International Journal of Advanced Robotic Systems
January-February 2021: 1–12
© The Author(s) 2021
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1729881421994447
journals.sagepub.com/home/arx
SAGE

JD-SLAM: Joint camera pose estimation and moving object segmentation for simultaneous localization and mapping in dynamic scenes

Yujia Zhai[1,2,3], Baoli Lu[1,2,3], Weijun Li[1,2,3], Jian Xu[1,2,3] and Shuangyi Ma[1,2,3]

Abstract
As a fundamental assumption in simultaneous localization and mapping, the static scenes hypothesis can be hardly fulfilled in applications of indoor/outdoor navigation or localization. Recent works about simultaneous localization and mapping in dynamic scenes commonly use heavy pixel-level segmentation net to distinguish dynamic objects, which brings enormous calculations and limits the real-time performance of the system. That restricts the application of simultaneous localization and mapping on the mobile terminal. In this article, we present a lightweight system for monocular simultaneous localization and mapping in dynamic scenes, which can run in real time on central processing unit (CPU) and generate a semantic probability map. The pixel-wise semantic segmentation net is replaced with a lightweight object detection net combined with three-dimensional segmentation based on motion clustering. And a framework integrated with an improved weighted-random sample consensus solver is proposed to jointly solve the camera pose and perform three-dimensional object segmentation, which enables high accuracy and efficiency. Besides, the prior information of the generated map and the object detection results is introduced for better estimation. The experiments on the public data set, and in the real-world demonstrate that our method obtains an outstanding improvement in both accuracy and speed compared to state-of-the-art methods.

Keywords
SLAM, dynamic scenes, object detection, 3D segmentation

Date received: 2 December 2020; accepted: 25 January 2021

Topic: Vision Systems
Topic Editor: Antonio Fernandez Caballero
Associate Editor: Shengyoung Chen

Introduction

Simultaneous localization and mapping (SLAM) is a technology that builds a map of an unknown environment while simultaneously localizing the sensor with the input information. It has been widely used in mobile robots, autopilot, or AR/VR applications over the past decades.

There is a basic assumption in the visual SLAM that the environment is static and the changes in visual field only come from the motion of the camera. So the previous SLAM systems treat the environment as a whole rigid body. Unfortunately, in some main application scenarios

[1] Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China
[2] Center of Materials Science and Optoelectronics Engineering & School of Microelectronics, University of Chinese Academy of Sciences, Beijing 100049, China
[3] Beijing Key Laboratory of Semiconductor Neural Network Intelligent Sensing and Computing Technology, Beijing 100083, China

Corresponding authors:
Jian Xu and Baoli Lu, Institute of Semiconductors, Chinese Academy of Sciences, University of Chinese Academy of Sciences, No. 35 A, Qinghua East Road, Haidian District, Beijing 100089, China.
Email: xujian@semi.ac.cn; lubaoli@semi.ac.cn

of the SLAM system such as urban roads or indoor places with people, this assumption will not hold. The dynamic objects in the view will have a tremendous negative impact on the accuracy and robustness of the SLAM system. Therefore, the SLAM problem in a dynamic environment has attracted much attention in recent years.

The core problem of the SLAM for dynamic scenes is the confusion of multiple sets of geometric constraints that existed in the data association. These constraints themselves are inconsistent. If only two pictures are given, it is challenging to distinguish whether objects in the scene are moving. Humans can make a judgment by experience, which is based on the semantic information. That makes the SLAM technology naturally develop to semantic SLAM. A typical choice is to introduce a pixel-wise segmentation network such as Mask R-CNN[1] and FCN[2] to eliminate the dynamic parts before tracking and mapping, which brings huge consume of graphics processing resources at the same time. For instance, the Mask R-CNN takes more than 2 s to process a frame on CPU. Previous works usually make use of GPU[3–5] to accelerate computation and run very hard on CPU, not to mention on small mobile devices or operate in real time. In this article, we focus on the capacity of real-time operation on CPU of the SLAM system with semantic segmentation for better application in practice.

In addition, most of the existing dynamic SLAM solutions are designed for RGB-D SLAM.[3,5–7] Since the RGB-D camera can obtain the depth directly, it makes the dynamic object easily to be detected by the inconsistency between the actual depth and the projection depth. However, the RGB-D camera may be easily affected by complex illumination, and its effective distance is limited, generally no more than 10 m. That makes it unsuitable for applications in outdoor environments such as autonomous driving and drone navigation. In contrast, the monocular camera has no limitation of perception distance and can be easily integrated into mobile devices at low cost. And the monocular camera is more sensitive to dynamic objects due to its lack of depth information. But there are few solutions based on monocular SLAM.[8,9]

In this article, we propose a lightweight method to mitigate the negative effects of dynamic objects, which attains a real-time performance on CPU. The solution can be used in monocular SLAM as well as in stereo and RGB-D SLAM. The main contributions of this system can be summarized as follows:

- Instead of performing instance segmentation on the 2D image, we use the object detection network combined with the spatial information to segment the 3D point clouds directly.
- A framework with a weighted-random sample consensus (RANSAC) solver is designed, which hierarchically solves the camera pose and the motion of each moving object. The weight of each data point introduces the prior reliability as guidance to the pending model.
- A comprehensive feature evaluation approach with a semantic probability map that integrates multiple dimensions of information including the tracking quality, motion probability, and multi-view geometric information.

