


A real-time semantic visual SLAM approach with points and objects

Peiyu Guan^{1,2} , Zhiqiang Cao^{1,2}, Erkui Chen³, Shuang Liang^{1,2},
Min Tan^{1,2} and Junzhi Yu^{1,4}

Abstract

Visual simultaneously localization and mapping (SLAM) is important for self-localization and environment perception of service robots, where semantic SLAM can provide a more accurate localization result and a map with abundant semantic information. In this article, we propose a real-time PO-SLAM approach with the combination of both point and object measurements. With point–point association in ORB-SLAM2, we also consider point–object association based on object segmentation and object–object association, where the object segmentation is employed by combining object detection with depth histogram. Also, besides the constraint of feature points belonging to an object, a semantic constraint of relative position invariance among objects is introduced. Accordingly, two semantic loss functions with point and object information are designed and added to the bundle adjustment optimization. The effectiveness of the proposed approach is verified by experiments.

Keywords

Semantic SLAM, data association, object segmentation, semantic constraint

Date received: 30 September 2019; accepted: 5 January 2020

Topic: Service Robotics

Topic Editor: Henry Leung

Associate Editor: Yan Zhuang

Introduction

Simultaneously localization and mapping (SLAM) has become a very popular research direction in recent years, which requires to construct and update an environment map while simultaneously tracking an agent's position.^{1,2} SLAM has a variety of applications such as autonomous driving, mobile robots, and virtual reality. Especially, visual SLAM has received extensive attentions due to the large amount of information, wide range of application scenarios, and low cost of visual sensors.^{3,4} Compared with monocular and stereo cameras, RGB-Depth (RGB-D) camera is widely used in indoor environments because they can directly provide the depth and color measurements of the scene. In this article, we focus on RGB-D SLAM.

For traditional visual SLAM, the feature-based approach^{5–7} and direct method^{8,9} are mainstream solutions, where low-level point information plays an important role. The former associates points in successive frames

according to the local appearance near every feature point, while the latter tracks points on the basis of constant brightness assumption.¹⁰ However, these methods suffer from

¹State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China

²School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

³College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao, China

⁴Department of Mechanics and Engineering Science, Beijing Innovation Center for Engineering Science and Advanced Technology, College of Engineering, Peking University, Beijing, China

Corresponding author:

Zhiqiang Cao, State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China; School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China.

Email: zhiqiang.cao@ia.ac.cn



illumination and viewpoint changes.^{11,12} Different viewpoints and illuminations can lead to the variations of local appearance and brightness of the same point, which will cause the tracking failure of points with incorrect data association. Thus, the localization accuracy of the visual SLAM is decreased. On the other hand, the traditional visual SLAM mainly focuses on low-level geometric information, which possibly results in a weak interaction with complex surrounding environments.¹³

With the development of deep learning, great progresses have been made in object detection and object segmentation whose high-level semantic information can better adapt to viewpoint and illumination changes. The purpose of object detection is to infer the locations and class labels of objects, where the location of the object is represented in the form of a bounding box. For object detection based on deep learning, it can be classified into approaches based on regional proposal and without regional proposal. The former is a two-stage process: firstly generate a series of candidate regions and then extract the features of the candidate regions for classification and boundary regression. Its popular methods include regions with convolutional neural network features (R-CNN), Fast R-CNN, and Faster R-CNN and so on. For the approaches without regional proposal, the global information of the image is directly used, and you only look once (YOLO),¹⁴ YOLO9000,¹⁵ YOLOv3,¹⁶ and single-shot multibox detector¹⁷ are the representative methods. Different from the bounding box of object detection, object segmentation predicts the class labels pixel by pixel, and it is related to semantic segmentation^{18,19} and instance segmentation.²⁰ A possible problem of object segmentation is its computation cost, which makes it hard to integrate into a real-time SLAM.

Driven by object detection and segmentation based on deep learning, the researchers concern semantic visual SLAM with the combination of object detection or segmentation. Semantics can not only help SLAM achieve better localization^{11,21–23} but also establish more abundant map. To improve the localization accuracy, semantic constraints are added. Lianos et al. constructed a semantic error function by utilizing semantic segmentation to promote point–point association.¹¹ An et al. evaluated the importance of each semantic category based on semantic segmentation for better visual features and the removal of outliers in the matching process.²¹ On the basis, the accuracy and robustness of localization are improved. Besides semantic constraint, pose optimization of objects is also considered. A 3-D cuboid object detection approach is proposed,²² and it is combined with the Oriented FAST and Rotated BRIEF (ORB) feature points to respectively build semantic error functions for static and dynamic environments. On this basis, poses of points, 3-D cuboids, and cameras are jointly optimized. Similarly, Li et al. utilized 3-D object detection with viewpoint classification as well as feature points for

constructing semantic constraints,²³ which is suitable for both static and dynamic conditions.

It shall be noted that existing semantic SLAM approaches mainly concern the constraints of camera–landmark, camera–camera, as well as different types of landmarks, where a landmark can be a point-type, and it can also be an object type. The constraint of landmarks with the same kind is seldom considered. In fact, there exists invariance in terms of relative distance and orientation between two static object landmarks, and it may be changed if only the aforementioned constraints are employed. It is beneficial for the localization by introducing the relative constraints among objects into the SLAM optimization process. In this article, we propose a real-time visual Point-Object SLAM (PO-SLAM) approach on the basis of RGB-D ORB-SLAM2, which incorporates object–object constraint in the bundle adjustment (BA) optimization process. To ensure the real-time performance of the system while considering the instantiation of the objects, YOLOv3¹⁶ is adopted and it is combined with a rough geometric segmentation based on depth histogram to obtain the contours of objects, which can improve the association quality. Moreover, the object–object constraint is reflected by the relative position invariance of objects, which is converted to the length and orientation invariances of the line segment connecting every two objects in each frame. This provides additional information for pose optimization.

In the following, we will describe the proposed PO-SLAM approach combining points and objects in detail. Then, the experiments are presented, and finally, we conclude the article.

