bonn dataset and compare our enhanced algorithm with the original version that does not include side edge detection. Fig. 9 illustrates the differences between the original algorithm and our improved algorithm. Although capable of handling most noise interference, the original algorithm leaves residual afterimages of humans. This is possibly due to motion blur or inaccuracies in the object detection box, which hampers subsequent advanced robotic navigation or path planning tasks. In contrast, our improved algorithm, employing deep edge detection and time filtering, effectively eliminates dynamic noise without compromising the accuracy of the static information. The experiments thus underscore the robustness of our algorithm to handling disturbances in highly dynamic environments.

4.4. Computational analysis

Table 5 reports the average processing time for each major component of our system compared to state-of-the-art methods. Our SLAM^2 system achieves real-time performance with an average frame rate of

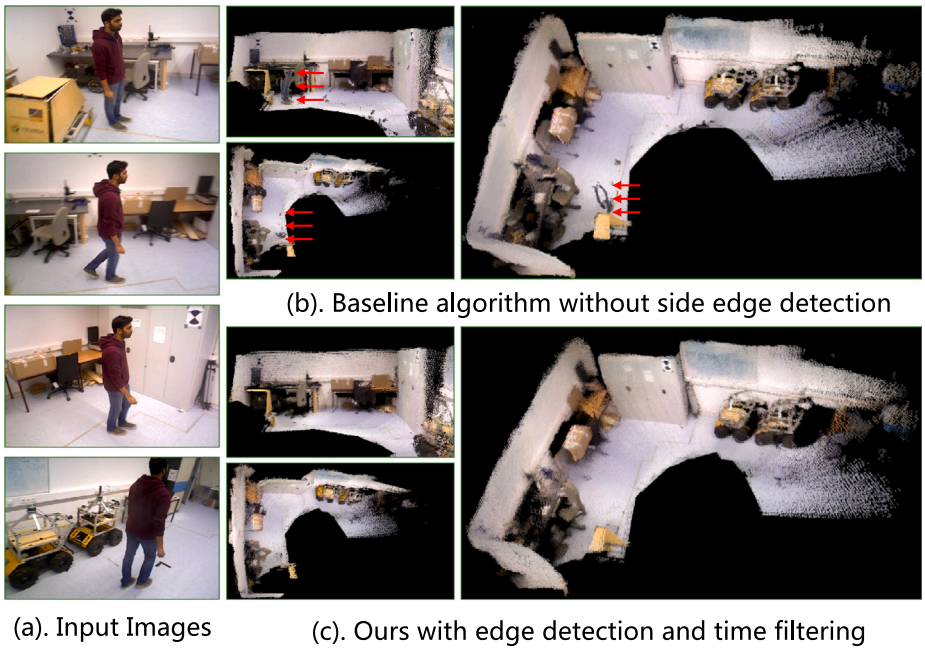

Fig. 9. Comparison of point cloud reconstruction methods. (a) Input images. (b) Results from the baseline algorithm without side edge detection. (c) Results from our improved algorithm with deep edge detection and time filtering. Our improved algorithm effectively culls dynamic points, producing a cleaner static point cloud.

Table 5
Computational analysis of SLAM^2 and State-of-the-Art methods.

Component	Time (ms)				
	O3	DS	Dyna	Blitz	Ours
Feature Extraction	13.6	9.4	9.1	22	16.7
Dynamic Object Detection	–	29.5	419.4	59.0	36.3
Pose Estimation	8.4	7.2	8.1	–	7.5
Mapping	18.7	15.3	17.6	–	18.1
Total	40.7	61.4	454.2	81.0	78.6
Frame Rate (fps)	24.6	16.3	2.2	12.3	12.7

12.7 fps. While slightly lower than the O3, latest static SLAM, our frame rate is comparable to other dynamic SLAM methods. The additional computational cost from multi-feature fusion and dynamic object handling is offset by parallel processing and GPU acceleration, maintaining real-time performance above 10 fps even in highly dynamic scenes. Table 5 presents the average processing time for each major component of our system compared to state-of-the-art methods.

4.5. Robustness and stability

Our system's robustness and stability are demonstrated by consistent performance across challenging scenarios in the TUM and Bonn datasets. In highly dynamic TUM sequences, we consistently outperform or match the best existing methods (Table 3). For the Bonn dataset, we achieve state-of-the-art performance in 14 out of 18 sequences (Table 4).

The system's stability is further evidenced by its performance in long sequences and closed-loop environments, as shown in our point cloud reconstruction experiments (Section 4.3). These results, combined with our accuracy measurements and computational analysis, demonstrate the effectiveness, robustness, and stability of our SLAM^2 system across multiple performance metrics and challenging scenarios.