The proposed method is integrated into the ORB-SLAM2[10] and shows obvious improvements to the system in dynamic scenes. The experiments on the TUM-RGBD data set and the KITTI data set demonstrate that our method outperforms the state-of-the-art method Dyna-SLAM in accuracy with a huge advantage in speed.

The rest of this article is organized as follows. In section "Related work," we discuss the previous work about SLAM systems in dynamic scenes. Section "System" shows the framework of the proposed method and details of the proposed framework to estimate the camera pose and perform 3D segmentation. Section "Experiment" provides qualitative and quantitative experimental results. At last, the conclusions and future work are given in section "Conclusion."

Related work

Saputra et al.[11] summarized the previous work about the SLAM methods for dynamic scenes before 2018 and divided them into three types according to their outputs and applications: (1) for robustness SLAM; (2) for dynamic object tracking; (3) for joint segmentation and reconstruction.

Among them, a convention is to remove the dynamic objects as outliers, and the commonly used strategy can be summarized into two categories: motion consistency detection based on multi-view geometry and semantic segmentation based on deep learning network. Semantic-based methods can detect those objects that people subjectively believe that they will not remain stationary, such as vehicles and pedestrians, while geometry-based methods can identify those objects that are objectively moving. The two are often used alone or in combination.

DS-SLAM[6] applies SegNet to achieve real-time pixel-level segmentation and eliminate pedestrians in the image. The authors conduct dynamic point detection by optical flow from the difference between the patch on both sides of tracking and the distance of reprojection and treat all points in the semantic segmentation boundary containing dynamic points as dynamic points. What's more, an octree map is established. Zhong et al.[12] proposed to combine SLAM with a deep-neural-network-based object algorithm, aiming to make target detection and SLAM promote each other. This article uses the SSD-NET fine-tuned on the SLAM data set along with the GrabCut algorithm to segment the background and foreground and dynamically estimate the probability of feature points and 3D points to be dynamic points. The perception of "motion" is mainly derived from CNN object detection and is propagated through the 2D-2D, 3D-2D correspondences with the assumption that "Proximity is similarity." Mask-SLAM[13] uses the mask generated by semantic segmentation to remove feature points from the sky and cars and improve the robustness of vSLAM system. Zhang et al.[14] solves object association and pose optimization in a tightly coupled processing of optimization, by which both aspects can promote each other.

In addition, RGB-D camera is widely used as the sensor for SLAM in dynamic scenes as it can directly obtain depth information. Zhou et al.[15] utilizes k-means clustering after combining RGB images and depth maps to realize image segmentation. Dyna-SLAM[3] integrates Mask RCNN into ORB-SLAM2 for instance segmentation and performs dynamic object detection based on multi-view geometry when the input is RGB-D. What's more, they project the RGB and depth map of the previous keyframe onto the current frame to inpaint the occluded background without dynamic objects. Sünderhauf et al.[16] proposed a method of semantic mapping based on RGB-D data, which uses SSD-NET to perform object detection on a single RGB frame, and then applies the 3D segmentation algorithm proposed by Pham et al.[17] to further segment the 3D point clouds based on the detection results and depth information. An ICP-like method is applied for data association and fusion between objects. Zhang et al.[5] proposed a dense RGB-D SLAM system, which treats dynamic/static segmentation, camera ego-motion estimation, and map reconstruction as a whole and solves them simultaneously. The RGB images are sent into the PWC-Net for optical flow estimation and dynamic object segmentation is realized based on the optical flow residual between the warped image and the real frame.

In general, subject to the principle of consistent motion detection, dynamic SLAM only on pure multi-view geometry has many limitations and cannot identify objects with low speed or slowly moving parts. Recent studies are mostly focused on the use of convolutional neural networks for semantic segmentation of images, assisted by motion consistency detection based on multi-view geometry, such as the most commonly used optical flow method.[18,19] These two methods are executed in sequence, and the geometric motion detection is mainly for the purpose of checking and filling the semantic segmentation results. Our method utilizes a lightweight object detection network as the replacement of the heavy instance segmentation network. A spatial clustering algorithm based on the relative motion solver is proposed, which combines 3D segmentation, motion detection, and camera pose estimation, aiming to achieve a more accurate and robust track as well as outperforming real-time performance. Unlike most of the existing methods that use RGB-D camera as the sensor, our method targets to monocular SLAM systems.

Notation

The set of 3D points (map points) in the system is denoted as $\mathcal{X} = \left\{ \mathbf{X}_i = (x_i, y_i, z_i)^T, i = 1, 2, \ldots, N \right\}$, while $\S = \left\{ \mathbf{x}_i = (u_i, v_i)^T, i = 1, 2, \ldots, N \right\}$ is the 2D points (image points).The 3D-2D correspondences found by data association can be represented as $\mathcal{C} = \{C_i = (X_i, x_i), i = 1, 2..., N\}$. An object instance detected by CNN with a boundbox is donated as $\mathcal{O} = \{O_j = \{o_j, w_j, h_j \text{ label}\}$, here o_j is the center position of the bounding box, w_j, h_j is the width and the height, and label is the category of the object. $P_j(X_i)$ describes the probability of the map point, and X_i is exacted from the classified object O_j or the background.

System

In this article, the processing for dynamic scenes is divided into two aspects. The first is to make judgments about the probability of a feature comes from a dynamic object. The second is to reduce the impact of the dynamic objects based on the probability information from the first part while increasing the weight of "reliable" data points to gain more stable and accurate camera pose estimation results.

