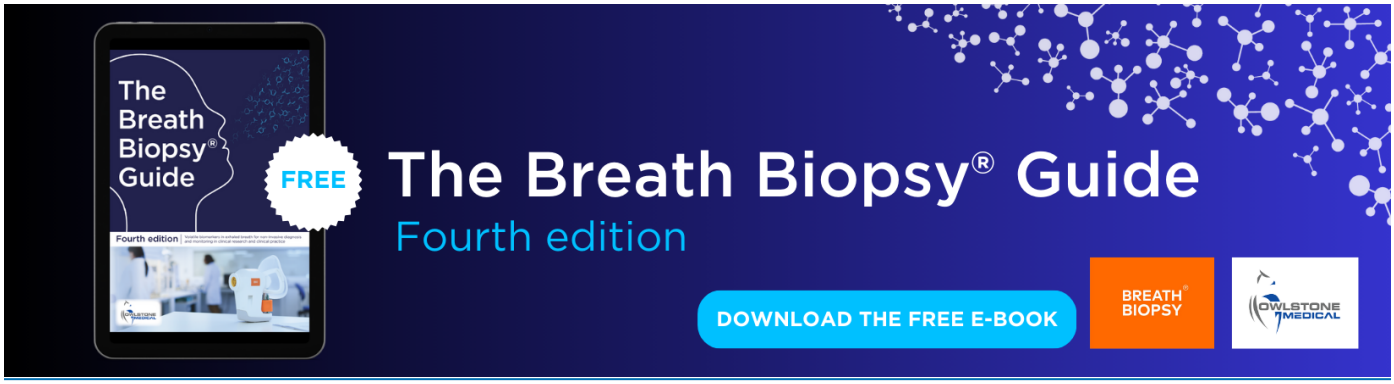
Measurement Science and Technology

PAPER

YES-SLAM: YOLOv7-enhanced-semantic visual SLAM for mobile robots in dynamic scenes

To cite this article: Hang Liu and Jingwen Luo 2024 *Meas. Sci. Technol.* **35** 035117



IOP Publishing Measurement Science and Technology
Meas. Sci. Technol. **35** (2024) 035117 (19pp) https://doi.org/10.1088/1361-6501/ad14e7

YES-SLAM: YOLOv7-enhanced-semantic visual SLAM for mobile robots in dynamic scenes

Hang Liu[1] **and Jingwen Luo**[1,2,*]

[1] School of Information Science and Technology, Yunnan Normal University, No.768 Juxian Street, Chenggong District, Kunming, Yunnan 650500, People's Republic of China
[2] Department of Education of Yunnan Province, Engineering Research Center of Computer Vision and Intelligent Control Technology, Kunming, Yunnan, People's Republic of China

E-mail: by1503117@buaa.edu.cn

Received 2 August 2023, revised 7 December 2023
Accepted for publication 12 December 2023
Published 19 December 2023

Abstract

In dynamic scenes, moving objects will cause a significant error accumulation in robot's pose estimation, and might even lead to tracking loss. In view of these problems, this paper proposes a semantic visual simultaneous localization and mapping algorithm based on YOLOv7. First, a light-weight network YOLOv7 is employed to acquire the semantic information of different objects in the scene, and flood filling and edge-enhanced techniques are combined to accurately and quickly separate the dynamic feature points from the extracted feature point set. In this way, the obtained static feature points with high-confidence are used to achieve the accurate estimation of robot's pose. Then, according to the semantic information of YOLOv7, the motion magnitude of the robot, and the number of dynamic feature points in camera's field-of-view, a high-performance keyframe selection strategy is constructed. On this basis, a robust loop closure detection method is developed by introducing the semantic information into the bag-of-words model, and global bundle adjustment optimization is performed on all keyframes and map points to obtain a global consistent pose graph. Finally, YOLOv7 is further utilized to carry out semantic segmentation on the keyframes, remove the dynamic objects in its semantic mask, and combine the point cloud pre-processing and octree map to build a 3D navigation semantic map. A series of simulations on TUM dataset and a case study in real scene clearly demonstrated the performance superiority of the proposed algorithms.

Keywords: dynamic scenes, simultaneous localization and mapping (SLAM), YOLOv7, depth camera, loop closure detection, 3D semantic map

1. Introduction

In recent years, as the widely application of intelligent robots in various fields, simultaneous localization and mapping (SLAM) has become a frontier research hotspot in the field of autonomous navigation. Typically, most SLAM methods treat the environment as static, resulting in a dynamic scene in which the robot treats dynamic objects as part of a static background. When dynamic targets occupy most of the robot's field-of-view, a large amount of dynamic error is undoubtedly introduced, which makes the accuracy and robustness of SLAM negatively affected. In order to solve the problem that SLAM systems are easily disturbed by moving objects, methods such as Random sample consensus (RANSAC) [1] and bundle adjustment (BA) [2] have been developed to remove some of the dynamic feature points. However, these methods do not perform well in the case of there are too many dynamic points. Therefore, developing an efficient method to identify and remove dynamic feature points in dynamic scenes

* Author to whom any correspondence should be addressed.

is the key to improving the stability and accuracy of SLAM systems.

Currently, the solutions of SLAM in dynamic scenes are usually optimized by introducing detection techniques for dynamic targets based on the architecture of existing open source SLAM schemes. As a result, accurately detecting dynamic objects and filtering out dynamic features in the image has become the key to this work. Most recently, with the rapid development of deep learning, more and more researchers have applied various kinds of deep neural networks to develop SLAM solutions in dynamic scenes.

Zhang *et al* [3] employed the MobileNets [4] to detect dynamic semantic features in the scene and combined the K-means clustering for depth images to cluster the depth values of point-line features, and then removed dynamic feature points based on the clustering results. Wu *et al* [5] proposed a SLAM algorithm combining target detection and clustering, which used the YOLO [6] target detection algorithm and Euclidean distance clustering to optimize the camera's pose and constructed a more accurate dense point cloud map in dynamic scenes. Ai *et al* [7] proposed an improved version of ORB-SLAM2 [8] that employed DUNet [9] to segment dynamic objects and embed background, successfully detecting moving objects by using semantic segmentation and multi-view geometry. Wang *et al* [10] utilized YOLOv3 [11] to detect specific moving objects and capitalized on flood fill algorithm (FFA) to extract dynamic object contours within the bounding box of object, achieving semantic segmentation and semantic map construction. Similarly, Wu *et al* [12] exploited YOLOv3 to recognize dynamic objects and combined it with the depth-RANSAC algorithm to filter dynamic feature points. Yu *et al* [13] presented a robust semantic visual SLAM (VSLAM) based on ORB-SLAM2, and called as 'DS-SLAM', which combined semantic segmentation and motion consistency check to filter the dynamic part of the scene, improved the accuracy and robustness of the algorithm in dynamic environments, and generated a dense semantic octree map that can be used for advanced tasks. Bescós *et al* [14] proposed a DynaSLAM algorithm that used the instance segmentation network Mask R-CNN [15] to segment dynamic objects in the image pixel by pixel, thereby preventing the SLAM algorithm from extracting features on dynamic objects. After this, Xiao *et al* [16] developed a monocular vision Dynamic-SLAM system. The method used object detection network single shot multibox detector (SSD) [17] to detect moving objects, and further proposed a missed detection compensation algorithm based on the invariance of adjacent frame speed to address the problem of low recall rate, greatly improving the recall rate of detection. In [18], a real-time SLAM approach was achieved by effective utilizing SegNet [19] semantic segmentation network to segment keyframes and combining K-Means algorithm for classification, which was capable of quickly rejecting dynamic objects without consuming excessive system resources. The DGS-SLAM proposed by Yan *et al* [20] used a learning-based semantic segmentation model to recognize and process dynamic objects, and then exploited bilinear interpolation to update the motion model of the objects. Notably, the method also drew on an adaptive weight optimization technique to balance the role of geometric and semantic information in SLAM.

Furthermore, as deep learning can achieve high-precision segmentation and detection, it provides a new perspective for loop closure detection. He *et al* [21] used FLCNN networks to extract image features and combined them with similarity matrices to improve the accuracy and effectiveness of closed-loop detection. Xia *et al* [22] chose the PCANet [23] deep learning model to extract features from images, and the results illustrated that the method was more effective than traditional loop closure detection methods. Guo *et al* [24] accelerated the loop closure detection by constructing an image feature vector set based on convolutional neural network VGG-19 [25] and combining the local sensitive hash algorithm for loop closure detection. For their method, Yu *et al* [26] further employed the improved triple constraint Loss function to train darknet to extract image features and constructed feature vectors. The method possessed high accuracy when deeper network layers were available. Subsequently, Wang *et al* [27] leveraged local feature matching based on SIFT and CNN features, and then obtained more comprehensive and accurate closed loop detection results through the construction of topological maps and the fusion of semantic features.

Through the above analysis, since the advantage of high detection accuracy, deep neural networks provide a new way to the SLAM in dynamic scenes. Accordingly, this paper adopts a lightweight deep network YOLOv7 to recognize objects in dynamic scenes and combines it with an improved FFA to remove feature points from dynamic objects, and further retain static feature points for robot's pose estimation. In order to reduce the influence of dynamic objects and robot's movement on keyframe selection, we construct a high-performance keyframe selection strategy using the semantic information obtained from YOLOv7 and combine the semantic information in the image and the translation and rotation of the camera to judge the interval of inserting keyframes. Moreover, to make full use of the semantic information, we also introduce it into the bag-of-words (BOW) model to achieve high-precision loop closure detection. Finally, by fully integrating high-quality keyframes, robot's poses and semantic information to construct a 3D dense semantic map, based on which a 3D semantic octree map for robot navigation can be further generated.

The main contributions of this paper are as follows:

1) A dynamic feature detection and elimination approach based on YOLOv7 combined with FFA is proposed, which can quickly and accurately obtain a high-confidence static feature point set, thereby improving the accuracy of robot's pose estimation in dynamic scenes.
2) A high-performance keyframe selection strategy is developed that combines the semantic information provided by YOLOv7, the motion amplitude of the robot, and the number of dynamic feature points in the camera's field-of-view. In this way, a robust loop closure detection approach is constructed by introducing semantic information into the BOW model and combining it with the

extracted keyframes, improving the effect of back-end optimization and the consistency of mapping.

3) By using YOLOv7 to perform semantic segmentation on the extracted keyframes, the dynamic objects in their semantic masks are removed, based on which the semantic information of static objects and octree map are used to generate 3D semantic navigation maps.

The rest of the paper is organized as follows. The system framework, improvement strategies and implementation process of our algorithm are detailed in section 2. The typical experimental results and comparative analyzes are presented in section 3, which include a series of simulation experiments under dataset and a case study with mobile robot in real scene. Section 4 summarizes the work of this paper and outlines future research directions.

2. Framework and methods

In our work, we adopt a graph optimization-based VSLAM algorithm architecture, it mainly includes two parts: front-end and back-end. The front-end initially extracts ORB [28] feature points from the input RGB image sequence. Then, YOLOv7 [29] is used to perform object detection on the image, and combined with FFA to generate dynamic object masks for the detected dynamic regions in the depth image. This method can effectively separate dynamic and static feature points, and then use static feature point set with a high-confidence to complete robot pose estimation. In order to improve the accuracy and real-time performance of the system, a high-performance keyframe selection strategy is developed by combining the semantic information obtained from YOLOv7 and the robot's rotation and translation vectors. The back-end introduces semantic features into the BOW to construct a loop closure detection method, which calculates similarity based on semantic information, and thus improving the capability of robot's scene understanding and making loop closure detection more accurate. Finally, according to the calculated keyframe poses and robot's trajectory, and coupling with semantic information, further introduce point cloud library (PCL) and point cloud pre-processing technologies to build a 3D dense semantic map that can intuitively reflect the scene information and a 3D semantic octree map for the application of navigation. Scheme of YOLOv7-enhanced-semantic 3D SLAM for mobile robots in dynamic scenes is shown in figure 1.

2.1. Dynamic feature detection and elimination

To ensure good real-time performance of our algorithm, we chose YOLOv7 as the object detection network for feature extraction in the front-end. Compared to the dual-stage networks such as Faster R-CNN [30] and Mask R-CNN, the single-stage YOLO model possesses significant advantages in both training and recognition speed. Likewise, YOLOv7 has higher accuracy than YOLOv5 at the same scale, with a 120% faster speed (FPS). Additionally, within the range of 5 FPS to 160 FPS, both the accuracy and speed of YOLOv7 have surpassed that of the existing detectors. The semantic output by YOLOv7 mainly involves the properties of the predicted bounding boxes, including the center coordinates, width, height, and confidence. By filtering the predicted bounding boxes with a confidence level below 60, the bounding boxes that meet the requirements are transferred to the SLAM system for the next step of operation. Hence, we define the semantic information S_Y of YOLOv7 as the following 6-tuple:

$$S_Y = \{N_Y, C, b_w, b_h, b_x, b_y\} \tag{1}$$

where N_Y denotes the class of target detected, in this paper, YOLOv7 supports identifying 79 different objects, i.e. $N_Y = 79$; C denotes the confidence of the target, C∈[0, 1]; b_w and b_h denote the width and height of the bounding box, respectively; b_x and b_y denote the center coordinate of the target bounding box.