5. Conclusion

We proposed SLAM^2 system that performs sparse, dense, and semi-dense map reconstruction while estimating 6DOF poses of objects in indoor dynamic environments. Our system integrates point, line, and plane features with semantic information. Key strengths include robust performance in dynamic environments, multi-feature fusion, versatile mapping modes, advanced object pose estimation, and effective dynamic object handling. The comprehensive experiments have demonstrated the superiority and effectiveness of our method over the state-of-the-art SLAM systems. The detailed implementation is open-source and can be accessed at: https://github.com/SEAL-UofG/SLAM2. However, we acknowledge certain limitations of our current system. The integration of multiple features and deep learning components increases computational complexity, which may affect real-time performance on less powerful hardware. Additionally, our system may face challenges in handling partially dynamic or very slow-moving objects, since slowly moving static objects satisfy both the static assumption and the epipolar constraint assumption, such as a slowly opening door. The scalability of our approach for very long sequences or large-scale environments also requires further investigation. In the future, we plan to devise more lightweight object detection models to address the aforementioned limitations by optimizing computational efficiency, improving the handling of partially dynamic objects, and extending our testing to larger-scale environments. These efforts will further enhance the robustness and applicability of our SLAM system across a wider range of scenarios. We will also consider applying the clean static 3D map obtained in this paper to complete advanced robotic tasks such as navigation, exploration, and grasping. The proposed method is poised to facilitate more precise navigation and manipulation in dynamic indoor environments, provide accurate spatial understanding for seamless integration of virtual objects in real-world scenes, enhance spatial awareness for home automation and security applications, and create detailed, semantic-rich maps of complex indoor spaces for facility management or emergency response planning.

CRediT authorship contribution statement

Zhihao Lin: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Qi Zhang:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal

analysis, Conceptualization. **Zhen Tian:** Data curation. **Peizhuo Yu:** Data curation. **Ziyang Ye:** Data curation. **Hanyang Zhuang:** Writing – review & editing, Supervision, Funding acquisition. **Jianglin Lan:** Writing – review & editing, Supervision, Resources, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jianglin Lan reports financial support was provided by Leverhulme Trust. Jianglin Lan reports financial support and travel were provided by The Royal Society. Zhihao Lin reports financial support was provided by China Scholarship Council. Jianglin Lan reports a relationship with Leverhulme Trust that includes: funding grants. Jianglin Lan reports a relationship with The Royal Society that includes: funding grants. Zhihao Lin reports a relationship with China Scholarship Council that includes: funding grants. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We use the public dataset to test our method. The detailed implementation is open-source and can be accessed at: https://github.com/SEAL-UofG/SLAM2.

References

[1] Y.M. Kim, S. Ryu, I.-J. Kim, Planar abstraction and inverse rendering of 3D indoor environments, IEEE Trans. Vis. Comput. Graphics 27 (6) (2021) 2992–3006, http://dx.doi.org/10.1109/TVCG.2019.2960776.

[2] B. Fang, G. Mei, X. Yuan, L. Wang, Z. Wang, J. Wang, Visual SLAM for robot navigation in healthcare facility, Pattern Recognit. 113 (2021) 107822, http://dx.doi.org/10.1016/j.patcog.2021.107822.

[3] J.C.V. Soares, M. Gattass, M.A. Meggiolaro, Crowd-SLAM: Visual SLAM Towards Crowded Environments using Object Detection, J. Intell. Robot. Syst. 102 (2) (2021) 50, http://dx.doi.org/10.1007/s10846-021-01414-1.

[4] C. Li, L. Yu, S. Fei, Large-scale, real-time 3D scene reconstruction using visual and IMU sensors, IEEE Sens. J. 20 (10) (2020) 5597–5605, http://dx.doi.org/10.1109/JSEN.2020.2971521.

[5] J. Dong, Y. Cong, G. Sun, T. Zhang, Lifelong robotic visual-tactile perception learning, Pattern Recognit. 121 (2022) 108176, http://dx.doi.org/10.1016/j.patcog.2021.108176.

[6] Y. Xie, F. Shu, J.R. Rambach, A. Pagani, D. Stricker, PlaneRecNet: Multi-task learning with cross-task consistency for piece-wise plane detection and reconstruction from a single RGB image, in: British Machine Vision Conference, 2021.

[7] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, LSD: A fast line segment detector with a false detection control, IEEE Trans. Pattern Anal. Mach. Intell. 32 (4) (2010) 722–732, http://dx.doi.org/10.1109/TPAMI.2008.300.