Figure 1 shows an overview of JD-SLAM. In addition to the tracking, local mapping, and loop closing thread of the ORB-SLAM2, JD-SLAM adds another object detecting thread, which is responsible for calling the parameters of the deep network to perform object detection on newly inserted key frames.

We combine the space information to perform the semantic segmentation of the 3D point clouds, which can be splitted into two parts. In the mapping thread, an object detection network is introduced to realize the perception and approximate position prediction of dynamic objects. In the tracking thread, we combine the results of object detector with previous tracking information and the generated map to perform instance-level segmentation on the 3D point clouds and update the probability map.

The system is mainly designed for monocular SLAM, but it can also be transplanted to stereo or RGB-D SLAM. The following article will be based on monocular SLAM. If it is applied to stereo or RGB-D SLAM, just ignore the triangulation step and replace with the depth information obtained by stereo or RGB-D camera.

Object detection

As mentioned earlier, we propose to achieve semantic segmentation with object detection network and spatial information as a replacement of the complex 2D instance segmentation net such as Mask-R-CNN.

To get fast detection speed along with high accuracy, we choose Efficient-Det[20] as the object detection network, which is claimed to achieve state-of-the-art performance in real time. Efficient-Det is trained on COCO data set[21]

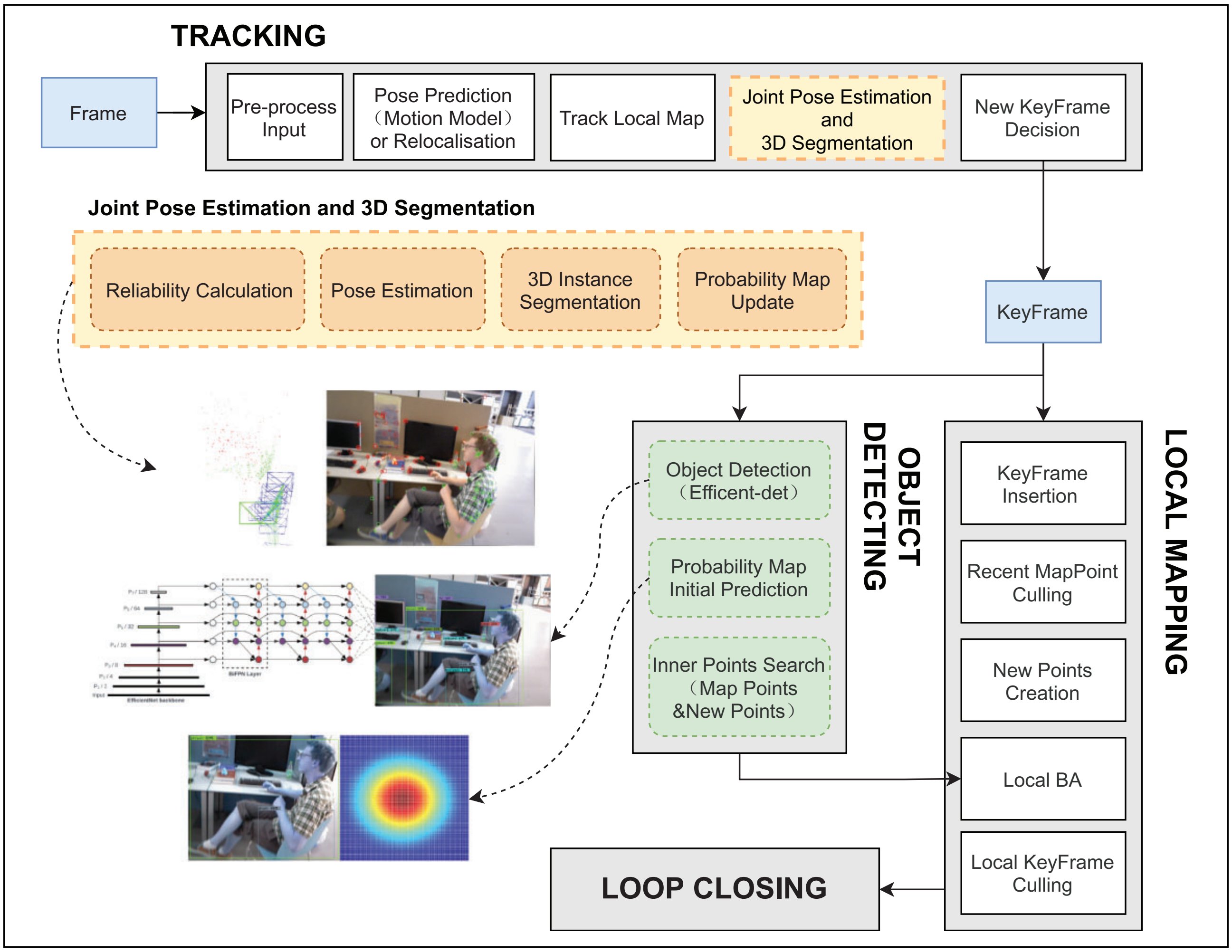

Figure 1. Overview of our JD-SLAM. The system is based on ORB-SLAM2, and the additional modules of JD-SLAM are marked in color. The segmentation of the point cloud is divided into two parts. The object detection runs in parallel with local mapping, and the pose estimation and segmentation are solved in a whole framework in the tracking thread in which the background and each moving objects are modeled as different relative poses. The points cloud can be segmented by hierarchically solving the different motions and searching inliers. It should be noted that the segmentation is not determined by a single track, but by a probability map constructed and updated from multiple observations.

and could segment 80 classes in total. It has seven versions of d0-d7, and the accuracy gradually improves with the speed decreases. The version d0 that we applied could achieve 38.2mAP on COCO 2017 validation data set with only 3.9 M parameters, which is enough to meet our accuracy requirements. The output of Efficient-Det is the object proposals with specified classes in the forms of a bounding box and a confidence score. We focus on several of the classes: "person," "bicycle," "car," "bus," "chair," and "keyboard." The left column of Figure 2 illustrates the result of Efficient-Det-d0. With respect to the amount of calculation, we only perform object detection in key frames.