After obtaining the detection results, in the event of removing all point features within the bounding box of a dynamic target, it would lead to excessive removal of point features, resulting in loss of image information and subsequently impacting the accuracy of pose estimation. Thus, in order to improve the accuracy of pose estimation after removing dynamic targets, this paper introduces the FFA to process the depth images to achieve the accurate separation of the moving target contours and backgrounds. Specifically, YOLOv7 is employed to obtain prediction information for each target in the image, and then combined with depth images, the FFA is used to separate the foreground and background within the predicted bounding box, thereby obtaining a dynamic object mask image. When performing feature point extraction, the feature points located within the mask are determined as dynamic feature points and removed to achieve accurate separation of dynamic and static feature points. Figure 2 illustrates the scenarios of the dynamic feature point elimination.

In general, there are three filling ways for the FFA, that is, 4-neighborhood, 8-neighborhood and scan-line based method. Among them, the complexity of 8-neighborhood filling is higher than that of 4-neighborhood, more points need to be considered, and the filling boundaries are unsatisfactory; scan-line filling may be affected by its direction, and in some cases it may require multiple fills. In order to take full account of the real-time and accuracy of our SLAM algorithm, the 4-neighborhood filling algorithm is adopted in our work. During the filling process, for a given seedPoint (i.e. the starting point of filling), the color value of that point is first calculated as the benchmark value. Then, for each unknown point in the image other than the seedPoint, the pixel needs to be filled by comparing it with the benchmark value.

In this work, we choose a fixed range of color images to determine filling, i.e. by comparing the three color values (*R, G, B*) of unknown point pixels to determine whether they are within the tolerance range of the three color values corresponding to the seedPoint, and then determining whether the pixel

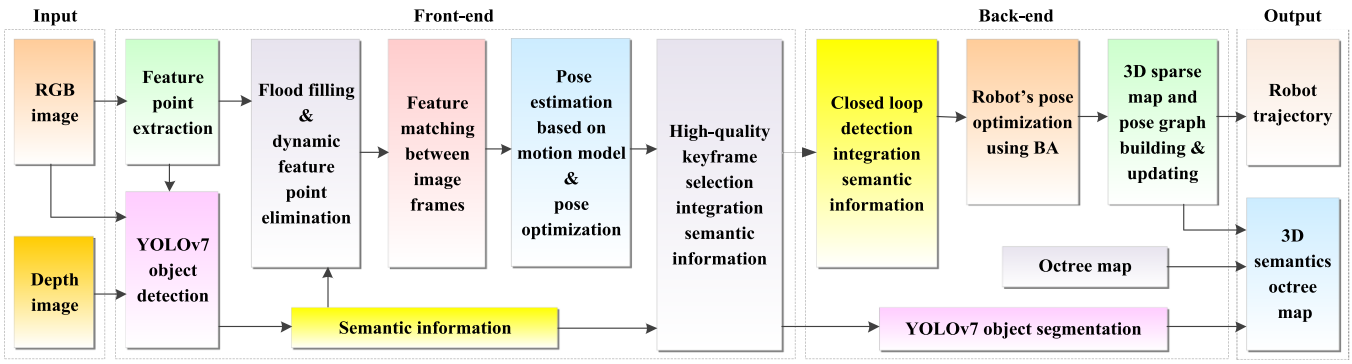


Figure 1. Scheme of YOLOv7-enhanced-semantic 3D SLAM for mobile robots in dynamic scenes.

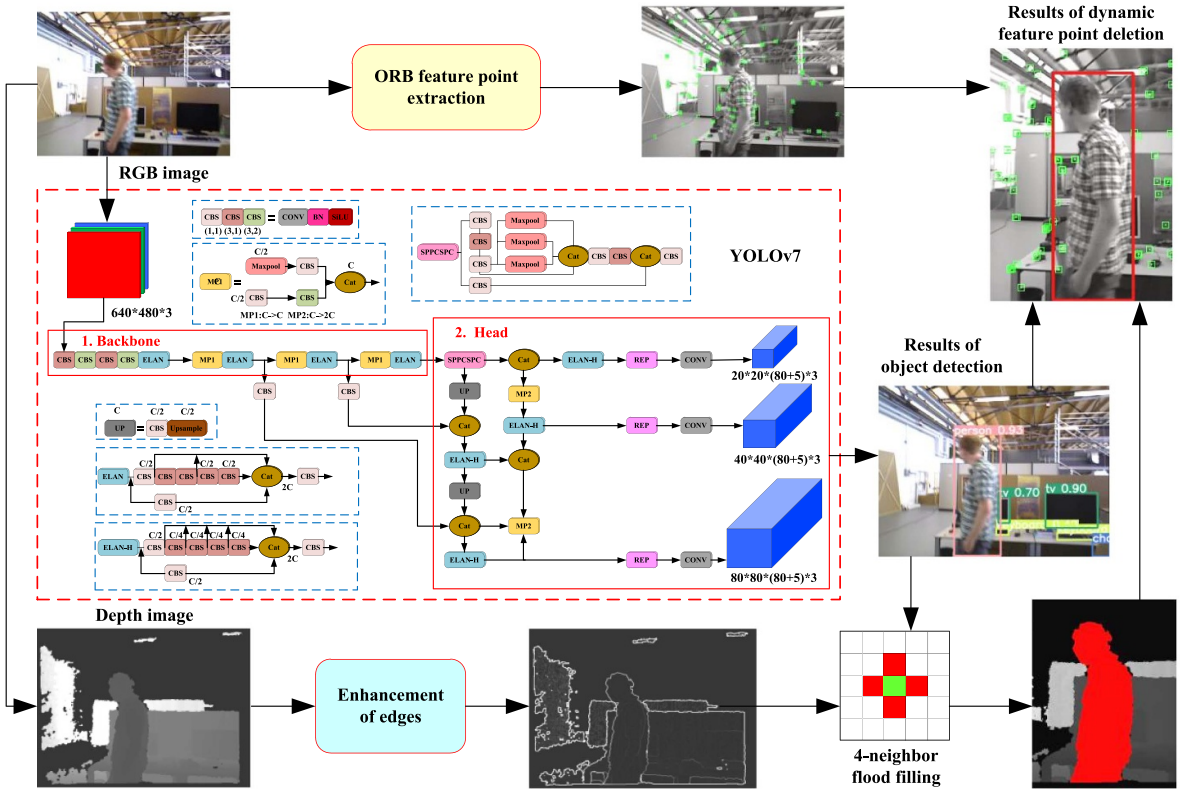


Figure 2. Scenarios for the dynamic feature point elimination.

should be filled. If satisfied, it indicates that the unknown point and seedPoint have similar colors and should be filled in; otherwise, it will be excluded from filling. The filling conditions are as follows:

$$\mathrm{src}(x_s, y_s)_{(R|G|B)} - d_{\mathrm{lo}} \leqslant \mathrm{src}(x, y)_{(R|G|B)} \leqslant \mathrm{src}(x_s, y_s)_{(R|G|B)} + d_{\mathrm{up}} \tag{2}$$

where $(x,\ y)$ denotes the pixel coordinates of unknown points in the image; $(x_s,\ y_s)$ denotes the pixel coordinates of the seedPoint in the image; $\mathrm{src}(x,\ y)$ denotes the color value at coordinates$(x,\ y)$ in the image; d_{lo} and d_{up} denote the difference between the minimum and maximum colors that control filling, respectively. The tolerance range of filling is controlled by adjusting the values of $\mathrm{d_{lo}}$ and d_{up}. A smaller tolerance range can obtain more accurate filling effects, while a larger tolerance range can achieve more relaxed filling, in our case, $d_{\mathrm{lo}} = d_{\mathrm{up}} = 5$.

By filling in the depth image, the accurate regions of dynamic objects are obtained, and then the feature points within the bounding box of the dynamic object are easily distinguished. If the dynamic point is within the filled region, the feature point is considered to be a dynamic object and should be rejected; otherwise, the feature point belongs to a static scene and should be retained, i.e.

$$f(p) = \begin{cases} 1, & p \in F \\ 0, & p \notin F \end{cases} \tag{3}$$

where p denotes the position of the current feature point; F denotes the filling area; and f denotes the discriminant function. If $f(p) = 1$, it indicates that the point is a dynamic feature point; if $f(p) = 0$, it indicates that the point is static.

As a note, in the case of the images with weak edge information, the filling effect of FFA is often undesirable. To this end, an improvement is made in our work for the filling of depth images, namely, combining the Laplacian edge enhancement to optimize the depth map before filling, making the edges of the depth image more prominent. The main idea of the Laplacian algorithm is to find the edge information in an image by applying the Laplace operator to the original image and detecting changes in the image by calculating its second-order derivatives, i.e.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{4}$$

where f denotes the pixel value in the image; $\nabla^2 f$ denotes the result of the Laplacian operator; $\frac{\partial^2 f}{\partial x^2}$ and $\frac{\partial^2 f}{\partial y^2}$ denote the second-order partial derivatives in the x and y directions, respectively.

Then, Gaussian blur is applied to the image to remove the noise in the image, so the 2-dimensional Gaussian filter template is constructed as follows:

$$G(i,j) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{i^2+j^2}{2\sigma^2}\right) \tag{5}$$

where (i, j) denotes the offset between the current element and the center of the template; σ denotes the standard deviation of the Gaussian function. Through the above processing, a better dynamic feature extraction effect can be obtained by utilizing the FFA. Figure 3 compares the filling effects of the standard FFA with our method. It is clear that, our method provides more accurately results by combining semantic and edge-enhanced to extract dynamic targets in the scene.

2.2. Robot's pose estimation

After completed the ORB feature point extraction, it is necessary to match the feature points in the image to calculate the robot's motion and further construct the robot's trajectory. In our case, the Fast Library for Approximate Nearest Neighbors based on Hamming distance is employed for feature matching. The similarity between descriptors is compared using the Hamming distance of two feature points, which in turn accomplishes the matching between neighboring image feature points to estimate the motion of the robot. Herein, the Hamming distance is defined as follows:

$$D(B_1, B_2) = \sum_{i=0}^{n-1} (B_1[i] \oplus B_2[i]) \tag{6}$$

where $\oplus$ denotes the exclusive or operation; B_1 and B_2 denote two different binary strings, i.e., ORB descriptor vectors; i denotes the ith bit of the descriptor vector.

In practical applications, inasmuch as various factors, e.g. image noise, occlusion, etc, there are often some mismatched points in the matching feature, which lead to a great negative impact on subsequent localization and mapping. Thus, we employ the RANSAC algorithm to eliminate these mismatched points. After finished the rejection of dynamic feature points, a static feature points set X and a map points set P with high confidence are obtained. Since the uncertainty of dynamic scenes and the noise of camera observation, the projection position of 3D map points and the position of corresponding feature points on the image are not completely consistent. As a result, the observation error function of the static feature point set X and the map point set P are constructed, and the robot's poses are optimized capitalizing on the following minimized re-projection error, i.e.

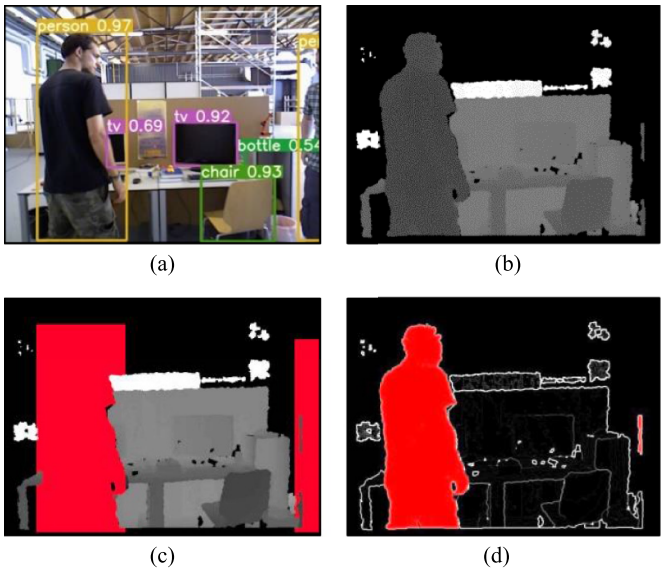


Figure 3. Comparison of the filling results among different algorithms: (a) and (b) are original RGB image and the corresponding depth image, respectively; (c) standard flood filling; (d) combining semantic and edge-enhanced.

$$T_{\mathrm{CW}} = \arg\min \frac{1}{2} \sum_{x_i} \|x_i - \pi(T_{\mathrm{CW}} P_i)\|_2^2 \tag{7}$$

where x_i denotes the ith feature point in the image; P_i denotes the 3D map point corresponding to the feature point x_i; π denotes the projection function; and T_{CW} denotes the robot's pose.