The proposed semantic PO-SLAM with points and objects

The framework of the proposed semantic PO-SLAM is shown in Figure 1, where point features, point–point association, and point–point constraint are directly used according to ORB-SLAM2.⁷ In the feature extraction module, object features are extracted from the color image provided by RGB-D camera using YOLOv3.¹⁶ Considering that object detection cannot accurately express the contours of objects, we utilize the depth image to geometrically segment the detected objects based on depth histograms. Then, combined with point features, point–object association is executed to obtain the feature points on each detected object. After extracting the features of every frame, we track the features between the current frame and the previous frame. And besides the point–point association, object–object interframe association is also executed. On this basis, the extracted point and object features as well as the association results are involved in the BA optimization process. With the help of loop closing of ORB-SLAM2, SLAM is finally implemented. In the following, we will address the PO-SLAM in detail.

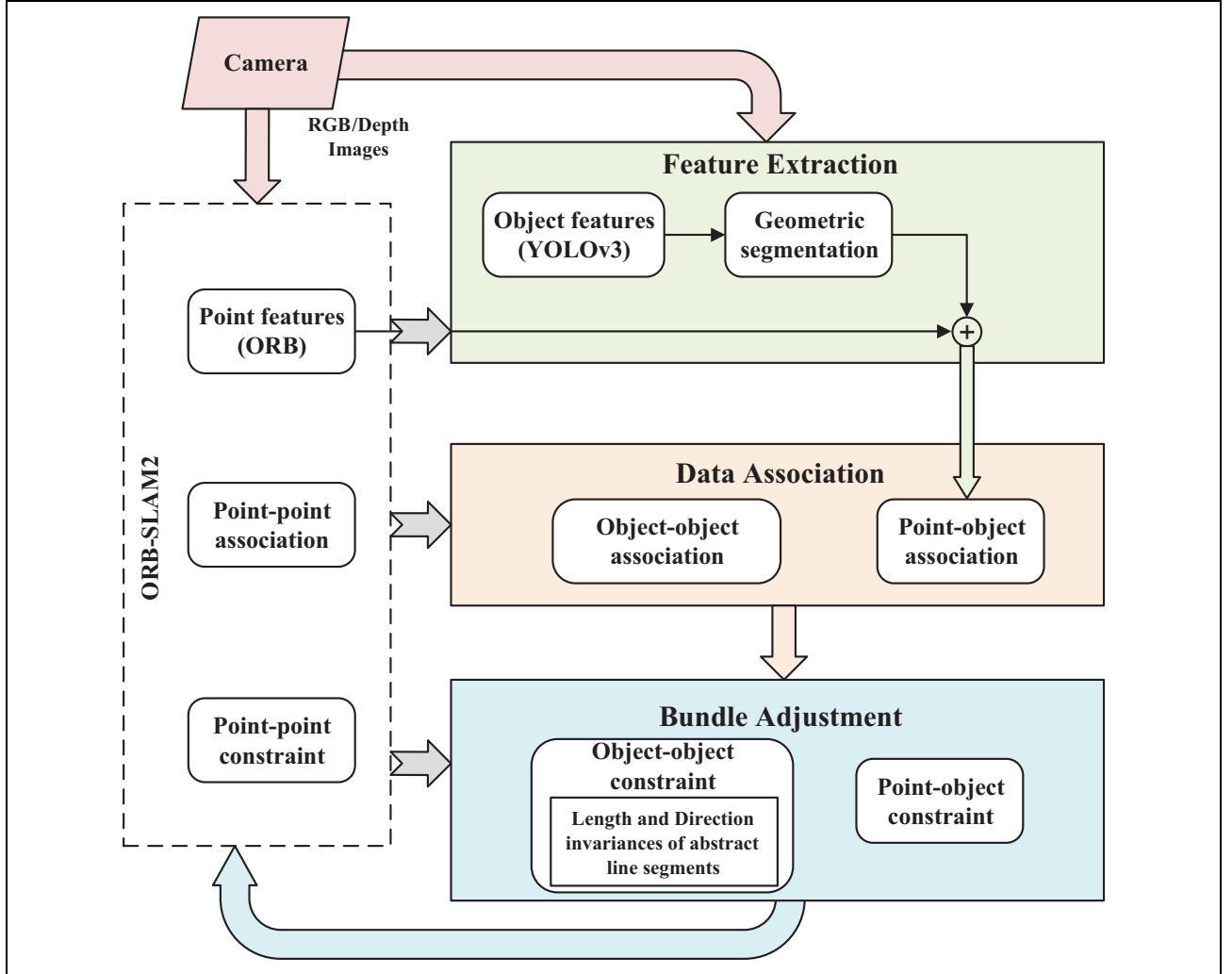


Figure 1. Overall framework of the semantic PO-SLAM approach. SLAM: simultaneously localization and mapping.

Feature extraction

Low-level point features are combined with high-level semantic object features in our SLAM. The reader may refer to the study by Mur-Artal and Tardós⁷ for point features extraction, and in this section, we focus on the extraction of object features.

Object features extraction. Object features including the objects number, categories, as well as the positions are favorable for data association of SLAM due to the reliability of high-level feature. In this article, YOLOv3¹⁶ is utilized to detect the objects at each frame, where the deep network is trained on the MS COCO data set including 80 categories of common objects. By object detection, the bounding boxes, labels, and label confidences of objects are obtained. Note that we only reserve the results with confidence of more than 70%.

Geometric segmentation. For object detection, the resulting bounding box surrounding an object cannot fit the actual

boundary of object completely, and some background information is inevitably contained. In this case, it is not easy to judge whether a feature point is on an object, which will affect the determination of the object's position. Also, in spite of good performance in segmentation effect, instance segmentation based on deep learning needs to take more time. A fast segmentation solution to extract the foreground in the bounding box of an object is required. Herein, a geometric segmentation based on depth histogram is presented.

In a detection bounding box, there are only two types of pixels: background and foreground. Their differentiation may be solved using depth information that reflects the distance between an object and the camera, and a depth threshold to separate the foreground from the background needs to be determined. With the depth values of foreground and background, we utilize the Otsu threshold segmentation method²⁴ to segment the depth values by maximizing the interclass variance of these two parts. Otsu is a method to automatically determine the threshold;

Algorithm 1. Geometrical segmentation process.

Input: bounding box $o = \{(b^l, b^b), (b^r, b^t)\}$ of detected object, depth image $D_{m \times n}$ of current frame.