Semantic probability map

Throughout the system, we maintain a probability map. The map records the probability that a data point comes from an object of interests or an unknown object (collectively processed as background). The probability map stores the label of a feature point and its probability of belonging to the current classification. To prevent overly complex calculations, we only allow each data point to have one possible label. If the point does not belong to any kind of objects of interests, it will be treated as a static point from the static background. In the subsequent estimation of camera poses, we will consider these points with a high possibility to be static to be more reliable and prefer to select these points for pose estimation.

Semantic probability map initialization. Since the object detection network can only provide us with the bounding box of the objects, the feature points in the bounding box cannot be distinguished to come from the object or the background. Given that the depth of the feature points is

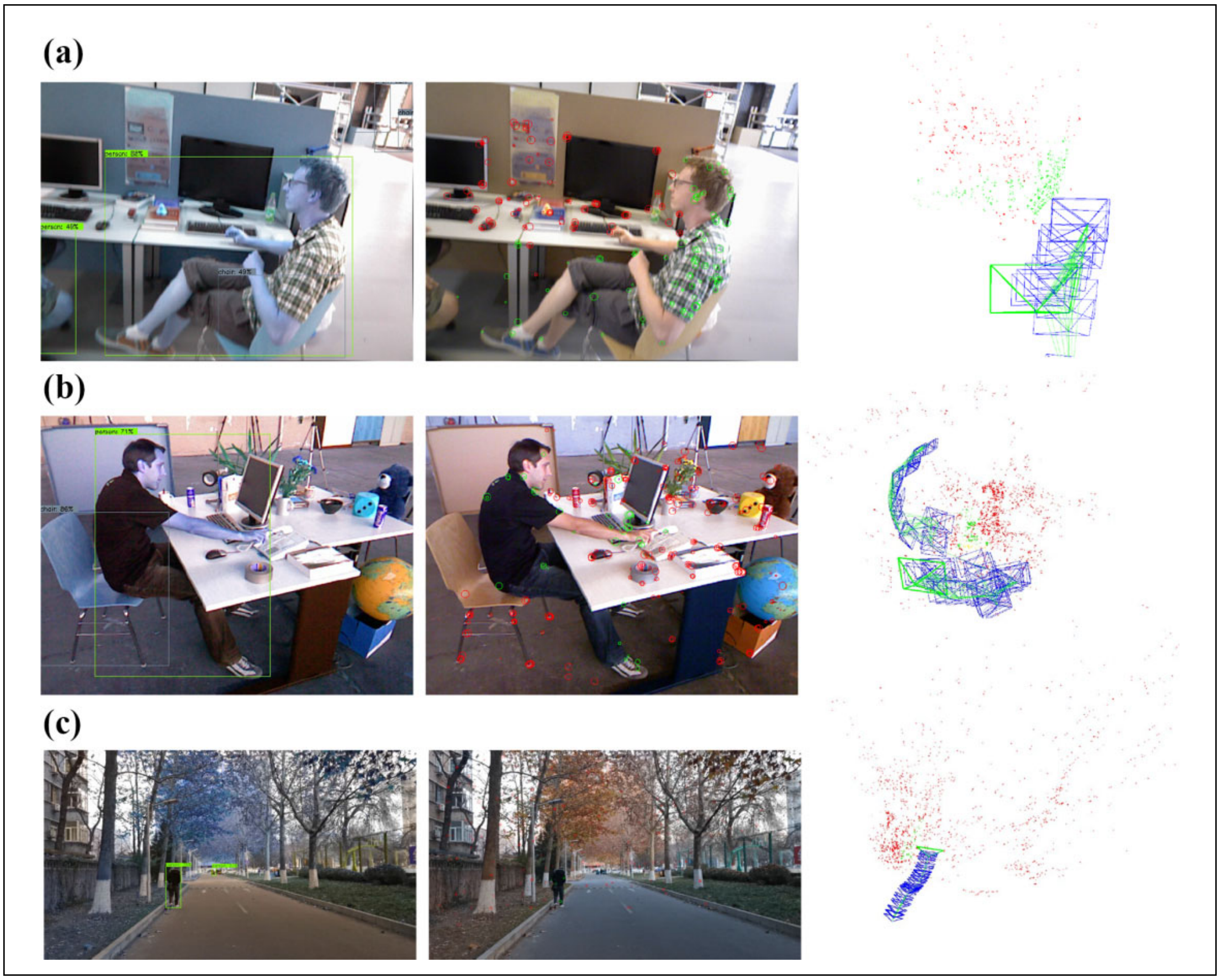

Figure 2. Results of the core steps in the proposed system. Left: Detection results of the Efficient-Det. Middle: Clustering results of the weighted-RANSAC solver. (The size of the circle reflects the probability that the point belongs to the object.) Right: Performance of the 3D point clouds segmentation. (a) fr3/sitting_halfsphere; (b) fr2/_desk_with_person; (c) run in the real-world.