2.3. High-quality keyframe selection

In dynamic scenes, the moving objects often obscure the field-of-vision of the robot's onboard camera. When the dynamic coverings are too close to the robot or their size is too large, the dynamic object will undoubtedly occupy most of the area of the current frame image, leading to incorrect matching between feature points and map points in the current frame, thereby causing the system to insert a large number of redundant keyframes. If we only focus on the redundancy of keyframes and reduce their insertion, when there is a situation that the robot moves too fast or the rotation angle is too large, it will lead to insufficient information obtained by the SLAM system,

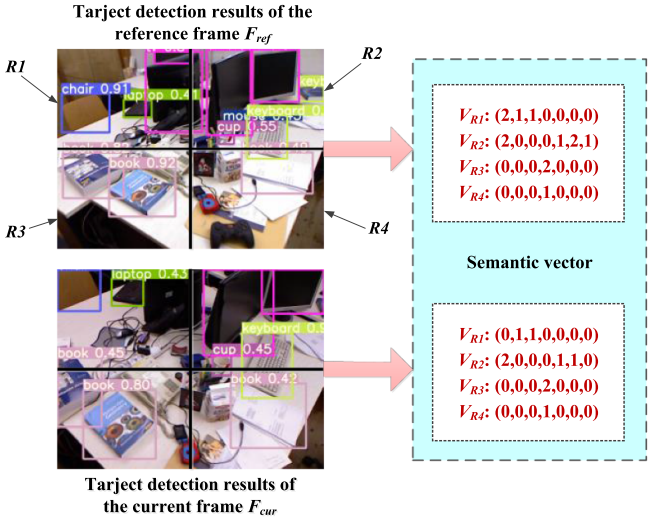


Figure 4. Construction of semantic vector.

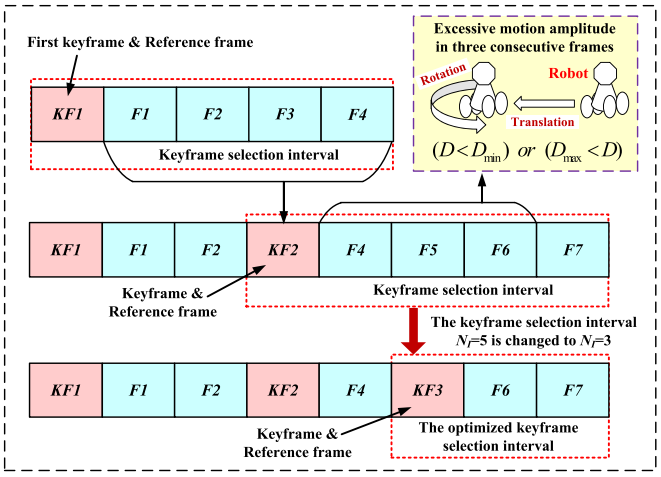


Figure 5. Scenarios for the keyframe selection strategy.

which makes the estimated trajectory have a large error with the real trajectory and even leads to the system tracking failure. Along these lines, this paper proposes a high-quality keyframe selection strategy that combines semantic information and interval division.

First, for the input image sequence $\{F_1, F_2,\ldots, F_N\}$, each image is evenly divided into n regions, and the semantic information S_Y provided by YOLOv7 is employed to determine the objects present in each region. Then, a semantic vector representing each region is generated based on the class and position information of the objects, as shown in figure 4. In this example, the image contains seven classes, and the reference frame F_{ref} and the current frame F_{cur} are evenly divided into four regions *R1, R2, R3, R4*, i.e. $n = 4$. According to S_Y, we can construct the semantic vectors V_{R1}, V_{R2}, V_{R1}, V_{R4} corresponding to each region.

Then, define an area similarity coefficient of the target bounding box as follows:

$$\eta = \frac{A_{\text{u}}}{A_{F_{\text{cur}}} + A_{F_{\text{ref}}}} \tag{8}$$

where $A_{F_{\text{cur}}}$ and $A_{F_{\text{ref}}}$ denote the area of the target bounding box for the current frame and reference frame, respectively; A_{u} denotes the area of the same part of the corresponding target bounding box in these two frames.

Further, construct the Jaccard similarity coefficient for the current frame F_{cur} and reference frame F_{ref} as follows:

$$J(F_{\text{cur}}, F_{\text{ref}}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|V_{Ri}^{F_{\text{cur}}} \cap V_{Ri}^{F_{\text{ref}}}\right|}{\left|V_{Ri}^{F_{\text{cur}}} \cup V_{Ri}^{F_{\text{ref}}}\right|} \tag{9}$$

where $V_{Ri}^{F_{\text{cur}}}$ and $V_{Ri}^{F_{\text{ref}}}$ denote the semantic vectors of the ith region of the current frame and reference frame, respectively. According to equations (8) and (9), the global similarity S_{G} of the current frame and reference frame is constructed as follows:

$$S_{\text{G}} = \gamma J(F_{\text{cur}}, F_{\text{ref}}) + (1-\gamma)\eta \tag{10}$$

where γ is a proportional coefficient.

The keyframe selection strategy combining global similarity and interval division is shown in figure 5. Initially, select a high-quality frame among the first few frames of the image sequence as the first keyframe, and set this frame as the reference frame KF_{ref} for subsequent frames. Considering the computational efficiency, starting from KF_{ref}, we choose every five frames as a keyframe selection interval and calculate the global similarity S_{G} between subsequent input frames and the reference frame KF_{ref}. Assuming that the current frame F_{cur} and reference frame KF_{ref} are divided into n blocks, and the semantic vector within each block is V_{Ri}. Then, the Jaccard similarity coefficient $J(F_{\text{cur}}, KF_{\text{ref}})$ of F_{cur} and KF_{ref} can be written as follows:

$$J(F_{\text{cur}}, KF_{\text{ref}}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|V_{Ri}^{F_{\text{cur}}} \cap V_{Ri}^{KF_{\text{ref}}}\right|}{\left|V_{Ri}^{F_{\text{cur}}} \cup V_{Ri}^{KF_{\text{ref}}}\right|} \tag{11}$$

Because our algorithm performs dynamic feature point rejection at the front-end, if the number of static feature points N_{sp} and the total number of feature points N_{p} in the image are satisfied $N_{\text{sp}} \leqslant 0.5N_{\text{p}}$, it is considered that the dynamic feature points in the current image account for most of the points, and it is not suitable for keyframe, so this frame is skipped when inserting a normal frame in the interval.

If the number of image frames N_{I} in the interval is equal to 5, i.e. $N_{\text{I}} = 5$, the frame with the lowest global similarity S_{G} between these five frames and the reference frame is selected as the candidate keyframe F_{cand}, and it is set as the reference frame KF_{ref} for the next keyframe selection interval. Considering that in the case of poor image quality or simple scene, it is not possible to extract enough internal points for

tracking, the number of inliers in this candidate keyframe has to meet the minimum requirement (i.e. the number of inliers $N_{\text{in}} > 20$). When this candidate keyframe meets the above conditions, it can be inserted into the keyframe sequence as a keyframe and set as the reference frame KF_{ref} for the next keyframe selection interval.

Moreover, when the speed of robot is too fast or the rotation angle is too large, we employ the ORB feature points extracted from the Front-end to calculate the centroid positions p and p' of these two sets of feature points, and calculate the centroid coordinates of each point as follows:

$$\begin{cases} q_{\text{i}} = p_i - p \\ q_i' = p_i' - p' \end{cases}. \tag{12}$$

The objective function is further constructed to calculate the rotation matrix as follows:

$$R = \arg\min \frac{1}{2} \sum_{i=1}^{n} \|q_i - Rq'_i\|^2. \tag{13}$$

Expand the error term as follows:

$$\|q_i - Rq_i'\|^2 = \left(q_i^T q_i + q_i'^T R^T R q' - 2 q_i^T R q_i'\right). \tag{14}$$

Couple with singular value decomposition (SVD) decomposition [31], we can obtain the following:

$$\begin{cases} R = UV^T \\ t = p - Rp' \end{cases} \tag{15}$$

where R and t represent the rotation matrix and translation vector between the current frame and the reference frame, respectively; U and V are diagonal matrix.

According to R and t, the motion amplitude of each ordinary frame in the interval relative to the reference frame can be calculated as follows:

$$D = \|t\| + \min\left(2\pi - \|R\|, \|R\|\right). \tag{16}$$

By setting the upper-bound $D_{\max}$ and lower-bound $D_{\min}$ of the motion amplitude to determine whether the current frame satisfies the following condition:

$$D_{\min} < D < D_{\max}. \tag{17}$$

If the above condition is satisfied, continue to insert image frames into the interval and keep the number of image frames in the interval, i.e. $N_{\text{I}} = 5$. If there are three consecutive frames that do not satisfy the above conditions, it means that the current frame has greater motion amplitude than the reference frame. Meanwhile, the keyframe selection interval is narrowed, and every three frames are regarded as an interval, i.e. $N_I = 3$, and the keyframes continue to be selected within the interval in accordance with the above steps. This processing increases the frequency of keyframe selection, but improves the robustness of robot's pose estimation. The procedure of high-quality keyframe selection is listed in table 1.

Table 1. Pseudo-code of the keyframe selection algorithm.

Algorithm 1. Keyframe selection

Input: $\{F_1, F_2, \ldots, F_N\}$, D, D_{min}, D_{max}, N_p, N_{sp}, the initial number of frames with excessive motion magnitude $i = 0$;
Output: High-quality keyframe KF;

```
while
  if D_min < D < D_max, then
    N_I = 5; //Set keyframe selection interval
  else
    i++; //Count the number of frames with excessive motion magnitude
    if i ⩾ 3
      N_I = 3; //The keyframe selection interval is changed to N_I = 3
      i = 0;
    end if
  end if
  S_min = 1.0; //Define a minimum global similarity S_min
  for k = 1 to N_I do
    if N_sp ⩽ 0.5N_p //If there are too few static feature points
      continue;
    end if
      S_G = γJ(F_k, F_ref) + (1 − γ)η; //Calculate the global similarity
    if (S_min > S_G)
      S_min = S_G; //Update the minimum similarity
      F_cand ← F_k; //Set the k^th frame F_k as a candidate keyframe
    end if
  end for
    F_ref ← F_cand; //Set F_cand as the reference frame for the next interval
  if N_in > 20
    KF = F_cand; //Return F_cand as a keyframe
  end if
  goto step2
```

2.4. Loop closure detection

As the robot continues to move, the errors in its pose and point cloud position gradually accumulate, which cannot be fully corrected through local BA optimization. Thus, loop closure detection needs to be performed based on the covisibility relationship and the similarity between the historical image and the current image. A commonly used method of loop closure detection is to adopt the BOW model, and the visual dictionary can be obtained through pre-training. Using visual words to describe the feature information in an image, the similarity $s(v_{\text{A}}, v_{\text{B}})$ between the BOW vectors v_{A} and v_{B} of two images is calculated to determine whether they are from the same location or not, i.e.

$$s(v_{\text{A}}, v_{\text{B}}) = 2 \sum_{i=1}^{N} \left(|v_{\text{A}i}| + |v_{\text{B}i}| - |v_{\text{A}i} - v_{\text{B}i}|\right) \tag{18}$$

where N is the number of words in the visual dictionary.

In this paper, we introduce the semantic information S_Y provided by YOLOv7 into the traditional BOW model

and construct the following similarity calculation method to improve the accuracy of loop closure detection in dynamic scenes, i.e.

$$S_{\text{Loop}} = \alpha s(v_{\text{A}}, v_{\text{B}}) + (1-\alpha) S_{\text{G}} \tag{19}$$

where S_{Loop} is the similarity for loop closure detection; α is a proportion coefficient; S_{G} is the global semantic similarity calculated by equation (10). It is noticed that, since our algorithm has eliminated dynamic feature points at the front end, and obtained high-quality keyframes for loop closure detection based on the improved keyframe selection strategy, which makes the BOW model exclude the interference of the dynamic feature points on the similarity calculation, and improves the accuracy of the BOW model in loop closure detection in dynamic scenes to a certain extent.