Output: segmentation mask $M_{h \times w}$ for the detected object.

```

1  $d_{h \times w} = D(o), \text{vector\_d} = [], D_{th} = 0;$ 
2  $\text{vector\_d}_{max} = 0, \text{vector\_d}_{min} = 0;$ 
3 for  $i=0, 1, \dots, h-1$ 
4   for  $j=0, 1, \dots, w-1$ 
5     if  $d[i][j] > 0$  then
6        $\text{vector\_d.push\_back}(d[i][j]);$ 
7     end if
8   end for
9 end for
10  $\text{vector\_d}_{max} = \max(\text{vector\_d});$ 
11  $\text{vector\_d}_{min} = \min(\text{vector\_d});$ 
12 for  $k=0, 1, \dots, \text{Size}(\text{vector\_d}) - 1$  do
13    $\text{vector\_d}[k] = \frac{\text{vector\_d}[k] - \text{vector\_d}_{min}}{\text{vector\_d}_{max} - \text{vector\_d}_{min}} \times 255;$ 
14 end for
15  $D_{th} = \text{Otsu\_threshold}(\text{vector\_d});$ 
16  $D_{th} = D_{th} \times (\text{vector\_d}_{max} - \text{vector\_d}_{min}) / 255 + \text{vector\_d}_{min};$ 
17 for  $i=0, 1, \dots, h-1$ 
18   for  $j=0, 1, \dots, w-1$ 
19     if  $d[i][j] > 0 \ \& \ d[i][j] < D_{th}$  then
20        $M[i][j] = 1;$ 
21     else
22        $M[i][j] = 0;$ 
23     end if
24   end for
25 end for
26 return  $M$ 

```

however, it is sensitive to noise. For the depth map provided by the RGB-D camera, there exists the case where the depth value of a pixel is 0, which may be caused by the pixels outside the depth range or miss detection. Those pixels with a depth value of 0 in the depth map should be first filtered out before calculating the depth threshold.

To obtain the geometrical segmentation for an object, the depth image of current frame is cropped according to the predicted bounding box, and one can obtain the depth submap $d_{h \times w}$. After $d_{h \times w}$ is filtered, its values are scaled to $[0, 255]$, which is used to acquire the threshold using the Otsu method. On this basis, the foreground and background corresponding to the object is separated. The detailed process is given in Algorithm 1. (b^l, b^b) and (b^r, b^t) are the left bottom coordinate and the upper right coordinate of the bounding box, respectively, and $h = b^b - b^t$, $w = b^r - b^l$. D_{th} refers to the depth threshold, and the segmentation mask is labelled as M .

Figure 2 illustrates the segmentation result. Take the teddy bear in the original image from the TUM data set²⁵ (see Figure 2(a)) as an example. Figure 2(b) provides the detection result, and the depth histogram of the pixels in the bounding box is presented in Figure 2(c). One can see that the depth values are divided into two parts by the yellow dashed line corresponding to the depth threshold D_{th} . The

left part and right part of the dashed line are corresponding to the depth values of foreground and background. With the segmentation, the extracted foreground is given in Figure 2(d) for the object in Figure 2(b).

Data association

As a reflection of the common view between frames, data association is important in solving camera poses and landmark positions of SLAM. In addition to the association of interframe point features used in ORB-SLAM2,⁷ we also take the correlation of point features and object features in each frame as well as the association of interframe object features into account.

Point-object association. As mentioned above, for each detected bounding box in each frame, the foreground image is separated by the depth image information, and the feature points located in the foreground area are used as the feature points corresponding to the object. The association of points and objects is used to calculate the point-object error in the subsequent BA optimization. Figure 3 gives an illustration of association results for a selected image in fr2/desk of the TUM RGB-D data set.²⁵ The bounding boxes of different classes of objects are represented by different colors, and the color of feature points belonging to the same object is consistent with that of the bounding box. Notice that multiple object instances of the same class can be distinguished by the positions of their bounding boxes, and the green points do not belong to any detected object, which are considered as the background. When the points fall within the bounding box of an object and their colors match the color of the bounding box, they are regarded as the feature points associated with the object.

Object-object association. Object-object association between two frames is similar to standard object tracking. Since we have known the categories of the objects in each frame, we can only concern the object categories that simultaneously appear in two frames. At first, the center (u_c, v_c) of an object in the previous color image is unprojected to the world coordinate system by its depth d_c and the camera pose $T_{cw,pre}$ of the previous frame. Then 3-D position P_c of the object center is projected to the current image using the camera pose $T_{cw,cur}$ of the current frame.

$$P_c = \pi^{-1}(T_{cw,pre}, d_c, u_c, v_c) \quad (1)$$

$$(u_c, v_c)_{proj} = \pi(T_{cw,cur}, P_c) \quad (2)$$

where π and π^{-1} represent the projection from 3-D space to 2-D image and unprojection from 2-D image to 3-D space, respectively. $(u_c, v_c)_{proj}$ refers to the projection of P_c on the current frame. After the projection of the object center on the current frame is acquired, we check the relationship of the projection and the bounding boxes in the current frame with the constraint of the same object label. If the distance