unknown at this time, we cannot make further judgments about the property of these points. To further segment the features in space, we mark all the feature points in the bounding box as potential semantic points. At the same time, an initial probability estimation is given. We believe that the feature points with a small distance from the center point of the bounding box will be more likely to come from that object. Suppose the center point coordinates of the bounding box are $\mathbf{O}_i = \{u_O, v_O\}$, the probability of a feature point $X_i = \{u_i, v_i\}$ comes from the object $\boldsymbol{O}_j$ is

$$\xi_i^j = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp[-\frac{1}{2}(X_i - \mathbf{O}_j)^T \Sigma^{-1}(Xi - \mathbf{O}_j)] \tag{1}$$

where Σ represents the covariance matrix, which is determined by the width w_j and the height h_j of the bounding box of object j

$$\Sigma = \begin{pmatrix} w_j/2 & 0 \\ 0 & h_j/2 \end{pmatrix} \tag{2}$$

If this frame is selected as a keyframe, the new generated points will be added as the candidate map points. In the next several frame tracking, the probability that the point belongs to this object will be calculated and updated continuously from the proposed 3D segmentation framework. If the point can be observed stably and the probability of belonging to the background remains to be greater than a threshold (in this article is 0.85) until a few frames later, the point will be ultimately added to the mature map, and its probability will be continuously verified and updated when it is subsequently observed.

The system has no local map information when it has not been initialized. With respect to the fact that the accuracy of the SLAM system seriously depends on the quality of initialization, we set a more strict condition for initialization. All the feature points in the bounding box of dynamic objects will be eliminated, only the remaining points are used for initialization.

Update the semantic probability map. To update the semantic probability map of existed mature points in the map, we combine the prediction of the CNN object detector with the geometry information from the local map. By projecting the local map into the current frame and checking the consistency of the new prediction from the network, the omission of CNN object detector could be compensated.

When an existing 3D point is observed again in the next frame, its probability will be updated according to the following equation

$$P^t_{\text{object}}(X_i) = (1-\lambda)P^{t-1}_{\text{object}}(X_i) + \lambda P^*_{\text{object}}(X_i) \tag{3}$$

where P^{t-1}_{object} is the origin probability and $P^*_{\text{object}}(X_i)$ is the newly estimated probability from the 3D instance segmentation by the weighted-RANSAC solver, and λ is the factor to adjust the impact of the past tracking and the current tracking.

If this is the first frame after generating the map point, the initial value of $P_{\text{object}}(X_i)$ will be the value of the 2D point which generates this point predicted through the second-order Gaussian distribution.

Joint camera pose estimation and moving object segmentation

As ORB-SLAM2 get an initial estimation of the camera pose by the feature matches from the past key frame, we project both the mature local map and the candidate map points into the current frame to get more 3D-2D correspondences. The correspondences obtained in data association cannot be completely accurate, the mismatches and noise may have a negative impact on subsequent pose estimation. In the scene with many dynamic objects, the assumption of overall rigidity cannot be satisfied, the correspondences from the dynamic objects and those from the static background may conflict when calculating the pose. If those correspondences from the dynamic objects are introduced into the calculation, it actually calculates the camera pose relative to the dynamic object other than the pose relative to the local map. The pose estimation in ORB-SLAM2 is based on nonlinear optimization of PNP estimation, which directly uses the motion of the reference frame as the initial value of the current frame for optimization and iteratively minimizes the reprojection error of the map points. This strategy may make better use of constraints and introduce more information when the data association is stable, while it may instead lead the optimization to a terrible solution when the outlier ratio increases.

A more common approach of pose estimation is to eliminate correspondences from the dynamic objects refer to the results of the segmentation networks and motion consistency detector, and then solve the camera pose within the remaining correspondences by the RANSAC algorithm. RANSAC iteratively searches the parameters of a mathematical model that can fit the most data points from a set of data containing outliers. The standard RANSAC considers that each data point has equal confidence when evaluating the estimated model. Therefore, the number of data elements that the model can fit within a given tolerance is used as the evaluation criteria of model quality. And the model with the most inliers found in iteration will be returned. However, that leads to the fact that the performance of standard RANSAC method will decrease rapidly as the outliers ratio increases. When it is higher than 50%, the optimal solution can hardly be found by standard RANSAC. And it is very sensitive to the threshold boundaries for dividing inliers and outliers. Matching from moving objects in dynamic scenes will further reduce the possibility of RANSAC to find an accurate pose model. Therefore, the method based on ordinary RANSAC is also not suitable for pose estimation in a dynamic environment.

We propose a novel framework with a weighted-RANSAC solver to jointly estimate the camera pose and perform 3D point clouds segmentation, which differs from the usual pose solver based on the ordinary RANSAC in the following terms:

1. Each moving object in the image and the background is modelled as different relative poses, then those motion will be solved from the constructed correspondences hierarchically. The data points that can be fitted by an object's relative pose will be regarded as a part of it. It should be noted that the segmentation is not determined by one observation. It is reflected by a probability map constructed and updated by multiple observations.
2. For different solving objects, different voting weights are assigned to different data points. The weights aim to introduce the prior information from the generated map and semantic segmentation which reflects the reliability of each point for current estimation. That strategy makes the model estimation results tend to be more affected by the parts with a high probability of being inliers and enables a stable and accurate estimation result under the disordered data.