In equation (19), the computation of the global semantic similarity S_{G} relies on the semantic information detected in the scene, so the more objects with semantic information are detected, the more accurate the similarity calculation is. Hence, in order to obtain S_{Loop} with higher confidence, it is necessary to determine the number of objects with semantic information detected in the scene, i.e. N_{sw}. In our case, the proportion coefficient α is set as follows:

$$\alpha = \begin{cases} 0.45, & N_{sw} \geqslant 5 \\ 0.65, & N_{sw} < 5 \end{cases}. \tag{20}$$

The connotation of equation (20) is that, if $N_{sw} \geqslant 5$, it indicates that sufficient semantic information has been obtained, thereby the confidence of S_{G} in S_{Loop} is higher than that of $s(v_{\text{A}}, v_{\text{B}})$, and vice versa.

The full scheme of loop closure detection combined with semantic similarity is illustrated in figure 6. In terms of the similarity S_{Loop} and the number of words in the BOW model, combined with the covisibility relationship of the keyframes (i.e. a covisibility keyframe is determined by the number of covisibility map points $N_{\text{cov}} > 15$), a loop closure candidate keyframe KF_{cand} that is not covisibility with the current keyframe KF_{cur} can be obtained. Then, all covisibility keyframes $\{KF^1_{\text{cov}}, KF^2_{\text{cov}}, \ldots, KF^n_{\text{cov}}\}$ of KF_{cand} together with KF_{cand} are combined to form a group of closed loop candidate keyframes by using the covisibility relationship and the similarity S_{Loop}, i.e.

$$\text{Group}(KF_{\text{cand}}) = \{KF^1_{\text{cov}}, KF^2_{\text{cov}}, \ldots, KF^n_{\text{cov}}\} \cup KF_{\text{cand}} \tag{21}$$

Furthermore, determine the continuous relationship between $\text{Group}(KF_{\text{cand}})$ and the candidate keyframe group $\text{Group}'(KF_{\text{cand}})$ of the previous loop closure detection, i.e.

$$\text{Group}(KF_{\text{cand}}) \cap \text{Group}'(KF_{\text{cand}}) \neq \Phi. \tag{22}$$

If equation (22) is satisfied, the consecutive times of loop closure candidate keyframe group $N_{\text{c}} = N_{\text{c}} + 1$. In our case, when the consecutive times of any set of candidate keyframes satisfies $N_{\text{c}} \geqslant 3$, it is considered that a loop closure candidate keyframe has been detected, that is, select the candidate keyframe group $\text{Group}^*(KF_{\text{cand}})$ with consecutive times $N_{\text{c}} \geqslant 3$, and the KF_{cand} constituting $\text{Group}^*(KF_{\text{cand}})$ is elected as a candidate loop closure keyframe LKF_{cand}. On this basis, as long as the number of map points obtained by re-projection matching of KF_{cur} with a LKF_{cand} that undergoes similarity transformation satisfies the requirement of closed loop, it is considered that a closed loop has been detected and the loop closure keyframe is marked, which will be used in subsequent optimizations.

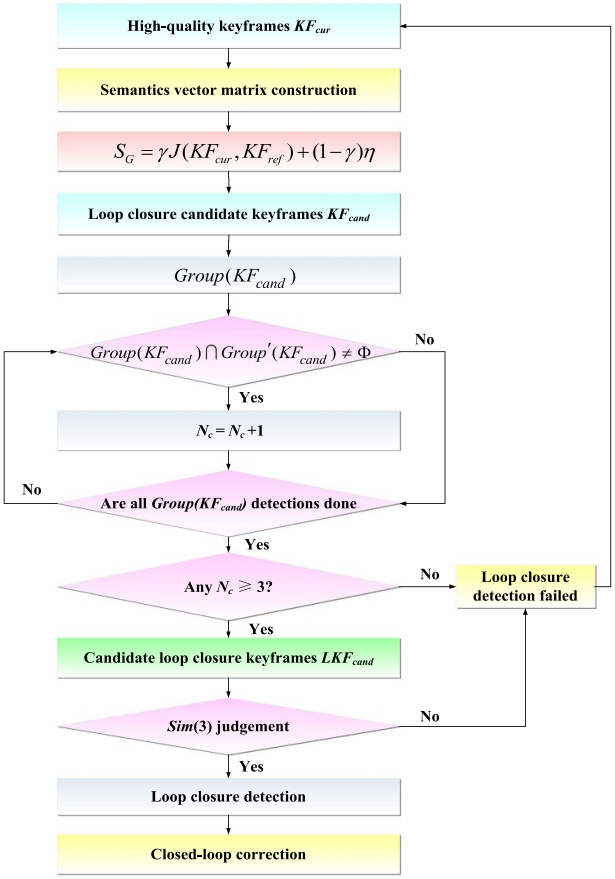


Figure 6. Scenarios for the loop closure detection combined with semantic similarity.

According to the matching relationship between the map point $P = \{p_1, p_2, \ldots, p_n\}$ of current keyframe and the map point $Q = \{q_1, q_2, \ldots, q_n\}$ of closed loop keyframe, the similarity transformation relationship is written as follows:

$$q_i = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix} \cdot p_i \tag{23}$$

where R is the rotation matrix; t is the translation matrix; and s is a scale factor.

In what follows, equation (23) will be added as a loop closure constraint to the back-end optimization framework, which together with other constraints perform a global BA

optimization on the pose of all keyframes and the position of map points to obtain accurate positioning results.

2.5. Robot's pose global optimization

In our work, we take the pose of all keyframes and the position of map points as optimization parameters, and for all the observations of each 3D point in the corresponding keyframe, an edge connecting the corresponding vertex is generated. Through these vertices and edges, the re-projection error function can be defined and optimized using the g2o library. Assume that z_{ij} is the observation of landmark p_j at the robot's pose T_i, the overall cost function can be constructed as follows:

$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\|e_{ij}\|^2 = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\|z_{ij} - h(T_i, p_j)\|^2 \tag{24}$$

where e denotes observation error.

By performing global BA optimization on the equation (24), the robot's pose and 3D map points are adjusted simultaneously, resulting in more accurate results. The optimized results will be further used for the subsequent localization and 3D map construction, improving the accuracy and robustness of the SLAM system.

2.6. 3D semantic map construction

After completing the keyframe selection, a 3D sparse map can be constructed using the keyframe poses and map points tracked by the keyframes. Unfortunately, the sparse map cannot directly represent the environment in which the robot is located in an intuitive way. Thus, on the basis of robot's pose estimation and high-quality keyframe selection, this paper introduces the PCL and semantic information provided by object detection for semantic segmentation, constructing a 3D dense semantic map that can help robots perceive the environment, improving the robot's autonomous navigation ability.

First, YOLOv7 is employed to perform semantic segmentation on the high-quality keyframes extracted from the front-end, accurately and quickly recognizing the target semantic information in the scene and generating the semantic segmentation mask map, and marking the semantic information in the keyframes, so obtaining the 2-dimensional semantically labeled images of the keyframes. This semantic information helps the robot to perceive and understand different features and objects of the surrounding environment in a more detailed way. In order to fuse the keyframe images with different object type labels after semantic segmentation into the 3D point cloud map, based on identifying the moving target in the scene, the target is removed according to the segmentation mask image, and a single keyframe point cloud is extracted through spatial coordinate alignment. The mapping of the keyframe's 2D image to the 3D point cloud is constructed, achieving the expression of 2D semantic information in the 3D point cloud, as shown in figure 7. Specifically, using the robot's pose and camera internal parameter, combined with the depth

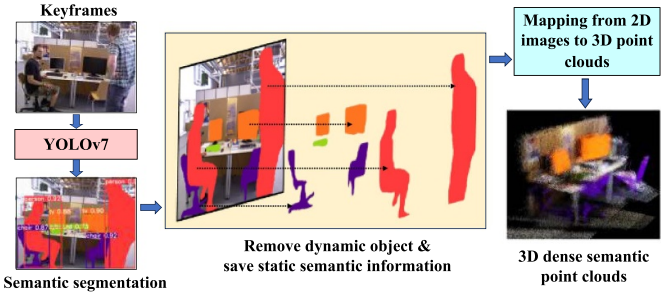


Figure 7. Dynamic object elimination and 3D dense semantic point cloud generation in keyframes.

information obtained by the RGB-D camera, convert each 2D point in the image into 3D coordinates, i.e.

$$\begin{bmatrix} X_{x,j} \\ Y_{i,j} \\ Z_{i,j} \end{bmatrix} = \begin{bmatrix} \frac{(u-C_x)Z_{i,j}}{f_x} \\ \frac{(u-C_y)Z_{i,j}}{f_y} \\ \frac{d}{s} \end{bmatrix} \tag{25}$$

where d is the distance from the camera to the object; s is a scale factor; f_x and f_y are the camera's internal parameters; u and v are pixel coordinates; C_x and C_y are the offsets of the image origin points.

Further, the pose matrix is used to transform the point cloud of each keyframe into the world coordinate system for splicing. Assuming that the transformed point cloud coordinate is $(X'_{x,j}, Y'_{i,j}, Z'_{i,j})$, the following hold:

$$\begin{bmatrix} X'_{x,j} \\ Y'_{i,j} \\ Z'_{i,j} \\ 1 \end{bmatrix} = T_i \begin{bmatrix} X_{x,j} \\ Y_{i,j} \\ Z_{i,j} \\ 1 \end{bmatrix}. \tag{26}$$

After acquiring the single frame point cloud, the point cloud corresponding to each keyframe is converted to the world coordinate system, and the global point cloud map can be obtained by splicing the point cloud using the pose of optimized keyframe. Since the uncertainty of sensors and the environment, 2D semantic segmentation of a single frame image may result in inconsistent labels between consecutive frames. Thus, we adopt Bayesian rule to obtain probability tags from multiple keyframes, namely, assuming that at time t, the class of a 3D point Q is represented as L_t, and all pixel measurements related to this point can be denoted as $x_L^t = \{x_0, x_1, \ldots, x_t\}$. Then, the label probability can be obtained according to the Bayesian rule, i.e.

$$p(L_t|x_0^t) = \frac{1}{Z_t} p\left(x_t|x_0^{t-1}, L_t\right) p\left(L_t|x_0^{t-1}\right) \tag{27}$$

where Z_t is a normalized coefficient. According to Markov hypothesis and the smoothness of the posterior hypothesis, equation (27) can be transformed into:

$$p(L_t|x_0^t) = \frac{1}{Z_t} \frac{p(L_t|x_t)\, p(x_t)}{p(L_t)} p\left(L_{t-1}|x_0^{t-1}\right). \tag{28}$$

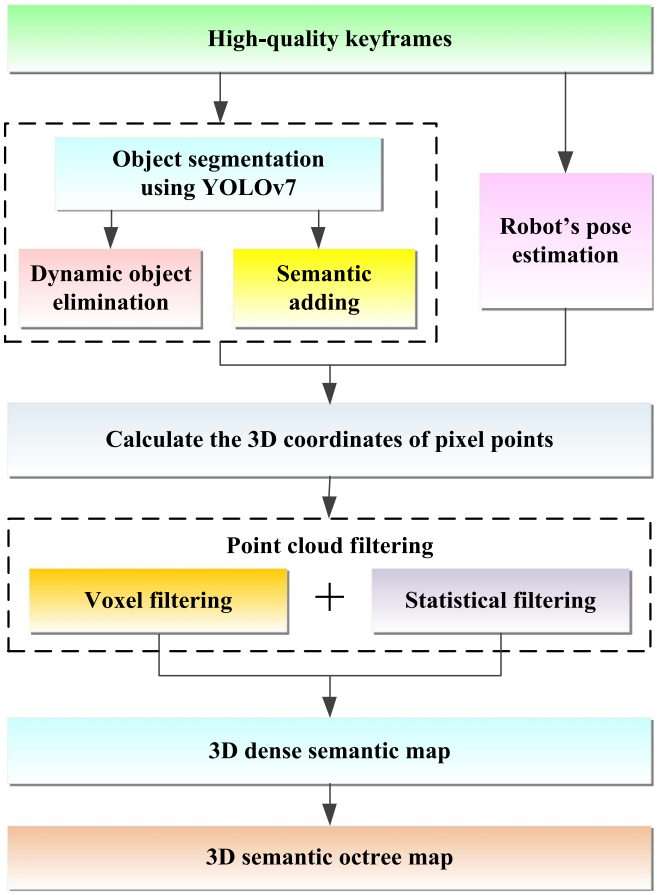


Figure 8. Scenarios for the 3D semantic map construction.

Because of the prior probability $p(x_t)$ is fixed, one can denote the posterior probability by $\hat{p}(L_t|x_t)$. Thus, the update rule for 3D point cloud semantic labels can be expressed as follows:

$$p\left(L_t|x_0^t\right) \leftarrow \hat{p}\left(L_t|x_t\right) p\left(L_{t-1}|x_0^{t-1}\right). \tag{29}$$

In addition, some outliers may appear in the point cloud owing to the presence of external interference. Hence, we adopt voxel filtering and statistical filtering [32] to pre-process the 3D point cloud by down-sampling and outliers filtering to obtain a 3D semantic dense map. Finally, we employ the octree map [33] to transform the 3D semantic point cloud map obtained by our algorithm into a 3D semantic octree map that can be used for navigation. The semantic map construction process is shown in figure 8.