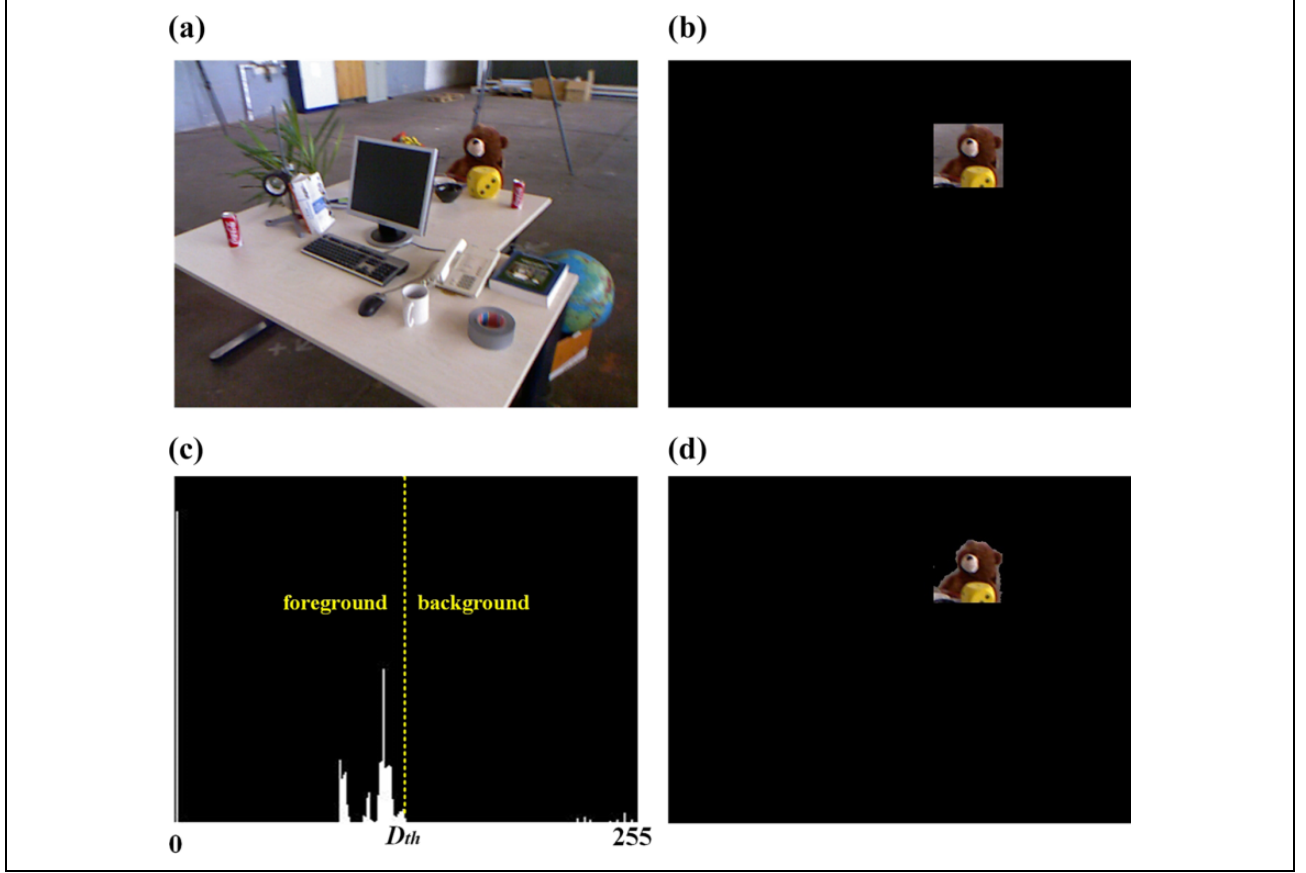


Figure 2. The geometric segmentation. (a) Original image. (b) One detected object. (c) The depth histogram of the bounding box in (b). (d) The extracted foreground after the segmentation.



Figure 3. Results of point-object association for an image in fr2/desk of TUM RGB-D data set, where the color of points belonging to the same object is the same as that of the corresponding bounding box.

between the projection and the center of a bounding box in the current frame is less than a given threshold, the corresponding two objects are considered as a successful match.

Bundle adjustment

Combining point and object features, constraints with geometric and semantic relationships are constructed to optimize camera poses and 3-D point positions. The sets of image sequence, positions of 3-D points, and objects in the world coordinate system are denoted as $I = \{I_k\}$, $P = \{P_i\}$, and $O = \{O_j\}$, respectively, where k , i , and j are their corresponding indexes. For a 3-D point, it is either on the object or belongs to the background. We label the position of the i th point on the j th object as jP_i . Also, the object is represented by the points inside the object and its position, and $\{O_j\} = \{{}^jP_i\}, C_j\}$, where C_j is the 3-D position of the j th object.

We can observe the measurements corresponding to 3-D points and objects from each frame. $o = \{o_{kj}\}$ and $z = \{z_{ki}\}$ are used to stand for the observations of j th object O_j and the i th point in the k th frame. $o_{kj} = \{b_{kj}^l, b_{kj}^b, b_{kj}^r, b_{kj}^t, c_{kj}, l_{kj}\}$, where c_{kj} and l_{kj} are the observations of object position and class label. We denote ${}^jz_{ki}$ with the observation on the j th object for the i th point in the k th frame.

BA formulation. Our semantic optimization process can be described as the following problem: given the observations

$\{z_{ki}\}$ of points in the k th frame and observations $\{o_{kj}\}$ of objects $\{O_j\}$ in the k th frame, find the optimized camera pose $T_{cw,k}^*$ and the positions $\{P_i^*\}$ of the points, where $T_{cw,k} \in SE(3)$ is used to convert 3-D points from the world coordinate system to the camera coordinate system, and $P_i \in R^3$. In the BA process, the optimization process is executed by minimizing the errors between the predicted values and the measured values, which is a nonlinear least-squares problem. Our measurement errors consist of point–point error, point–object error, and object–object error, and the optimization function can be formulated as follows

$$T_{cw,k}^*, P_i^* = \arg \min_{T_{cw,k}, P_i} \sum_{k,j,i} \left\{ \|e_{pp}(z_{ki}, T_{cw,k}, P_i)\|_{\Sigma_{k,i}}^2 + \|e_{oo}(o_{kj}, T_{cw,k}, O_{j_1}, O_{j_2})\|_{\Sigma_{k,j_1,j_2}}^2 + \|e_{po}(o_{kj}, T_{cw,k}, P_i)\|_{\Sigma_{k,i,j}}^2 \right\} \quad (3)$$

where $e_{pp}(\cdot)$, $e_{po}(\cdot)$, and $e_{oo}(\cdot)$ represent the errors between the projected point on the image by the camera pose and the observation point for P_i , between projected point and 2-D bounding box and between two objects, respectively. In this article, the Levenberg–Marquardt method is adopted to solve this problem.

Error functions. Point–point error. With the ORB features, point–point error (i.e. re-projection error) is given by⁷:

$$e_{pp}(z_{ki}, T_{cw,k}, P_i) = \pi(T_{cw,k}, P_i) - z_{ki} \quad (4)$$

Point–object error. Based on the point and object data association, we get $\{^jP_i\}$ that belong to an object O_j . Theoretically, after these points are projected into the current frame, they should fall into the corresponding 2-D bounding box of the object O_j but that is not always the case. Our point–object error for the point jP_i is as follows.

$$e_{po}(o_{kj}, T_{cw,k}, ^jP_i) = (\text{err}_u, \text{err}_v) \quad (5)$$

where

$$\text{err}_u = \begin{cases} 0 & b_{kj}^l \leq u_{\text{proj}} \leq b_{kj}^r \\ u_{\text{proj}} - b_{kj}^r & u_{\text{proj}} > b_{kj}^r \\ b_{kj}^l - u_{\text{proj}} & u_{\text{proj}} < b_{kj}^l \end{cases} \quad (6)$$

$$\text{err}_v = \begin{cases} 0 & b_{kj}^b \leq v_{\text{proj}} \leq b_{kj}^t \\ v_{\text{proj}} - b_{kj}^t & v_{\text{proj}} > b_{kj}^t \\ b_{kj}^b - v_{\text{proj}} & v_{\text{proj}} < b_{kj}^b \end{cases} \quad (7)$$

$$(u_{\text{proj}}, v_{\text{proj}}) = \pi(T_{cw,k}, ^jP_i) \quad (8)$$

$(u_{\text{proj}}, v_{\text{proj}})$ is the projected pixel coordinate of jP_i , and err_u and err_v are the u -axis error and v -axis error between projected point and 2-D bounding box. It shall be noted that when the projection point is inside the detected bounding

box, the cost function is always zero, and thus this constraint is relatively coarse. Only when the projection point falls outside the detection box, does the penalty take effect.