Hierarchically estimation framework. The whole framework is described by Alogrithm 1, where M represents the best model of camera pose found in the iterations, and the rest symbols are consistent with those in section "Notation." It has three steps. The first step aims to initially estimate the pose of the camera relative to the background. By assigning higher dominance to those correspondences with higher reliability in pose calculation, the impact of correspondences from dynamic objects and inferior association can

Algorithm 1. Joint Pose Estimation and 3D Segmentation.

```
Input: $\boldsymbol{O}, \boldsymbol{C}_{\{mature\}}, \boldsymbol{C}_{\{candidate\}}$
 1: $\boldsymbol{O} \leftarrow$ the bounding boxes offered by CNN object detection
 2: $M_{background}, \boldsymbol{C}_{\{inliers\}} \leftarrow$ Weighted_RANSAC$(Q_{C_i}, C_{mature}, \theta)$
 3: while $O_j$ in $\boldsymbol{O}$ do
 4:     $\boldsymbol{C}_j \leftarrow$get correndences of object $O_j$ from $\boldsymbol{C}_{\{candidate\}}$ and $(\boldsymbol{C}_{\{mature\}} - \boldsymbol{C}_{\{inliers\}})$
 5:     $M_j, \boldsymbol{C}^j_{\{inliers\}} \leftarrow$ Weighted_RANSAC$(P^j_{object}, C^j, \theta)$
 6:     $P^*_{object}(X_i) \leftarrow$ update the semantic probability map
 7:     $\boldsymbol{C}_{\{background\}} \leftarrow (\boldsymbol{C}_{\{mature\}} - \boldsymbol{C}^j_{\{inliers\}})$
 8: end while
 9: $M_{background}, \boldsymbol{C}_{\{background\}} \leftarrow$ Nonlinear optimization with $\boldsymbol{C}_{\{inliers\}}$
10: return $M_{background}, \boldsymbol{C}_{\{inliers\}}$ of each object
```

be significantly reduced. The reliability score is a comprehensive response of multiple dimensions including the past tracking quality, multi-view geometric information, and generated point clouds map equipped with the motion probability. The calculation method will be detailed in the following. Here, only the pairs from mature map points are chosen to participate in the calculation. After obtaining the pose, we take all the data points that can be fitted by the current model as the part from the static background $\{C_{\text{background}}\}$.

The second step estimates the motion of each object and segments the point clouds based on the motion clustering. For each object j detected by the Efficient-Det, we denote those correspondences that are extracted from the region in the bounding box and do not belong to the background as $\{C_i^j = (x_i, Xi) | x_i \in S^j_{\text{boundbox}}, C_i^j \notin C_{\text{background}}\}$. We perform the weighted-RANSAC with $\{C_i^j = (x_i, Xi) | x_i \in S^j_{\text{boundbox}}, C_i^j \notin C_{\text{background}}\}$ as the input data and $P^{\text{object}}_{3D}(X_i)$ of the 3D point X_i as the weight to calculate the pose of the camera relative to this object. Meanwhile, the inliers that can be fitted by the estimated pose can be obtained. This process repeats until each object's pose and inliers have been obtained. After the loop finishes, the points which cannot be fitted by any object poses will be treated as a part of the background.

At last, the camera pose of the current frame will be refined by a nonlinear optimization whose elements are all of the points from the background.

The probability of the map points calculated in this process P^*_{object} will be assigned as

$$P^*_{\text{object}}(X_i) = \begin{cases} = 1 & \text{if } C_i \text{ is inliers} \\ = 0 & if\ C_i \text{ is outiers} \end{cases} \tag{4}$$

To speed up tracking, for the objects whose bounding box accounted for less than a certain value of the screen, we just remove all the features in the scope of the bounding box without solving its relative poses. If more than 90% of the feature points in a bound box can be fitted by the pose of background, it is considered that the object is currently in a static state. And features from that object will participate in the pose estimation in tracking, but not be added into the mature map, which aims to make the generated map reusable.

At the same time, in each iteration, we will supplement the missing points according to their 3D spatial distribution. To be exact, if a 3D point without object label is surrounded by three or more points with the same label within a certain distance, then this point will also be marked as coming from this object.

Weighted-RANSAC solver. The core of the proposed framework is the weighted-RANSAC solver, which integrates with an evaluation function with different voting weights and an extra local optimization step. The weighted-RANSAC solver is summarized in Algorithm 2. Here $\mathcal{I}$ is the selected data set of the 3D-2D correspondences. The find_inliers function evaluates the samples with input model M and returns the subset of inliers whose errors are smaller than threshold θ.

To obtain higher robustness, a local optimization step[22] is introduced into the ordinary RANSAC method to optimize the results from coarse to fine. The samples input here are only the ones from $\mathcal{E}_s$ (the inliers to the model found by a minimal solver under a slightly bigger in-out-threshold). As the sampling is running on so-far-inner data, in this iteration we could deploy a nonlinear optimization with small element size as the PNP solver to introduce more information, just like the original intention of ORB-SLAM2. The local optimization step is summarized in Algorithm 3.