3. Simulation experiments and results analysis

The simulation studies and a case study with mobile robot for the proposed algorithm are given in this section. All of the experiments were performed on a mobile computer with an Intel i7-4712mq CPU with 16 GB of DDR3 RAM, and NVIDIA GTX960 m GPU, running under Ubuntu 18.04 operating system.

3.1. Simulations with TUM dataset

To verify the feasibility and effectiveness of the proposed algorithm, a comparative analysis of different algorithms was conducted using the TUM dataset [34], which is a benchmark dataset for VSLAM created by the Technical University of Munich in Germany, and contains RGB-D image sequences and robot motion trajectories in different scenes.

In this study, since the *walking* series in the TUM dataset can well reflect the influence caused by dynamic objects on the SLAM system in dynamic scenes, the *walking* series dataset was chosen to verify the performance of our algorithm, which mainly includes *walking_xyz, walking_ halfsphere, walking_rpy, walking_ static*. Meanwhile, we adopted absolute trajectory error (ATE) and relative positional error (RPE) as the main indicators to evaluate the accuracy of different algorithms. The connotation of ATE is the direct difference between the estimated pose and the actual pose, it can intuitively reflect the accuracy and global consistency of the algorithm. The RPE indicator mainly describes the accuracy of the pose difference between two frames with a fixed time difference (compared to the true pose), which is equivalent to directly measuring the error of the odometer. Moreover, *sitting_ static* was also chosen, it covers different types of camera motion along *xyz* axis, hemispherical trajectory, yaw-pitch-roll and stationary.

Figure 9 illustrates the dynamic feature point eliminate effects of our algorithm compared to ORB-SLAM2 under the *walking_xyz* dataset. From figures 9(a) and (b), it can be observed that, the ORB-SLAM2 retained feature points on the pedestrians, resulting in a significant error in the estimated trajectory compared to the true trajectory shown in figures 9(c) and (d). Likewise, each frame of the image was severely interfered with when matching with map points, and many redundant keyframes were inserted, resulting in a large number of redundant incorrect map points in sparse maps. As can be seen from figures 9(e) and (f), our method has eliminated the feature points on the dynamic target, also, the keyframes and feature points used in constructing the sparse map have been optimized, so the map points of the dynamic target did not appear in the sparse maps in figures 9(g) and (h), and the estimated trajectory was more consistent with the real trajectory.

Figure 10 describes the ATE and RPE of ORB-SLAM2 and our algorithm utilizing the *walking_xyz* and *walking_ halfsphere* image sequences as inputs, where, the red part represents the error between the estimated and true trajectories. Compared with ORB-SLAM2, it can be seen from figures 10(a)–(h) that, the camera's trajectory estimated by our algorithm was more consistent with the true trajectory. This is because our method can effectively eliminate dynamic feature points, and thus improve the localization accuracy of the robot in dynamic scenes and reduce the impact of dynamic objects on the relative pose transformation of the camera during continuous tracking.

In order to comprehensively validate the accuracy of the proposed method, six experiments were carried out under different sequences in TUM dataset using ORB-SLAM2,

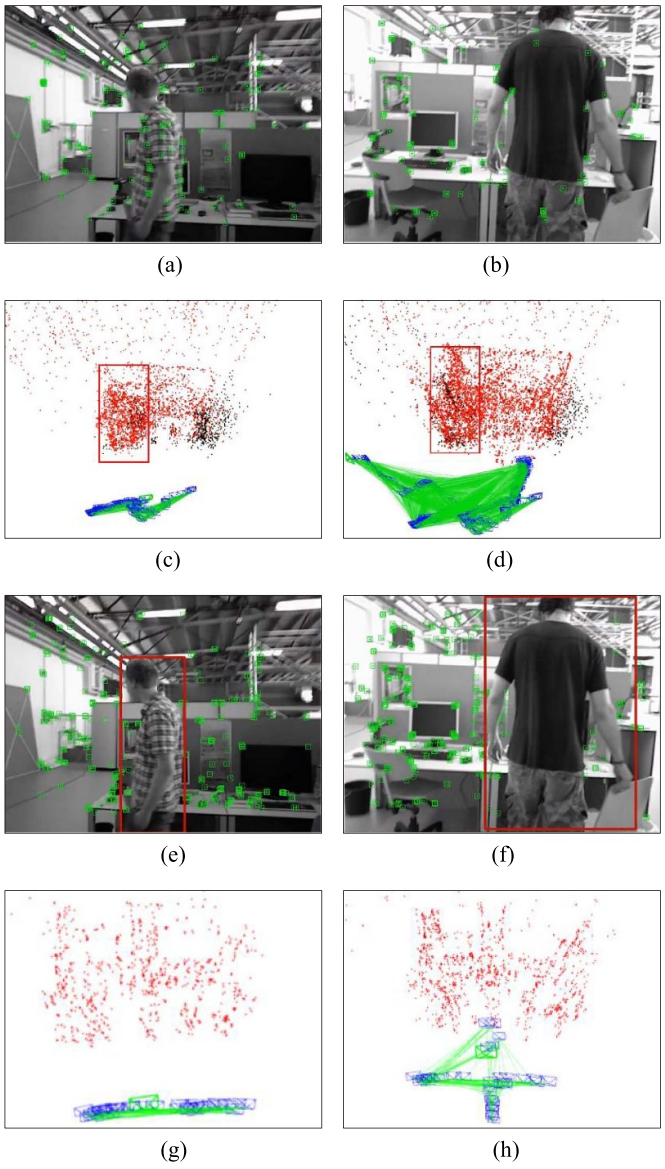


Figure 9. Dynamic feature points removal effect of different algorithms under *walking_xyz* dataset: (a) to (d) ORB-SLAM2; (e) to (h) ours.

DS-SLAM, DynaSLAM, RDS-SLAM [35], RTS-SLAM and our method. In the simulations, the quantitative comparison results of the six algorithms under different sequences are reported in tables 2–4. It can be concluded from table 2 that, as compared to the ORB-SLAM2, the RSME of the proposed algorithm was reduced by 96.8% in high dynamic scenes and 21.8% in low dynamic scenes, whereas, decreased by 34.9% in static scenes. This is because the combination of semantic information generated by YOLOv7 and the flood filling can accurately identify dynamic feature points, which in turn improved the accuracy of the pose estimation. As a result, ATE was generally reduced by one to two orders of magnitude, basically reaching centimeter or millimeter level. Compared with DS-SLAM, we find that, the RSME of our algorithm was reduced by 38.2% in the high dynamic scenes, especially in the *fr3_walking_rpy* dataset, which was reduced by 94.1%. Inasmuch as DS-SLAM has higher accuracy in low dynamic scenes, the accuracy improvement of DS-SLAM was more obvious than our algorithm under the *fr3_sitting_static* dataset. Likewise, the performance in terms of RMSE of our method was slightly inferior to that of DynaSLAM under the *fr3_sitting_static* dataset, but it outperformed DynaSLAM in all the remaining datasets. Furthermore, the results also show that, the accuracy of our algorithm has been improved over that of RDS-SLAM and RTS-SLAM. This is because the fact that, the target detection algorithm used in this paper was highly adaptable to the scene, especially in scenarios where the geometric texture was weak, it can still accurately recognize dynamic objects and eliminate them.

From tables 3 and 4, it is clear that in high dynamic scenes, the RMSEs of the relative translation error.(RTE) and relative rotation error (RRE) of our algorithm have decreased by 93.8% and 91.1% compared to ORB-SLAM2, and by 36.9% and 26.2% compared to DS-SLAM, respectively. Also, there was a significant improvement compared with DynaSLAM, RDS-SLAM, and RTS-SLAM. This is because the main advantage of our method lies in its ability to effectively eliminate dynamic objects, making the calculation of translation and rotation more accurate, significantly improving the stability of the SLAM system in high dynamic scenes and situations with large robot motion amplitudes. Moreover, our algorithm combined semantic information to develop a high-quality keyframe selection and loop closure detection strategies, reducing the overall error of the system, resulting in better RMSEs for RTE and RRE compared to the three mentioned SLAM systems that also used semantic networks to remove dynamic objects. In low dynamic scenes, the RMSEs of RTE and RRE for our method were not much different compared to ORB-SLAM2, decreasing by 10.9% and 6.8%, respectively. Compared with DS-SLAM, the RMSE of RTE increased by 1.3%, while the RMSE of RRE decreased by 2.6%. We also find that, the RTE and RRE of our algorithm were superior to DynaSLAM, RDS-SLAM, and RTS-SLAM. The main reason is that our method mainly focused on high dynamic scenes, while the results provided by ORB-SLAM2 and DS-SLAM were already satisfactory in low dynamic scenes. Notably, DS-SLAM adopted a combination of motion consistency check and segmentation threads to improve the overall performance of the SLAM system. Thus, we can conclude that, our algorithm performed better in high dynamic scenes compared to the other tested methods, but the performance improvement in RTE and RRE in static and low dynamic scenes was not very significant.

Table 5 compares the average time consumption tracking each frame between our algorithm and other mentioned algorithms. It can be concluded from the results that, due to our algorithm added the steps of dynamic point detection and rejection at the front-end, it took longer time in tracking compared to ORB-SLAM2, but the system ran stably and met the requirements for real-time performance. It should be

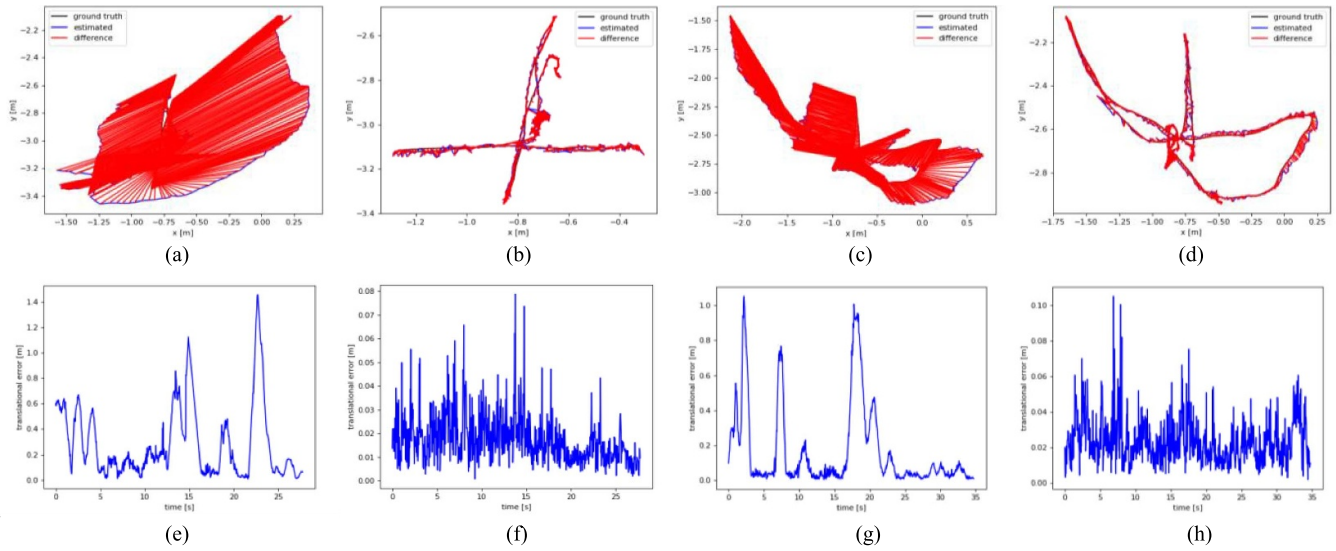


Figure 10. The trajectory error between ORB-SLAM2 and the proposed algorithm; (a) and (c) indicate the ATE of ORB-SLAM2; (b) and (d) indicate the ATE of our algorithm; (e) and (g) indicate the RPE of ORB-SLAM2; (f) and (h) indicate the RPE of our algorithm.

Table 2. Comparison of RMSE for the absolute trajectory error (ATE) among different algorithms.

	Sequences	RMSE(m s^{-1})					
		ORB-SLAM2	DS-SLAM	DynaSLAM	RDS-SLAM	RTS-SLAM	Ours
High dynamic scenes	*fr3_walking_xyz*	0.7521	0.0246	0.0155	0.0571	0.0194	**0.0151**
	fr3_walking_static	0.3900	0.0082	**0.0069**	0.0206	0.0111	0.0080
	fr3_walking_rpy	0.8705	0.4438	0.0378	0.1604	0.0371	**0.0261**
	fr3_walking_half	0.4563	0.0311	0.0257	0.0807	0.0290	**0.0255**
Low dynamic scenes	*fr3_sitting_static*	0.0087	**0.0065**	0.0108	0.0084	—	0.0068
	fr3_sitting_xyz	0.0091	—	0.0159	—	0.0117	**0.0083**
	fr3_sitting_rpy	0.0216	—	—	—	—	**0.0213**
	fr3_sitting_half	0.0254	—	0.0206	—	0.0172	**0.0199**
Static scene	*fr3_long_office_household*	0.0149	—	—	—	—	**0.0097**

Note: Bold indicates that the value of data is the smallest in this line.