Object–object error. we acquire the feature points belonging to the objects as well as corresponding 3-D points through the point–object data association. And then use the coordinate centroid of these 3-D points as the 3-D position of the object with the coordinate centroid of corresponding ORB feature points in the image as the observation of the object position, which are described as

$$C_j = \frac{1}{N} \sum_i ^jP_i \quad (9)$$

$$c_{kj} = \frac{1}{N} \sum_i ^jz_{ki} \quad (10)$$

where N is the number of points anchored to the object O_j .

The relative position between two objects is constrained by distance and orientation. To solve the problem, we connect the positions of two objects into an abstract line segment, and thus the distance and direction constraints can be converted to the invariance of length and direction of the line segment. We define c_{kj_1} and c_{kj_2} as the observations of positions for objects O_{j_1} and O_{j_2} in the k th frame, respectively. Correspondingly, C_{j_1} and C_{j_2} represent the 3-D positions of objects O_{j_1} and O_{j_2} . According to Hartley and Zisserman,²⁶ we define $c_{kj_1}^h$ and $c_{kj_2}^h$ as the homogeneous coordinates of c_{kj_1} and c_{kj_2} for the parameterized representation of the line segment. Thus, the line through c_{kj_1} and c_{kj_2} can be expressed as follows

$$l = \frac{c_{kj_1}^h \times c_{kj_2}^h}{c_{kj_1}^h \times c_{kj_2}^h} \quad (11)$$

According to the direction invariance constraint, we can infer that the projection points of object O_{j_1} and object O_{j_2} should be located on the line l . The direction error can be denoted as follows

$$e_{oo_dir} = (l^T \pi(T_{cw,k}, C_{j_1}), l^T \pi(T_{cw,k}, C_{j_2})) \quad (12)$$

The length invariance of the line segment indicates that the distance between the projected points is the same as that of c_{kj_1} and c_{kj_2} . Then, the distance error is given by

$$e_{oo_dis} = D(p_1, p_2) - D(c_{kj_1}, c_{kj_2}) = \sqrt{(p_1^u - p_2^u)^2 + (p_1^v - p_2^v)^2} - \sqrt{(c_{kj_1}^u - c_{kj_2}^u)^2 + (c_{kj_1}^v - c_{kj_2}^v)^2} \quad (13)$$

where

$$p_1 = \pi(T_{cw,k}, C_{j_1}) \quad (14)$$

$$p_2 = \pi(T_{cw,k}, C_{j_2}) \quad (15)$$

Table 1. Comparison of our methods with ORB-SLAM2 according to absolute trajectory errors.

Sequence	ORB-SLAM2 ATE (m)	PO-SLAM1		PO-SLAM2		PO-SLAM	
		ATE (m)	ATE _{rel} (%)	ATE (m)	ATE _{rel} (%)	ATE (m)	ATE _{rel} (%)
fr1/desk	0.0153	0.0158	-3.27	0.0152	0.65	0.0153	0.00
fr1/desk2	0.0239	0.0220	7.95	0.0234	2.09	0.0214	10.46
fr1/room	0.0537	0.0516	3.91	0.0484	9.87	0.0481	10.43
fr1/xyz	0.0097	0.0097	0.00	0.0096	1.03	0.0096	1.03
fr2/desk	0.0094	0.0092	2.13	0.0091	3.19	0.0090	4.26
fr2/xyz	0.0036	0.0037	-2.78	0.0036	0.00	0.0035	2.78
fr3/long_office	0.0100	0.0097	3.00	0.0102	-2.00	0.0096	4.00
fr3/sitting_xyz	0.0093	0.0090	3.23	0.0091	2.15	0.0091	2.15
fr3/sitting_static	0.0087	0.0080	8.05	0.0084	3.45	0.0079	9.20
fr3/walking_xyz	0.7127	0.7012	1.61	0.7128	-0.01	0.7025	1.43

SLAM: simultaneously localization and mapping; ATE: absolute trajectory error with root mean square.

Table 2. Comparison of our methods with ORB-SLAM2 according to relative pose errors.

Sequence	ORB-SLAM2 RPE (m)	PO-SLAM1		PO-SLAM2		PO-SLAM	
		RPE (m)	RPE _{rel} (%)	RPE (m)	RPE _{rel} (%)	RPE (m)	RPE _{rel} (%)
fr1/desk	0.0279	0.0283	-1.43	0.0271	2.87	0.0277	0.72
fr1/desk2	0.0409	0.0408	0.24	0.0399	2.44	0.0388	5.13
fr1/room	0.0840	0.0763	9.17	0.0762	9.29	0.0748	10.95
fr1/xyz	0.0129	0.0128	0.78	0.0127	1.55	0.0129	0
fr2/desk	0.0324	0.0313	3.40	0.0312	3.70	0.0319	1.54
fr2/xyz	0.0106	0.0104	1.89	0.0103	2.83	0.0103	2.83
fr3/long_office	0.0234	0.0226	3.42	0.0230	1.71	0.0226	3.42
fr3/sitting_xyz	0.0121	0.0117	3.31	0.0118	2.48	0.0116	4.13
fr3/sitting_static	0.0115	0.0109	5.22	0.0111	3.48	0.0107	6.96
fr3/walking_xyz	0.8439	0.8266	2.05	0.8464	-0.30	0.8115	3.84

SLAM: simultaneously localization and mapping; RPE: mean relative pose error.

where p_1 and p_2 represent the 2-D pixel coordinates of the projections of C_{j_1} and C_{j_2} in the image, respectively, and $D(\cdot)$ refers to the Euclidean distance of two pixels. e_{oo_dir} and e_{oo_dis} constitute the object-object error function.