Unlike the usual RANSAC, which evaluates the models by the proportion of inliers, we use the sum of the Lagrangian distances between the estimated value and the true value as the criterion. And the score $\mathcal{E}_M$ of a model can be calculated as follows

$$\mathcal{E}_M = \sum_{i=1} \max(\omega \cdot |p_i^m - p_i|, \text{Thr}_{\text{error}}), \tag{5}$$

Algorithm 2. Weighted-RANSAC.

```
Input: $\mathcal{I}, \omega, \theta$
1: $\mathcal{I} \leftarrow$ input samples
2: $\omega \leftarrow$ the weights of input data
3: $\theta \leftarrow$ the inlier-outlier error threshold
4: for $k = 1 \rightarrow \mathrm{K}(\mathcal{I})$ do
5:     $\mathcal{S}_k \leftarrow$ randomly drawn minimal sample from $\mathcal{I}$
6:     $M_k \leftarrow$ model estimated from sample $\mathcal{S}_k$
7:     $\mathcal{E}_k \leftarrow$score_model$(M_k, \omega, \theta)$
8:     if $\mathcal{E}_k > \mathcal{E}_s^*$ then
9:         $M_s^* \leftarrow M_k; \mathcal{E}_s^* \leftarrow \mathcal{E}_k$
10:        $M_{LO}, \mathcal{E}_{LO} \leftarrow$ run Local Optimization $(M_s^*, \omega, \theta)$
11:        if $\mathcal{E}_{LO} > \mathcal{E}^*$ then
12:            $M^* \leftarrow M_{LO}; \mathcal{E}^* \leftarrow \mathcal{E}_{LO}$
13:            update K
14:        end if
15:    end if
16: end for
17: return $M^*$
```

Algorithm 3. Local optimization.

```
Input: $M_s, \omega, \theta, , m_\theta$
1: $M_s$ model estimated by standard RANSAC
2: $\omega \leftarrow$ the weights of input data
3: $\theta \leftarrow$ the inlier-outlier error threshold
4: $m_\theta \leftarrow$ the threshold multiplier
5: $\mathcal{I}_s \leftarrow$ find_inliers $(M_s, \theta)$
6: for $r = 1$ to iters1 do
7:     $\mathcal{S}_{is} \leftarrow$ sample of size $s_{is}$ randomly drawn from $\mathcal{I}_s$
8:     $M_{is} \leftarrow$model estimated from $\mathcal{S}_{is}$ by least squares
       solution
9:     $M' \leftarrow$ model estimated by least squares solution on
       find_inliers $(M_{is}, \theta)$
10:    $\theta' \leftarrow m_\theta \cdot \theta$
11:    for $j = 1$ to iters2 do
12:        $\mathcal{I}' \leftarrow$ find_inliers $(M', \theta')$
13:        $\omega \leftarrow$ computed weights of $\mathcal{I}'$(depend on model)
14:        $M' \leftarrow$ model estimated by least squares solution on
           $\mathcal{I}'$
15:    end for
16:    $\mathcal{E}' \leftarrow$ score_model $(M', \omega, \theta')$
17:    if $\mathcal{E}' > \mathcal{E}'^*$ then
18:        $M_s^* \leftarrow M'$
19:    end if
20:    $M_r \leftarrow$ the best of $M'$
21: end for
22: return the best of $M_r$, with its inliers
```

Where p_i is the real position of an image point, p_i^m represents the reprojected position of p_i using model M, and $\mathrm{Thr}_{\mathrm{error}}$ is a given threshold to limit the impact of a single data point.

Reliability of cornenspondences. We evaluate the reliability Q_{C_i} of a correspondence C_i from three dimensions:

1. Probability of being dynamic:

According to the motion probability of the point calculated in equation (3), if this point has high possibility driven from the background, it will be considered relatively reliable.

2. Motion consistency of adjacent frames detected by optical flow:

Optical flow is a way of describing the motion of individual pixels between images over time. It can serve as an approximation of the physical motion projected onto the image plane. With the assumption that the intensity of a pixel is invariant between two images, the optical flow tracker iteratively searches the correspondence of the source patch in the target image by minimizing the sum of squared difference (SSD) between them. Optical flow method is widely used for object detection and track.

After ORB-SLAM2 obtains the initial pose of the current frame through the assumption model such as uniform motion, the relative translation between the current frame and the previous frame can be solved. By multiplying the last image with the estimated transformation, the warped image can be obtained as a rough estimate of the current frame. We adopt an LK-optical-flow-tracker[23] to search the corresponding position p_i' of the feature point p_i on the warped image. The points with a small Euclidean distance between p_i and p_i' may consider to be reliable and will be given high weight in camera pose estimation.

Since the disparity between the two adjacent frames is small, we can assume that the camera motion contains only rotation and no translation, so the transformation between the two images can be approximately described by a homography matrix H

$$H = KRK^{-1} \tag{6}$$

Figure 3 shows the detection results of motion consistency on the TUM RGB-D data set. It can be seen that although it has some deviations, the detected motion probability is approximately consistent with the actual situation.

3. Continuous observability in tracking:

We use the past recurrence rate of a map point ω as an index to evaluate the reliability of the map point. The recurrence rate ω is given by

$$\omega = N'/N \tag{7}$$

where N is the number of past frames at which the point can be observed and N' is the number of the times that the map point and its corresponding 2D point stay within inliers after a local bundle adjustment.