[8] G. Jocher, YOLOv5 by Ultralytics, 2020, http://dx.doi.org/10.5281/zenodo.3908559, URL https://github.com/ultralytics/yolov5.

[9] Y. He, J. Zhao, Y. Guo, W. He, K. Yuan, PL-VIO: Tightly-coupled monocular visual–Inertial odometry using point and line features, Sensors 18 (4) (2018) http://dx.doi.org/10.3390/s18041159.

[10] X. Zhang, W. Wang, X. Qi, Z. Liao, R. Wei, Point-plane SLAM using supposed planes for indoor environments, Sensors 19 (17) (2019) http://dx.doi.org/10.3390/s19173795.

[11] C. Arndt, R. Sabzevari, J. Civera, From points to planes - adding planar constraints to monocular slam factor graphs, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2020, pp. 4917–4922, http://dx.doi.org/10.1109/IROS45743.2020.9340805.

[12] Y. Li, N. Brasch, Y. Wang, N. Navab, F. Tombari, Structure-SLAM: Low-drift monocular SLAM in indoor environments, IEEE Robot. Autom. Lett. 5 (4) (2020) 6583–6590, http://dx.doi.org/10.1109/LRA.2020.3015456.

[13] S. Yang, Y. Song, M. Kaess, S. Scherer, Pop-up SLAM: Semantic monocular plane SLAM for low-texture environments, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2016, pp. 1222–1229, http://dx.doi.org/10.1109/IROS.2016.7759204.

[14] Y. Sun, M. Liu, M.Q.-H. Meng, Motion removal for reliable RGB-d SLAM in dynamic environments, Robot. Auton. Syst. 108 (2018) 115–128, http://dx.doi.org/10.1016/j.robot.2018.07.002.

[15] J. Cheng, Y. Sun, M.Q.-H. Meng, Improving monocular visual SLAM in dynamic environments: an optical-flow-based approach, Adv. Robot. 33 (12) (2019) 576–589, http://dx.doi.org/10.1080/01691864.2019.1610060.

[16] Z.-J. Du, S.-S. Huang, T.-J. Mu, Q. Zhao, R.R. Martin, K. Xu, Accurate dynamic SLAM using CRF-based long-term consistency, IEEE Trans. Vis. Comput. Graphics 28 (4) (2022) 1745–1757, http://dx.doi.org/10.1109/TVCG.2020.3028218.

[17] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, Q. Fei, DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2018, pp. 1168–1174, http://dx.doi.org/10.1109/IROS.2018.8593691.

[18] V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (12) (2017) 2481–2495, http://dx.doi.org/10.1109/TPAMI.2016.2644615.

[19] Y. Fan, Q. Zhang, Y. Tang, S. Liu, H. Han, Blitz-SLAM: A semantic SLAM in dynamic environments, Pattern Recognit. 121 (2022) 108225.

[20] N. Dvornik, K. Shmelkov, J. Mairal, C. Schmid, BlitzNet: A Real-Time Deep Network for Scene Understanding, in: 2017 IEEE International Conference on Computer Vision, ICCV, 2017, pp. 4174–4182, http://dx.doi.org/10.1109/ICCV.2017.447.

[21] Z. Hu, J. Zhao, Y. Luo, J. Ou, Semantic SLAM Based on Improved DeepLabv3+ in Dynamic Scenarios, IEEE Access 10 (2022) 21160–21168, http://dx.doi.org/10.1109/ACCESS.2022.3154086.

[22] Y. Liu, J. Miura, RDS-SLAM: Real-Time Dynamic SLAM Using Semantic Segmentation Methods, IEEE Access 9 (2021) 23772–23785, http://dx.doi.org/10.1109/ACCESS.2021.3050617.

[23] J. Wang, Y. Qi, Visual camera relocalization using both hand-crafted and learned features, Pattern Recognit. 145 (2024) 109914, http://dx.doi.org/10.1016/j.patcog.2023.109914.

[24] C. Yuan, Y. Xu, Q. Zhou, PLDS-SLAM: Point and Line Features SLAM in Dynamic Environment, Remote Sens. 15 (7) (2023) http://dx.doi.org/10.3390/rs15071893.

[25] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: 2017 IEEE International Conference on Computer Vision, ICCV, 2017, pp. 2980–2988, http://dx.doi.org/10.1109/ICCV.2017.322.

[26] Y. Wang, K. Xu, Y. Tian, X. Ding, DRG-SLAM: A Semantic RGB-D SLAM using Geometric Features for Indoor Dynamic Scene, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2022, pp. 1352–1359, http://dx.doi.org/10.1109/IROS47612.2022.9981238.