Experiments and results

In this section, we will evaluate the localization performance of our approach and conduct the comparison with ORB-SLAM2.

Experimental setup

We adopt the TUM RGB-D SLAM data set and benchmark^{25,27} to test and validate the approach. TUM data set consists of different types of sequences, which provide color and depth images with a resolution of 640×480 using a Microsoft Kinect sensor. YOLOv3 scales the original images to 416×416 . Combining objects we concerned such as book, keyboard, mouse, TV-monitor, cup, cell phone, remote, bottle, teddy bear, and potted plant, 10 sequences related to office environments are selected.

We adopt the following evaluation metrics²⁷: absolute trajectory error with root mean square (ATE) and mean relative pose error (RPE), where ATE quantifies the difference between points of the estimated trajectory and their ground truths, whereas RPE assesses the local accuracy of the estimated poses in a fixed interval. All of the experiments are repeated five times and the median of these five results is considered as the final result. To clearly demonstrate the improvement of our method, ATE_{rel}^{11} and RPE_{rel} are considered, where the former refers to the relative ATE and the latter reflects the relative RPE. $ATE_{rel} = (ATE_{ORB_SLAM2} - ATE_{+semantics})100 / ATE_{ORB_SLAM2}$ and $RPE_{rel} = (RPE_{ORB_SLAM2} - RPE_{+semantics})100 / RPE_{ORB_SLAM2}$, where $ATE_{+semantics}$ and $RPE_{+semantics}$ represent the ATE and RPE of our semantic SLAM. All of our experiments are run on a desktop with an Intel Core i7-7700HQ CPU and Nvidia GTX 1080 GPU. Only the object detection is executed on the GPU.

Experiment on the TUM RGB-D data set²⁵

Tables 1 and 2 give the comparison of our PO-SLAM and ORB-SLAM2 over 10 sequences. To better address our

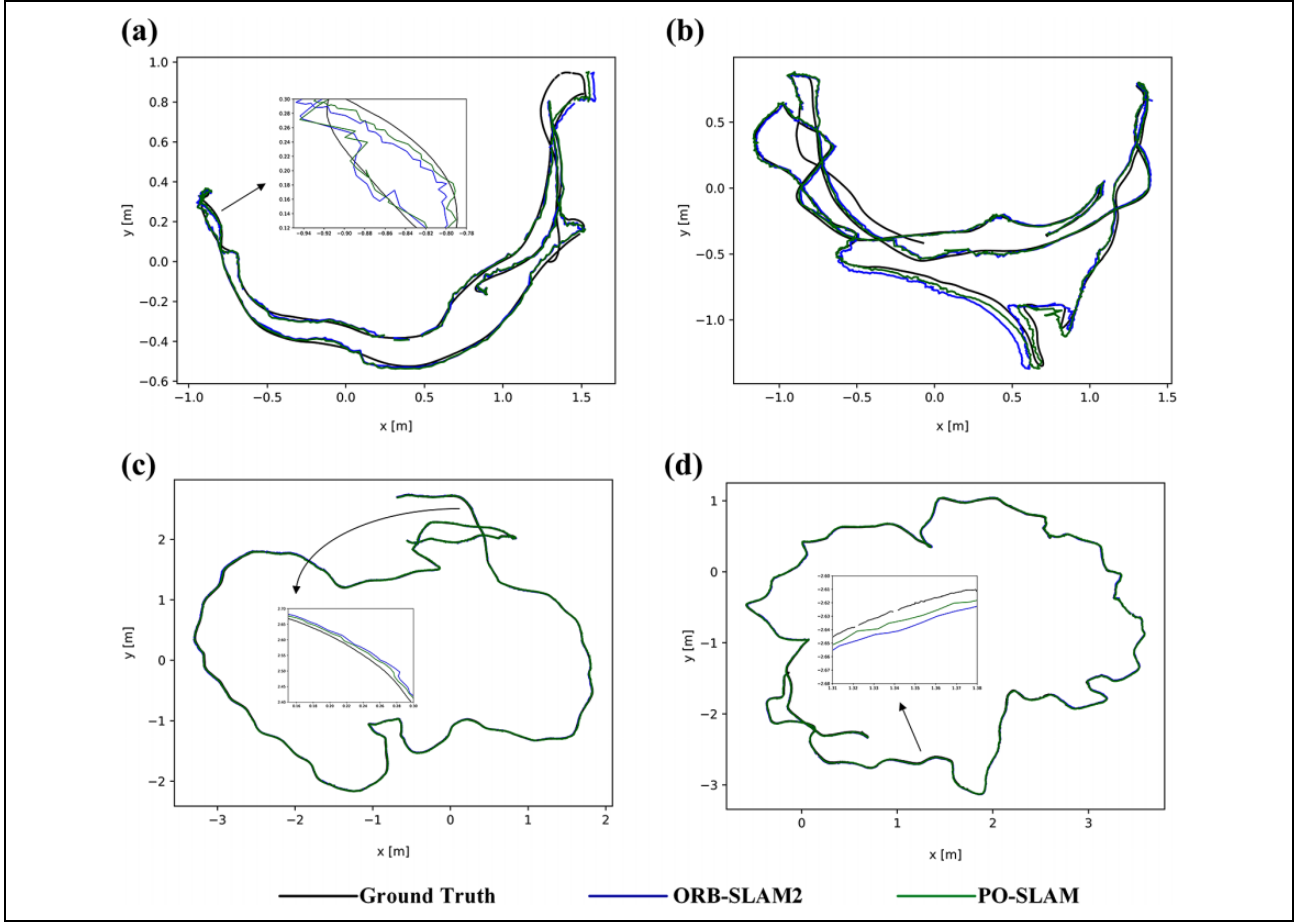


Figure 4. Comparison of trajectories estimated by our PO-SLAM, ORB-SLAM2, and ground truth on the TUM RGB-D data set. (a) fr1/dek2, (b) fr1/room, (c) fr3/office and (d) fr2/desk. SLAM: simultaneously localization and mapping.

approach, we also consider two other methods PO-SLAM1 and PO-SLAM2. These two methods correspond to the cases of PO-SLAM without point–object error in (3) and PO-SLAM without object–object error in (3), respectively. Noticing that the first seven sequences describe static scenes, whereas the last three sequences are related to dynamic scenes.