Table 3. Comparison of RMSE for the relative translation error (RTE) among different algorithms.

	Sequences	RMSE(m s^{-1})					
		ORB-SLAM2	DS-SLAM	DynaSLAM	RDS-SLAM	RTS-SLAM	Ours
High dynamic scenes	*fr3_walking_xyz*	0.4124	0.0333	0.0254	0.0426	0.0234	**0.0200**
	fr3_walking_static	0.2162	0.0102	0.0133	0.0221	0.0117	**0.0080**
	fr3_walking_rpy	0.4249	0.1503	0.0415	0.1320	0.0471	**0.0379**
	fr3_walking_half	0.3550	0.0297	0.0394	0.0482	0.0423	**0.0264**
Low dynamic scenes	*fr3_sitting_static*	0.0095	**0.0078**	0.0126	0.0123	—	0.0079
	fr3_sitting_xyz	0.0115	—	0.0208	—	0.0166	**0.0109**
	fr3_sitting_rpy	0.0277	—	—	—	—	**0.0265**
	fr3_sitting_half	0.0240	—	0.0306	—	0.0259	**0.0199**
Static scene	*fr3_long_office_household*	0.0085	—	—	—	—	**0.0081**

Note: Bold indicates that the value of data is the smallest in this line.

noted that, since the dynamic target detection and map building threads took up a large amount of computation time, the use of better hardware devices can improve the processing speed of the algorithm to a certain extent.

To analyze the number of keyframes, we compared the number of keyframes obtained by the ORB-SLAM2 and our method under the TUM series datasets, as illustrated in figure 11. It is clear that the keyframe selection algorithm

Table 4. Comparison of RMSE for the relative rotation error (RRE) among different algorithms.

Sequences		RMSE(°/s)					
		ORB-SLAM2	DS-SLAM	DynaSLAM	RDS-SLAM	RTS-SLAM	Ours
High dynamic scenes	*fr3_walking_xyz*	7.7432	0.8266	0.6252	0.9222	0.6368	**0.6128**
	fr3_walking_static	3.8958	0.2690	0.3000	0.4944	0.2872	**0.2688**
	fr3_walking_rpy	8.0802	3.0042	0.9047	13.1693	1.0587	**0.8498**
	fr3_walking_half	7.3744	0.8142	0.8933	1.8828	0.9650	**0.7558**
Low dynamic scenes	*fr3_sitting_static*	0.2881	0.2735	0.3416	0.3338	—	**0.2665**
	fr3_sitting_xyz	0.4853	—	0.6249	—	0.5968	**0.4759**
	fr3_sitting_rpy	0.8411	—	—	—	—	**0.7091**
	fr3_sitting_half	0.6401	—	0.8423	—	0.7891	**0.6272**
Static scene	*fr3_long_office_household*	0.4590	—	—	—	—	**0.4579**

Note: Bold indicates that the value of data is the smallest in this line.

Table 5. Comparison of the time consumption among different algorithms.

Algorithm	ORB-SLAM2	DS-SLAM	DynaSLAM	RDS-SLAM	RTS-SLAM	Ours
GPU	—	P4000	Nvidia Tesla M40 GPU	GeForce RTX 2080Ti	512-core Volta GPU	**Nvidia GTX960m**
Network	—	SegNet	Mask R-CNN	SegNet/Mask R-CNN	SegNet	**YOLOv7**
Segmentation/ Detection time (ms)	—	37.5733	195	30/200	72.36	**35**
Tracking average time (ms)	21.5778	>65	>300	50–60	75.82	**27.3832**

Note: Bold indicates that the value of data is the smallest in this line.

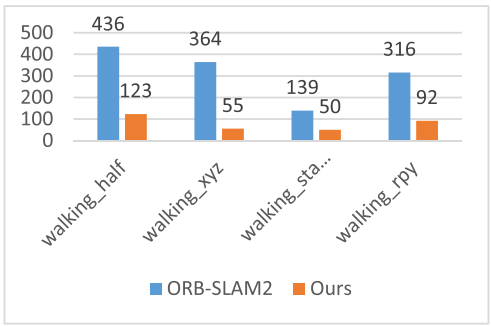


Figure 11. The number of keyframes obtained by different algorithms under TUM series dataset.

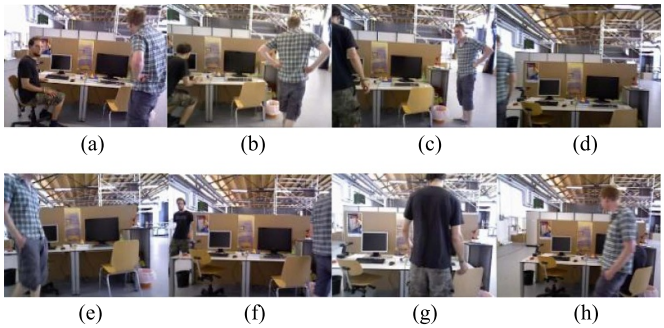


Figure 12. Test images selected from the *r3_walking_xyz* dataset for similarity calculation.

proposed in this paper obtained a relatively small number of keyframes. The main reason is that, the dynamic feature points have been removed in the front-end and only static feature points were retained in the keyframes, so the quality of the obtained keyframes was better than ORB-SLAM2, and the accuracy of the retained feature points was higher when matched with the map points. It should also be noted that, the selected keyframe was the one with the lowest similarity to the previous keyframe in the interval and the most representative one, which effectively reduces the redundancy of keyframes. Accordingly, combined with the ATE and the number of keyframes on the TUM dataset, it can be concluded that, our algorithm reduced the redundancy of keyframes while effectively improving the localization accuracy and stability.

To verify the effectiveness of the loop closure detection algorithm in this paper, eight discontinuous images in the *fr3_walking_xyz* dataset were selected for the similarity comparison, as show in figure 12. In this study, the similarity detection experiments were carried out for our algorithm and the conventional BOW model, the similarity was calculated for the eight images in pairs, with a similarity value range of [0, 1]. When the images were identical, the similarity equal to 1, and when completely different, it was 0. Table 6 lists the calculation results of the similarity between the BOW model and our algorithm, among them, S_B and S_G indicate the similarity calculated by the traditional BOW model and our algorithm, respectively.

In these images, figures 12(d) and (f) are essentially similar, and the interference of moving objects in the scene was small. From table 6, the similarity calculated by the BOW was 0.0400, which was lower than that of our algorithm, 0.0451. Figures 12(g) and (h) are relatively similar in the static area, but are greatly disturbed by moving objects, making the similarity calculated by BOW was lower, and only

Table 6. Comparison of the similarity calculation results between the traditional BOW model and our method.

Figure	Figure 12(a)	Figure 12(b)	Figure 12(c)	Figure 12(d)	Figure 12(e)	Figure 12(f)	Figure 12(g)	Figure 12(h)
Figure 12(a)	$S_G = S_B = 1$	$S_B = 0.0347$	$S_B = 0.0324$	$S_B = 0.0413$	$S_B = 0.0197$	$S_B = 0.0366$	$S_B = 0.0423$	$S_B = 0.0443$
Figure 12(b)	$S_G = 0.0338$	$S_G = S_B = 1$	$S_B = 0.0555$	$S_B = 0.0364$	$S_B = 0.0284$	$S_B = 0.0204$	$S_B = 0.0368$	$S_B = 0.0334$
Figure 12(c)	$S_G = 0.3286$	$S_G = 0.0500$	$S_G = S_B = 1$	$S_B = 0.0346$	$S_B = 0.0207$	$S_B = 0.0389$	$S_B = 0.0232$	$S_B = 0.0311$
Figure 12(d)	$S_G = 0.0413$	$S_G = 0.0376$	$S_G = 0.0349$	$S_G = S_B = 1$	$S_B = 0.0402$	$S_B = 0.0400$	$S_B = 0.0208$	$S_B = 0.0420$
Figure 12(e)	$S_G = 0.0236$	$S_G = 0.0338$	$S_G = 0.0274$	$S_G = 0.0398$	$S_G = S_B = 1$	$S_B = 0.0356$	$S_B = 0.0203$	$S_B = 0.0293$
Figure 12(f)	$S_G = 0.0403$	$S_G = 0.0261$	$S_G = 0.0361$	$S_G = 0.0451$	$S_G = 0.0347$	$S_G = S_B = 1$	$S_B = 0.0363$	$S_B = 0.0441$
Figure 12(g)	$S_G = 0.0363$	$S_G = 0.0362$	$S_G = 0.0298$	$S_G = 0.0310$	$S_G = 0.0277$	$S_G = 0.0348$	$S_G = S_B = 1$	$S_B = 0.0230$
Figure 12(h)	$S_G = 0.0415$	$S_G = 0.0373$	$S_G = 0.0346$	$S_G = 0.0425$	$S_G = 0.0329$	$S_G = 0.0395$	$S_G = 0.0331$	$S_G = S_B = 1$

0.0230, while the similarity of our algorithm was 0.0331, so the similarity credibility of our algorithm was higher. Figures 12(c) and (h) own lower similarity, the similarity of the BOW was 0.0311, while the similarity of our algorithm was 0.0346. This is because the occlusion of moving objects in the scene makes the types of objects detected less, which affects the accuracy of our algorithm. Therefore, in our case, we set a judgment on the number N_{sw} of objects with semantic information in the scene. When N_{sw} was small, the similarity obtained from the BOW had a greater weight in S_{Loop}, thus providing a similarity with higher confidence.

To further illustrate the comprehensive performance of ORB-SLAM2 and the proposed loop closure detection algorithm under the dynamic scene dataset *fr3_walking_xyz*, we adopted the Precision-Recall curve as a measure to visually express the effectiveness of loop closure detection. Among them, the Precision indicates the percentage of real loopbacks among the loop closure results detected by the robot. Recall represents the percentage of true loop closure detected by the robot among all true closed loops. Precision and Recall are respectively defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{30}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{31}$$

where TP is true positive, FP is false positive, and FN is false negative.

As can be seen in figure 13, the accuracy of our algorithm was higher than that of ORB-SLAM2 with the same Recall rate. This is because our algorithm eliminates dynamic feature points in the front-end, reducing the impact of dynamic objects on the similarity calculation of the BOW model, thereby improving the credibility of the similarity calculation. Moreover, since the addition of semantic similarity, the performance of the loop closure detection in our case was more prominent in dynamic scenes. In contrast, if only the BOW detection strategy was used for judgment, dynamic objects will greatly affect the calculation of similarity, leading to mismatches in the BOW detection strategy.

To verify the effects of map building for our algorithm, we exploited the extracted keyframes and the corresponding pose graph, also, combined the point cloud pre-processing technologies to build a dense map and a corresponding octree map of the scene under *fr3 walking_ halfsphere* dataset. From

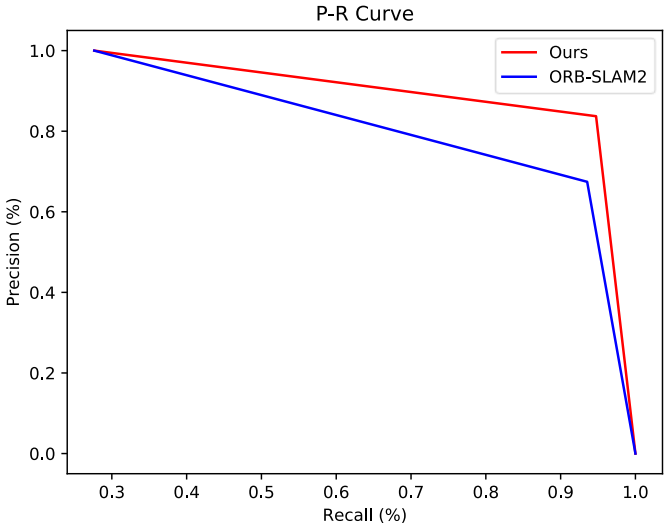


Figure 13. P-R curves of different loop closure detection methods under fr3 *walking_halfsphere* dataset.

figure 14, we can see that the algorithm in this paper combined YOLOv7 semantic segmentation to remove the point cloud of dynamic objects, and the whole map generated was more consistent with the real environment, without distortion and deformation.