[27] F. Shu, J. Wang, A. Pagani, D. Stricker, Structure PLP-SLAM: Efficient sparse mapping and localization using point, line and plane for monocular, RGB-D and stereo cameras, in: IEEE International Conference on Robotics and Automation, IEEE, 2023, pp. 2105–2112.

[28] Y. Wang, K. Xu, Y. Tian, X. Ding, DRG-SLAM: A Semantic RGB-D SLAM using Geometric Features for Indoor Dynamic Scene, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2022, pp. 1352–1359.

[29] C. Feng, Y. Taguchi, V.R. Kamat, Fast plane extraction in organized point clouds using agglomerative hierarchical clustering, in: 2014 IEEE International Conference on Robotics and Automation, ICRA, 2014, pp. 6218–6225, http://dx.doi.org/10.1109/ICRA.2014.6907776.

[30] C. Campos, R. Elvira, J.J.G. Rodríguez, J.M. M. Montiel, J. D. Tardós, ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM, IEEE Trans. Robot. 37 (6) (2021) 1874–1890, http://dx.doi.org/10.1109/TRO.2021.3075644.

[31] F. Shu, J. Wang, A. Pagani, D. Stricker, Structure PLP-SLAM: Efficient sparse mapping and localization using point, line and plane for monocular, RGB-d and stereo cameras, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, 2023, pp. 2105–2112, http://dx.doi.org/10.1109/ICRA48891.2023.10160452.

[32] M.A. Fischler, R.C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, Commun. ACM 24 (6) (1981) 381–395.

[33] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, A benchmark for the evaluation of RGB-D SLAM systems, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 573–580, http://dx.doi.org/10.1109/IROS.2012.6385773.

[34] M. Menze, C. Heipke, A. Geiger, Object scene flow, ISPRS J. Photogram. Remote Sens. (JPRS) (2018).

[35] Q. Zhang, C. Li, Semantic SLAM for mobile robots in dynamic environments based on visual camera sensors, Meas. Sci. Technol. 34 (8) (2023) 085202, http://dx.doi.org/10.1088/1361-6501/acd1a4.

[36] Y. Li, R. Yunus, N. Brasch, N. Navab, F. Tombari, RGB-d SLAM with structural regularities, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, 2021, pp. 11581–11587, http://dx.doi.org/10.1109/ICRA48506.2021.9561560.

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

[38] L. Yan, X. Hu, L. Zhao, Y. Chen, P. Wei, H. Xie, DGS-SLAM: A fast and robust RGBD SLAM in dynamic environments combined by geometric and semantic information, Remote Sens. 14 (3) (2022) http://dx.doi.org/10.3390/rs14030795.

Zhihao Lin, a graduate of Liaocheng University (2018), pursued an M.S. at Jilin University and is now a Ph.D. candidate at the University of Glasgow. His research concentrates on multi-sensor fusion SLAM systems and robotic perception in challenging environments.(39 words)

Qi Zhang received his B.S. degree from North University of China in 2022, pursued an MSc degree from the University of Glasgow, UK and is now a Ph.D. student at the University of Amsterdam, Netherlands. His current research interests include algorithms and systems for SLAM.(45 words)

Zhen Tian earned his B.S. in Electronic and Electrical Engineering from the University of Strathclyde in 2020 and is now a Ph.D. student at the University of Glasgow, focusing on interactive vehicle decision systems and autonomous racing strategies.(38 words)

Ziyang Ye received his Bachelor's degree in Computer Science from the University of Adelaide, Adelaide, South Australia, Australia, in 2020. He is currently pursuing his Master's degree in Artificial Intelligence and Machine Learning at the same institution. His research interests include 2/3D Computer Vision tasks and Reinforcement Learning.(49 words)

Peizhuo Yu received his B.Eng from Beijing University of Chemical Technology, China, 2017. He is currently pursuing a PhD in the School of Engineering, University of Glasgow, UK. His research interests lie in planning and control for mobile robots.(39 words)

Hanyang Zhuang, a Ph.D. graduate from Shanghai Jiao Tong University in 2018, served as a postdoctoral researcher there until 2022. He is now an assistant research professor at the same university, focusing on intelligent vehicles, with particular interest in SLAM and cooperative driving systems.(44 words)

Jianglin Lan, with a Ph.D. from Hull University, held a postdoc position at Imperial College London before becoming a Glasgow University Lecturer in 2022 and a Visiting Professor at Carnegie Mellon in 2023. His research spans AI, optimization, control theory, and autonomy.(37 words)