As can be seen in Tables 1 and 2, our PO-SLAM has an improvement of up to 10.46% in ATE and up to 10.95% in RPE compared with ORB-SLAM2. Overall, our three methods perform better than ORB-SLAM2 in both ATE and RPE for most of the sequences, and PO-SLAM performs best.

Figure 4 depicts the comparison of the trajectories obtained by PO-SLAM and ORB-SLAM2 on four sequences with the ground truth. It is seen that our trajectories are closer to the ground truth than ORB-SLAM2. Note that all ORB features extracted by ORB-SLAM2 are used in our point–point error. From Tables 1 and 2, our method has proved a better adaptability to dynamic environments. Figure 5 illustrates a performance comparison of PO-SLAM and ORB-SLAM2 on fr3/walking_xyz dynamic sequence.²⁵ Clearly, ORB-SLAM2 fails to track

on the frame 696 and frame 768, while PO-SLAM is still in the SLAM mode with enough matching points with the previous frame.

The average running time per frame of PO-SLAM is demonstrated in Figure 6 for 10 sequences on the TUM RGB-D data set. It is seen that the average time is 71.47 ms with a speed about 14 fps, which meets the real-time requirement.

Conclusions

In this article, we propose a semantic visual SLAM approach combining 2-D object detection and ORB feature points with additional semantic constraints for the process of BA optimization. The object segmentation approach combining object detection and the depth histogram of 2-D bounding box is used to associate feature points and their corresponding objects. Besides, the correlation between any two detected objects within the field of view of each frame is also introduced. Experimental results on the TUM RGB-D data set indicate that our approach can improve the accuracy and robustness compared with ORB-SLAM2.

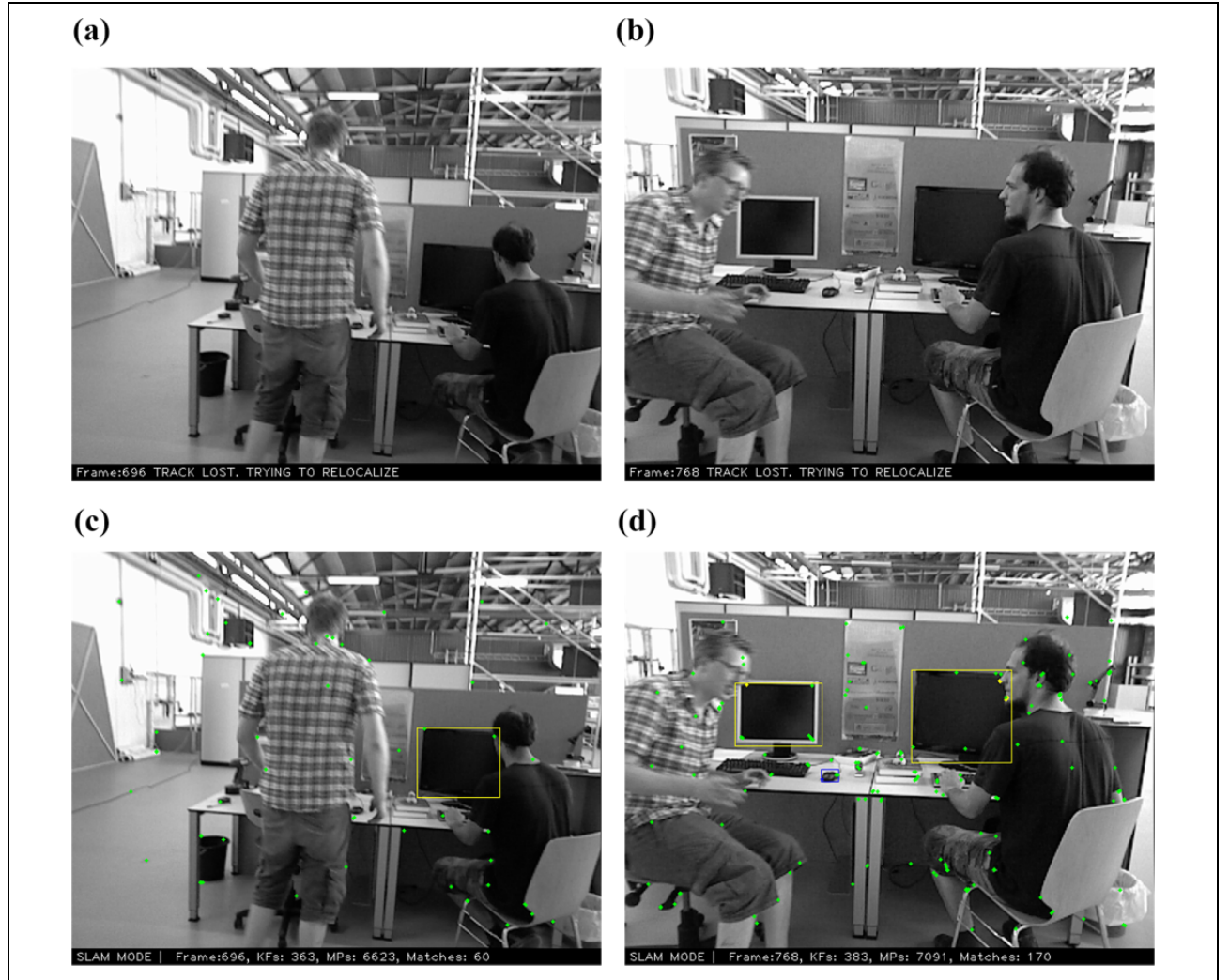


Figure 5. Comparison of our PO-SLAM and ORB-SLAM2 on fr3/walking_xyz. (a) and (b) The results of ORB-SLAM2, and (c) and (d) the results of PO-SLAM. SLAM: simultaneously localization and mapping.