3.2. Case study with a mobile robot

In this study, the proposed approach was implemented and tested using a wheeled mobile robot equipped with a ASUS Xtion depth camera. As shown in figure 15(a), the camera consists of three main components: an RGB camera, an infrared structured light emitter, and a receiver. With an effective ranging range between 0.8 m and 3.5 m, the camera can capture RGB image and depth value of objects in the scene and transmit them to the computer in real-time through a USB interface. The experimental scene (figure 15(b)) was a laboratory with an area of 8.4 m $\times$ 6.4 m. The reference trajectory of the robot was indicated by the red dotted line, and had a dimension of 4.8 m $\times$ 1.5 m. During the experiment, a pedestrian with a speed of 0.5 m s^{-1} walked back and forth within the robot's field-of-view to simulate a dynamic scene.

From figures 16(a) and (b), it is evident that, ORB-SLAM2 has extracted many feature points on pedestrians, and combined with the corresponding sparse pose graph, i.e. Figures 16(c) and (d), we observed that the dynamic feature

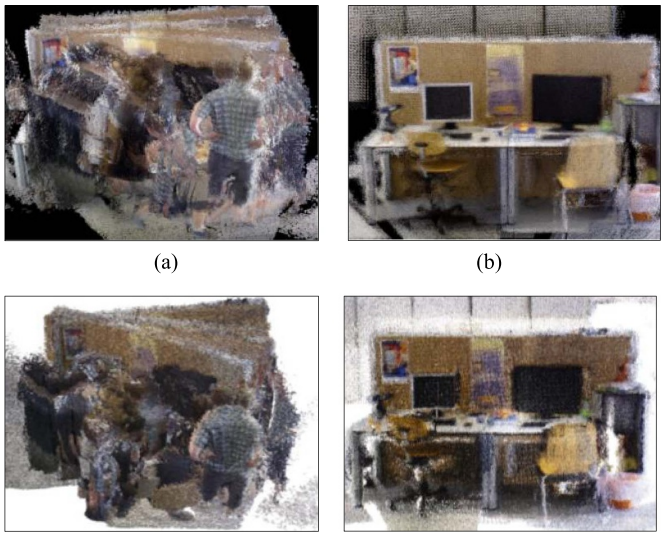


Figure 14. Comparison of map effects constructed by different algorithms under *fr3 walking_ halfsphere* dataset: (a) and (c) are dense map and the corresponding octree map of ORB-SLAM2; (b) and (d) are dense map and the corresponding octree map of ours.

points were added to the sparse point cloud map, and when the pedestrian leaves the field-of-view of the robot, the dynamic feature points in the sparse point cloud were not removed, resulting in poor consistency between the constructed sparse point cloud and the real scene. Consequently, the estimated trajectory deviates significantly from the real trajectory.

It can be concluded from figures 17(a) and (b) that our algorithm effectively eliminated the dynamic feature points and retained static feature points. Coupling with figures 17(c) and (d), we could reasonably to conclude that, since the moving objects had no influence on the pose estimation, obtaining a robot's pose with low ATE and RTE. Thus, the experimental results indicate that our algorithm exhibits good robustness and accuracy in dynamic scene.

To verify the global localization accuracy in real scenes, comparison experiments were conducted between our algorithm and ORB-SLAM2. The trajectories obtained by different algorithms were displayed using the trajectory accuracy evaluation tool EVO developed by Grupp *et al* [36]. Figure 18 compares the estimated trajectories for our algorithm and ORB-SLAM2 in real dynamic scene. It is obvious that, the robot's pose estimated by ORB-SLAM2 was greatly negatively affected when the pedestrian walk within the robot's field-of-view, resulting in a significant error between its output trajectory and reference trajectory. Moreover, due to the influence of pedestrians walking back and forth, when a closed loop was formed in the scene, ORB-SLAM2 was unable to correctly detect the loop. In contrast, for the entire process, the estimated trajectory of our algorithm had a high similarity with the reference trajectory, and successfully detected loop, further improving the accuracy of the robot's trajectory estimation. Figure 19 illustrates similarity of objects within the field-of-view of the robot located in different positions in real scene.

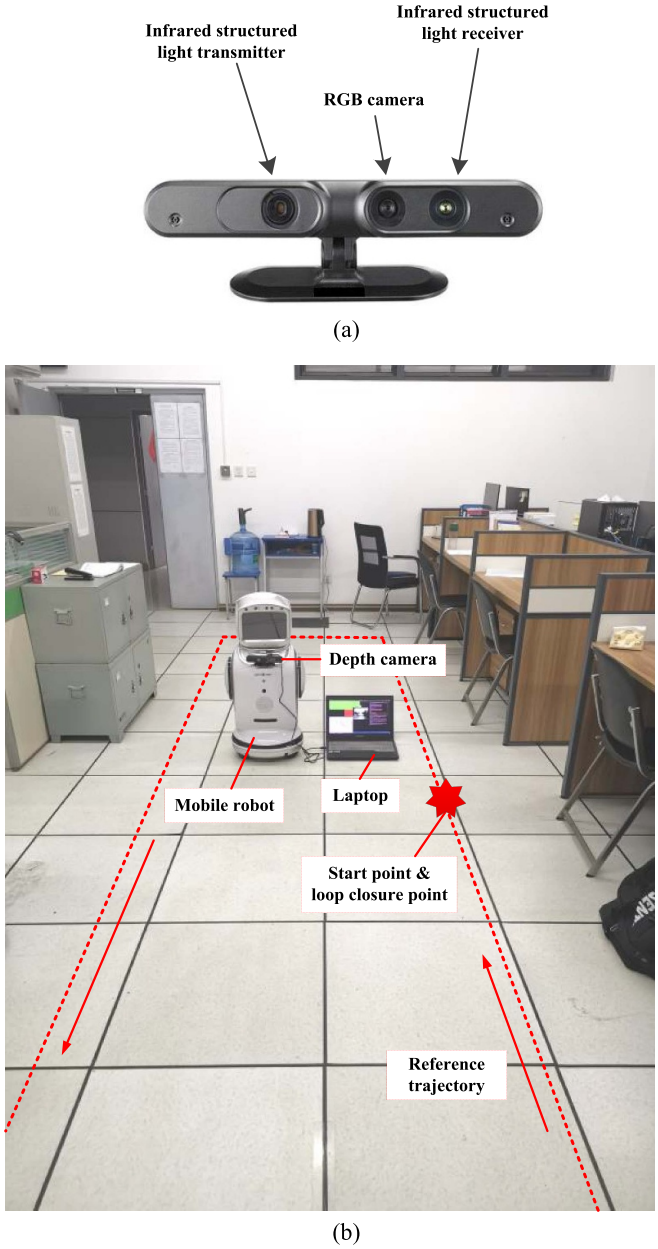


Figure 15. Experimental scene: (a) the ASUS Xtion depth camera used by the mobile robot; (b) the experimental platform of the mobile robot and the reference trajectory.

In the experiment, let the robot moved three circles along the reference trajectory, i.e. the movement process contains three closed loops. The comparison of performance for the two tested detection methods was given in table 7. Clearly, neither of the two algorithms appeared perception deviation, i.e. there was no case where a loop closure was wrongly considered as a loop ($FP = 0$). The traditional BOW model appeared perception variation, i.e. there was a case that the same place was considered to be different ($FN = 1$), but this case did not occur in our algorithm ($FN = 0$). In terms of Recall, the two methods also reached 100%. Nevertheless, in terms of Precision, the traditional BOW model was only 66.7% ($TP = 2$), while

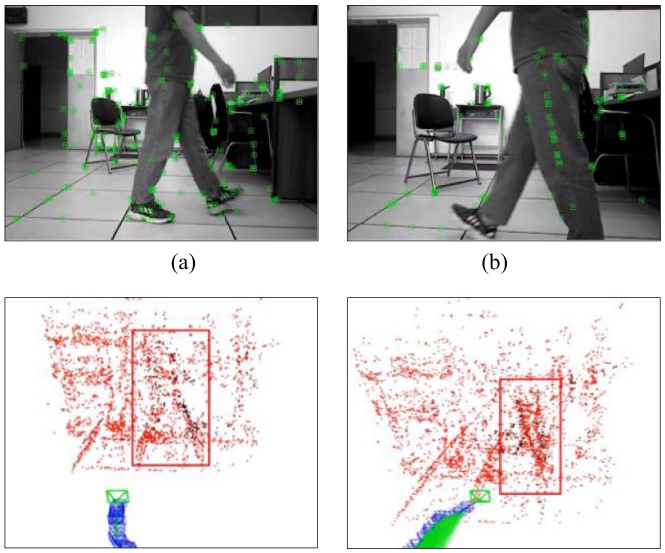


Figure 16. The results of ORB-SLAM2 in real scene: (a) and (b) denote the feature point extraction results; (c) and (d) denote corresponding sparse pose graph.

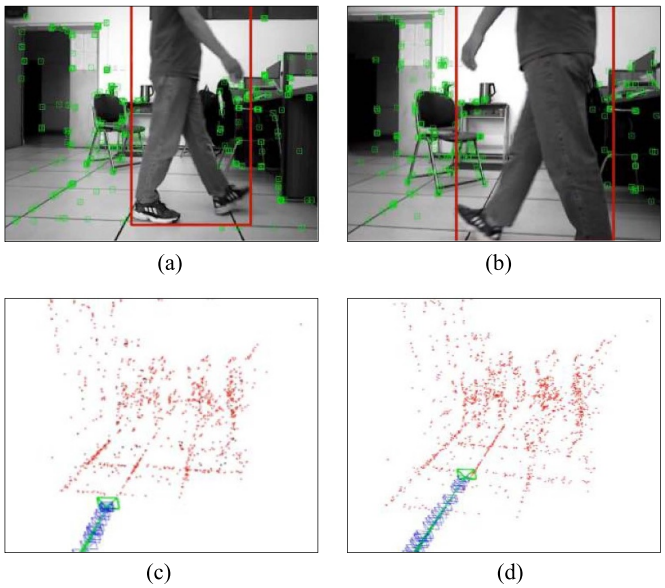


Figure 17. The results of our algorithm in the real scene: (a) and (b) denote the feature point extraction results; (c) and (d) denote corresponding sparse pose graph.

our algorithm reached 100% ($TP = 3$), indicating three loops can be successfully detected.

To verify the effects of octree map constructed by our method, below we conducted experimental analysis for both global map and local map, respectively. It can be seen from figure 20(a) that, although ORB-SLAM2 can complete the final map building, it contained clear deviations. Further, a large number of dynamic features were extracted from the pedestrian in ORB-SLAM2, thereby the entire movement trace of pedestrians appears clearly as an obstacle in the octree constructed by ORB-SLAM2. From figure 20(b), we observed

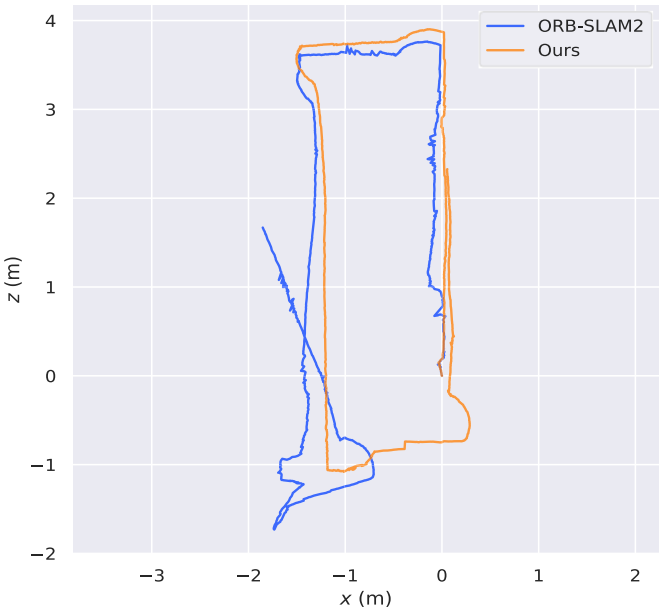
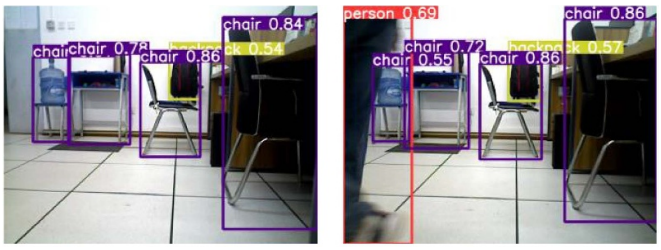


Figure 18. Comparison of the estimated trajectories for two tested algorithms in real dynamic scene.



(a) (b)

Figure 19. Similarity of objects within the field-of-view of the robot located in different positions: (a) starting point; (b) loop closure point.

that, the map constructed by our method was consistent with the real environment, the edge of the map was distinct and there was no overlap or distortion. Also, the semantic information of objects in the map was clearly visible. This is because a main advantage of our approach lies in its ability to effectively detect and remove dynamic feature points.