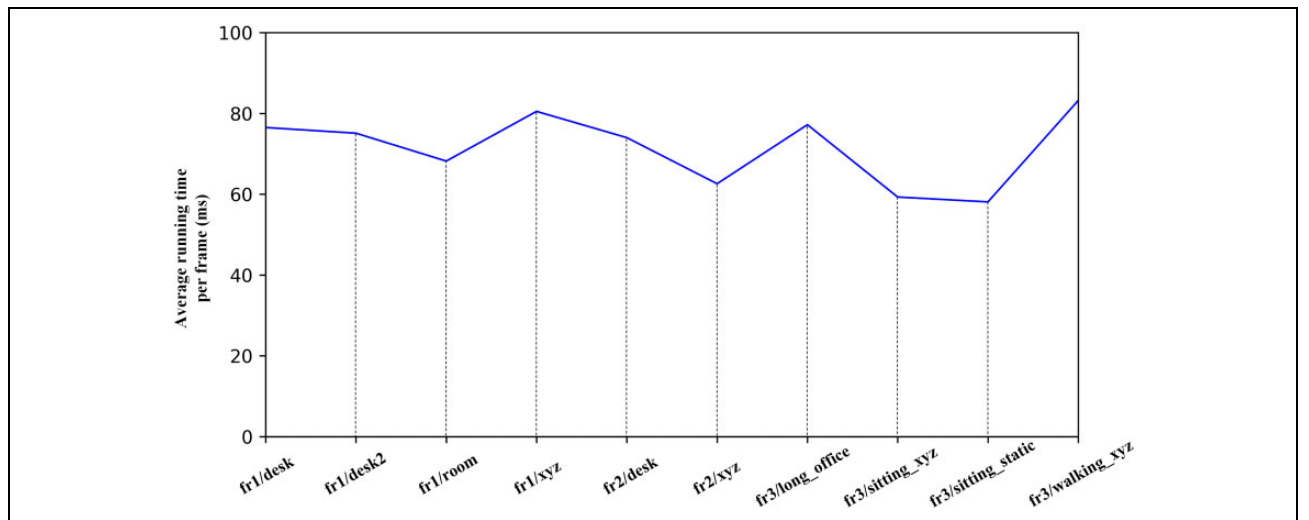


Figure 6. Average running time per frame of PO-SLAM on the TUM RGB-D data set. SLAM: simultaneously localization and mapping.


Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the Key Research and Development Program of Shandong Province under grant 2017CXGC0925, in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under grant 2018IRS21, and in part by the National Natural Science Foundation of China under grants 61633017, 61633020, and 61836015.

ORCID iD

Peiyu Guan  <https://orcid.org/0000-0003-1909-1953>

References

1. Taketomi T, Uchiyama H, and Ikeda S. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Trans Comput Vis Appl* 2017; 9: 16.
2. Cadena C, Carlone L, Carrillo H, et al. Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. *IEEE Trans Robot* 2016; 32(6): 1309–1332.
3. Fuentes-Pacheco J, Ruiz-Ascencio J, and Rendón-Mancha JM. Visual simultaneous localization and mapping: a survey. *Artif Intell Rev* 2015; 43(1): 55–81.
4. Wang Y, Wang R, Wang S, et al. Underwater bio-inspired propulsion: from inspection to manipulation. *IEEE Trans Ind Electron* 2019. DOI: 10.1109/TIE.2019.2944082.
5. Klein G and Murray D. Parallel tracking and mapping for small AR workspaces. In: *IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, 13–16 November 2007, pp. 225–234.
6. Mur-Artal R, Montiel JMM, and Tardós JD. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans Robot* 2015; 31(5): 1147–1163.
7. Mur-Artal R and Tardós JD. ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans Robot* 2017; 33(5): 1255–1262.
8. Engel J, Koltun V, and Cremers D. Direct sparse odometry. *IEEE Trans Pattern Anal Mach Intell* 2018; 40(3): 611–625.
9. Engel J, Schöps T, and Cremers D. LSD-SLAM: large-scale direct monocular SLAM. In *European Conference on Computer Vision*, Zurich, Switzerland, 6–12 September 2014, pp. 834–849.
10. Park S, Schöps T, and Pollefeys M. Illumination change robustness in direct visual SLAM. In: *IEEE International Conference on Robotics and Automation*, Singapore, 29 May–3 June 2017, pp. 4523–4530.
11. Lianos KN, Schönberger JL, Pollefeys M, et al. VSO: visual semantic odometry. In *European Conference on Computer Vision*, Munich, Germany, 8–14 September 2018, pp. 246–263.
12. Toft C, Stenborg E, Hammarstrand L, et al. Semantic match consistency for long-term visual localization. In *European Conference on Computer Vision*, Munich, Germany, 8–14 September 2018, pp. 391–408.
13. Zhang L, Wei L, Shen P, et al. Semantic SLAM based on object detection and improved octomap. *IEEE Access* 2018; 6: 75545–75559.
14. Redmon J, Divvala S, Girshick R, et al. You only look once: unified, real-time object detection. In: *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 27–30 June 2016, pp. 779–788.
15. Redmon J and Farhadi A. YOLO9000: better, faster, stronger. In: *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 21–26 July 2017, pp. 6517–6525.
16. Redmon J and Farhadi A. YOLOv3: an incremental improvement. *arXiv:1804.02767*, 2018.
17. Liu W, Anguelov D, Erhan D, et al. SSD: single shot multi-box detector. In *European Conference on Computer Vision*, Amsterdam, Netherlands, 8–16 October 2016, pp. 21–37.
18. Long J, Shelhamer E, and Darrell T. Fully convolutional networks for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, 7–12 June 2015, pp. 3431–3440.
19. Badrinarayanan V, Kendall A, and Cipolla R. SegNet: a deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans Pattern Anal Mach Intell* 2017; 39(12): 2481–2495.
20. He K, Gkioxari G, Dollar P, et al. Mask R-CNN. In: *IEEE International Conference on Computer Vision*, Venice, Italy, 22–29 October 2017, pp. 2961–2969.
21. An L, Zhang X, Gao H, et al. Semantic segmentation-aided visual odometry for urban autonomous driving. *Int J Adv Robot Syst* 2017; 14(5): 1–11.
22. Yang S and Scherer S. CubeSLAM: monocular 3-D object SLAM. *IEEE Trans Robot* 2019; 35(4): 925–938.
23. Li P, Qin T, and Shen S. Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving. In *European Conference on Computer Vision*, Munich, Germany, 8–14 September 2018, pp. 664–679.
24. Otsu N. A threshold selection method from gray-level histograms. *IEEE T Syst Man Cy* 1979; 9(1): 62–66.
25. <http://vision.in.tum.de/data/datasets/rgbd-dataset>.
26. Hartley R and Zisserman A. *Multiple view geometry in computer vision*. Cambridge: Cambridge University Press, 2000.
27. Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems. In: *IEEE/RSSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, 7–12 October 2012, pp. 573–580.