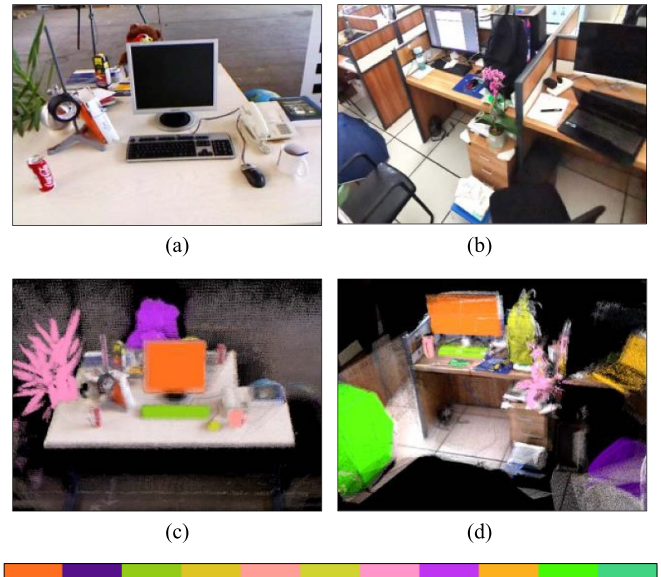
To clearly demonstrate the details of the 3D semantic map constructed by our method, the 3D semantic dense local maps and the corresponding octree maps were constructed for the *freiburg2_xyz* dataset scene (figure 21(a)) and the actual scene (figure 21(b)), respectively. Different colors in the map indicate different semantic objects, and the objects that can be detected in these two scenes include a total of 11 semantic classes, as shown in figure 21(e). From figures 21(c) and (d), it can be seen that the constructed 3D dense semantic map was consistent with the actual scene, the dynamic targets were eliminated, and the objects in the scenes such as keyboards, cups, mouse etc., also the plant with complex structures were correctly identified and segmented for coloring.

Table 7. Comparison of different loop closure detection methods.

Algorithm	Number of closed loop	TP	FP	FN	P	R
BOW	3	2	0	1	66.7%	100%
Semantics + BOW	3	3	0	0	100%	100%

(a) (b)

Figure 20. Comparison of the global semantic octree map for different algorithms in real dynamic scene: (a) ORB-SLAM2; (b) ours.



(a) (b) (c) (d) (e)

Figure 21. 3D semantic dense local maps for different scenes constructed by our method: (a) and (c) are dataset scene and the corresponding 3D semantic dense map, respectively; (b) and (d) are real scene and the corresponding 3D semantic dense map, respectively; (e) semantic class.

Additionally, we converted the 3D semantic dense map into corresponding octree map. As can be seen in figures 22(a) and (b), the features and contours of objects in the octree map were retained intact, such as computers, tables, plant etc., and the position relationship between objects can be clearly determined. Also noteworthy, the file size of the octree map was reduced by about 80% compared to that of the dense map, greatly reducing memory consumption.

(a) (b)

Figure 22. 3D semantic octree local maps for different scenes constructed by our method: (a) dataset scene; (b) real scene.

4. Conclusions and feature works

This paper was concerned of the low localization accuracy and might even tracking loss of SLAM algorithm for mobile robots in dynamic scenes. By adopting graph optimization-based VSLAM algorithm architecture, and combining light-weight deep network YOLOv7 and FFA in the Front-end of the system, the extracted ORB feature points were filtered to quickly and accurately remove dynamic feature points, and the robot's pose was estimated using the obtained static feature point set, improved the accuracy and robustness of the algorithm in dynamic scenes. Meanwhile, a high-performance keyframe selection strategy was constructed by combining the semantic information and pose transformation, which enables our system to reduce keyframe redundancy while improving the accuracy of robot's pose estimation. Moreover, we

introduced semantic information into BOW model to develop an efficient loop closure detection method. This method not merely increased the ability of loop closure detection to understand the scene, but also improved the accuracy of loop closure detection and generated a global consistent map. Finally, a series of simulations and experiments were conducted on public datasets and real scenes, respectively. The results indicated that, the proposed algorithm possessed good localization accuracy and real-time performance in dynamic scenes, also, the constructed map was consistent with actual scene and the program ran smoothly, thereby indicating the feasibility and effectiveness of our method.

As part of future work, we intend to further optimize the association between SLAM and deep networks to improve the real-time and accuracy of system. Another perspective of this research is to combine geometric or optical flow methods to enhance the robustness of the algorithm in weakly textured scenes, and further expand the application scenarios from indoor dynamic scenes to outdoor large-scale complex dynamic scenes. In addition, this paper regards the detected dynamic objects as outlier and does not use them for mapping, which makes the constructed map lack of dynamic object information, and it is not conducive to human-computer interaction and collaborative work. To this end, we also plan to further track and reconstruct dynamic objects based on the recognition and segmentation of dynamic objects through detection networks, and adding the velocity and motion transformation states of dynamic objects to the static scene map in real-time. Indeed, such processing can effectively enhance the robot's perception and adaptability to the environment.

Data availability statement

The data cannot be made publicly available upon publication because they are owned by a third party and the terms of use prevent public distribution. The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (Grant No. 62063036), Research Foundation for Doctor of Yunnan Normal University (Grant No. 01000205020503115) and 'Xingdian Talent Support Program' Youth Talent Special Project of Yunnan Province (Grant No. 01000208019916008)

Funding

National Nature Science Foundation of China (Grant No. 62063036), Research Foundation for Doctor of Yunnan Normal University (Grant No. 01000205020503115) and 'Xingdian Talent Support Program' Youth Talent Special Project of Yunnan Province (Grant No. 01000208019916008)

Ethical statement

This study does not involve ethical and moral issues such as human or animal subjects.

ORCID iD

Jingwen Luo https://orcid.org/0000-0003-3366-6995

References

[1] Fischler M A and Bolles R C 1981 Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography *Commun. ACM* **24** 381–95
[2] Strasdat H, Montiel J M M and Davison A J 2012 Visual SLAM: why filter? *Image Vis. Comput.* **30** 65–77
[3] Zhang C Y, Huang T, Zhang R C and Yi X 2021 PLD-SLAM: a new RGB-D SLAM method with point and line features for indoor dynamic scene *ISPRS Int. J. Geo-Inf.* **10** 163
[4] Howard A G, Zhu M and Chen B 2017 MobileNets: efficient convolutional neural networks for mobile vision applications (https://doi.org/10.48550/arXiv.1704.04861)
[5] Wu Z Y, Deng X Y, Li S M and Li Y 2021 OC-SLAM: steadily tracking and mapping in dynamic environments *Front. Energy Res.* **9** 1–10
[6] Redmon J, Divvala S and Girshick R 2016 You only look once: unified, real-time object detection *Proc. Computer Vision & Pattern Recognition* (https://doi.org/10.1109/CVPR.2016.91)
[7] Ai Y B, Rui T, Lu M, Fu L, Liu S and Wang S 2020 DDL-SLAM: a robust RGB-D SLAM in dynamic environments combined with deep learning *IEEE Access* **8** 162335–42
[8] Mur-Artal R and Tardos J D 2017 ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras *IEEE Trans. Robot.* **33** 1255–62
[9] Jin Q G, Meng Z P, Pham T D, Chen Q, Wei L and Su R 2019 DUNet: a deformable network for retinal vessel segmentation *Knowl.-Based Syst.* **178** 149–62
[10] Wang Z M, Zhang Q, Li J S, Zhang S and Liu J 2019 A computationally efficient semantic SLAM solution for dynamic scenes *Remote Sens.* **11** 1363
[11] Redmon J and Farhadi A 2018 YOLOv3: an incremental improvement (https://doi.org/10.48550/arXiv.1804.02767)
[12] Wu W X, Guo L, Gao H L, You Z, Liu Y and Chen Z 2022 YOLO-SLAM: a semantic SLAM system towards dynamic environment with geometric constraint *Neural Comput. Appl.* **34** 6011–26
[13] Yu C, Liu Z and Liu X 2018 DS-SLAM: a semantic visual SLAM towards dynamic environments *Proc. 2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (https://doi.org/10.1109/iros.2018.8593691)
[14] Bescos B, Facil J M, Civera J and Neira J 2018 DynaSLAM: tracking, mapping, and inpainting in dynamic scenes *IEEE Robot. Autom. Lett.* **3** 4076–83
[15] He K, Gkioxari G, Dollár P and Girshick R Mask R-CNN *Proc. 2017 IEEE Int. Conf. on Computer Vision (ICCV) (Venice, Italy, 22–29 October 2017)* pp 2980–8
[16] Xiao L H, Wang J G, Qiu X S, Rong Z and Zou X 2019 Dynamic-SLAM: semantic monocular visual localization and mapping based on deep learning in dynamic environment *Robot. Autonom. Syst.* **117** 1–16
[17] Liu W, Anguelov D and Erhan D 2016 SSD: single shot multibox detector *Computer Vision—ECCV 2016, Lecture Notes in Computer Science* pp 21–37

[18] Ji T T, Wang C and Xie L H 2021 Towards real-time semantic RGB-D SLAM in dynamic environments *2021 IEEE Int. Conf. on Robotics and Automation (Icra 2021)* pp 11175–81
[19] Badrinarayanan V, Kendall A and Cipolla R 2017 SegNet: a deep convolutional encoder-decoder architecture for image segmentation *IEEE Trans. Pattern Anal. Mach. Intell.* **39** 2481–95
[20] Yan L, Hu X and Zhao L Y 2022 DGS-SLAM: a fast and robust RGBD SLAM in dynamic environments combined by geometric and semantic information *Remote Sens.* **14** 795
[21] Yuanlie H, Jiateng C and Bi Z 2018 Fast loop closure detectionmethod based on reduced convolutional neural network *Comput. Eng.* **44** 6
[22] Xia Y *et al* 2018 An evaluation of deep learning in loop closure detection for visual SLAM *Proc. 2017 IEEE Int. Conf. on Internet of Things (Ithings) and IEEE Green Computing and Communications (Greencom) and IEEE Cyber, Physical and Social Computing (Cpscom) and IEEE Smart Data (Smartdata)* (https://doi.org/10.1109/ithings-greencom-cpscom-smartdata.2017.18)
[23] Chan T-H, Jia K, Gao S, Lu J, Zeng Z and Ma Y 2015 PCANet: a simple deep learning baseline for image classification? *IEEE Trans. Image Process.* **24** 5017–32
[24] Guo J *et al* 2021 Visual SLAM closed-loop detection method based on deep learning *Photon. Laser* **32** 628–36
[25] Simonyan K and Zisserman A 2014 Very deep convolutional networks for large-scale image recognition *Comput. Sci.* 1–14
[26] Yu Y and Hu F 2020 Visual SLAM loop detection method based on deep learning *Comput. Eng. Des.* **41** 8
[27] Wang Y, Qiu Y, Cheng P and Duan X 2020 Robust loop closure detection integrating visual–spatial–semantic information via topological graphs and CNN features *Remote Sens.* **12** 3890
[28] Rublee E *et al* 2011 ORB: an efficient alternative to SIFT or SURF *Proc. IEEE Int. Conf. on Computer Vision, ICCV 2011* (*Barcelona, Spain*, *6–13 November 2011*) (https://doi.org/10.1109/iccv.2011.6126544)
[29] Wang C Y, Bochkovskiy A and Liao H Y M 2022 YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors (https://doi.org/10.48550/arXiv.2207.02696)
[30] Ren S, He K, Girshick R and Sun J 2017 Faster R-CNN: towards real-time object detection with region proposal networks *IEEE Trans. Pattern. Anal. Mach. Intell.* **39** 1137–49
[31] Arun K S 1987 Least-squares fitting of two 3D point sets *IEEE Trans. Pattern Anal. Mach. Intell.* **9** 698–700
[32] Rusu R B, Blodow N and Beetz M 2009 Fast point feature histograms (FPFH) for 3D registration *Proc. IEEE Int. Conf. on Robotics & Automation* (https://doi.org/10.1109/robot.2009.5152473)
[33] Hornung A, Wurm K M, Bennewitz M, Stachniss C and Burgard W 2013 OctoMap: an efficient probabilistic 3D mapping framework based on octrees *Autonom. Robot.* **34** 189–206
[34] Sturm J, Engelhard N and Endres F 2012 A benchmark for the evaluation of RGB-D SLAM systems *2012 IEEE/Rsj Int. Conf. on Intelligent Robots and Systems (Iros)* pp 573–80
[35] Liu Y B and Jun M R 2021 RDS-SLAM: real-time dynamic SLAM using semantic segmentation methods *IEEE Access* **9** 23772–85
[36] Grupp M 2017 EVO: python package for the evaluation of odometry and SLAM[EB/OL] (available at: https://michaelgrupp.github.io/evo/) (Accessed 28 April 2022)