In general, the score of a 3D-2D cornenspondence C_i can be calculated as follows

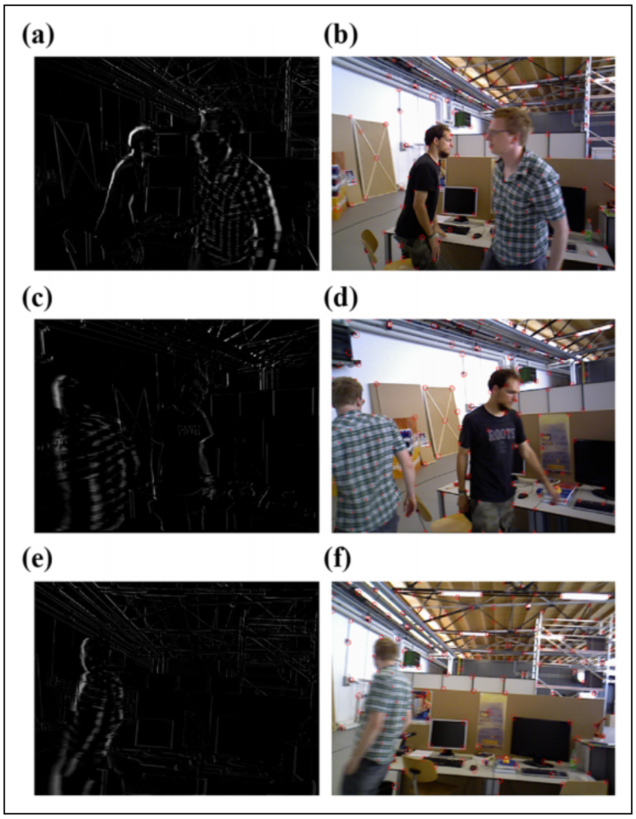

Figure 3. Results of the motion consistency detection of adjacent frames by optical flow. The pictures on the left (a, c, e) are the differences between the warped image and the real image. The pictures on the right (b, d, f) are the visualized motion probability $P_{\text{motion}}(x_i)$ obtained from the Euclidean distance of optical flow. Bigger circles represent lower probability of moving while smaller circles mean higher probability of moving.

$$Q(C_i) = \alpha f(P_{\text{object}}(X_i)) + \gamma f(P_{\text{Observed}}(X_i)) + \beta f(P_{\text{motion}}(x_i)) \quad (8)$$

where the $P_{\text{object}}(X_i)$ represents the probability of being dynamic, $P_{\text{Observed}}(X_i)$ is a reflect of the recurrence rate ω, $P_{\text{motion}}(x_i)$ is the score of motion consistency detection by optical flow, and α, β, γ are the impact factors to adjust the proportion of the three dimensions. As Figure 4 shows, our framework with the improved RANSAC can split data points with consistent motion more accurately than the standard RANSAC.

Experiment

To demonstrate the robustness and real-time performance of JD-SLAM, we integrate it to ORB-SLAM2 and operate experiments on a public data set and in a real-world environment. Figure 2 shows the performance of the core modules of our method. The dynamic objects and the background are represented in green and red, respectively, and the different circle sizes represent the probability that the point belongs to the current object. The images in Figure 2(c) are captured by the camera equipped by HUAWEI V20 with the resolution of 1920 $\times$ 1080. All experiments are carried out with an Intel Core i7-8750 H CPU (12 cores@ 2.20 GHz) and 32 GB RAM. We compare our system with the ordinary ORB-SLAM2 and Dyna-SLAM (a state-of-the-art work of dynamic SLAM) and provide qualitative and quantitative experimental results.

Evaluation of accuracy

The proposed method is integrated into ORB-SLAM2. We evaluate the improved system on the TUM RGB-D data set[24] and the KITTI odometry data set.[25] The TUM RGB-D data set contains dozens of indoor sequences captured by an RGB-D camera. Among these sequences, several of them are classified as “dynamic objects” and widely used for evaluating the SLAM system for dynamic scenes. The KITTI data set contains outdoor sequences recorded from a moving car on urban roads. Although it does not specifically classify scenes containing dynamic objects, we selected several sequences that contain moving vehicles for experiment. Both of the selected data sets provide the ground truth of the camera pose for accuracy evaluation. Since JD-SLAM is mainly designed for monocular SLAM, all of the experiments are run with the monocular mode.

Along with comparing the JD-SLAM against the original ORB-SLAM2, the Dyna-SLAM is chosen as a comparison since it is also built on the ORB-SLAM2 and is the state-of-the-art method of SLAM in dynamic scenes. The experimental results of them are obtained by running the authors’ open-source codes. Accounting for the randomness of multi-threading system, we run each sequence for five times and pick the median result. For comprehensive result, the camera poses of all frames are recorded in accuracy evaluation.

We adopt absolute trajectory error (ATE) and relative pose error (RPE) to conduct the quantitative evaluation of the system. The ATE directly calculates the difference between the ground truth and the estimated camera poses, which reflects the overall performance of system. And the RPE describes the pose difference of two frames during a fixed time interval. It reflects the drift of the system. The root-mean-square error of ATE and RPE is shown in Table 1, which is calculated by the benchmark tool.[26]

It can be seen in Table 1 that our system outperforms ORB-SLAM2 and Dyna-SLAM on most of the sequences and indexes, especially on the TUM-RGBD data set. As the proportion of dynamic objects occupying the screen is not large, the improvement on the KITTI data set is not as obvious as that on the TUM-RGBD data set. Some sequences in KITTI data set contain many vehicles parked on the roadside. These vehicles will be cut out in Dyna-SLAM, which reduces the available information and makes its accuracy lower than the original ORB-SLAM2, such as on sequence 03. We retain those points in tracking and

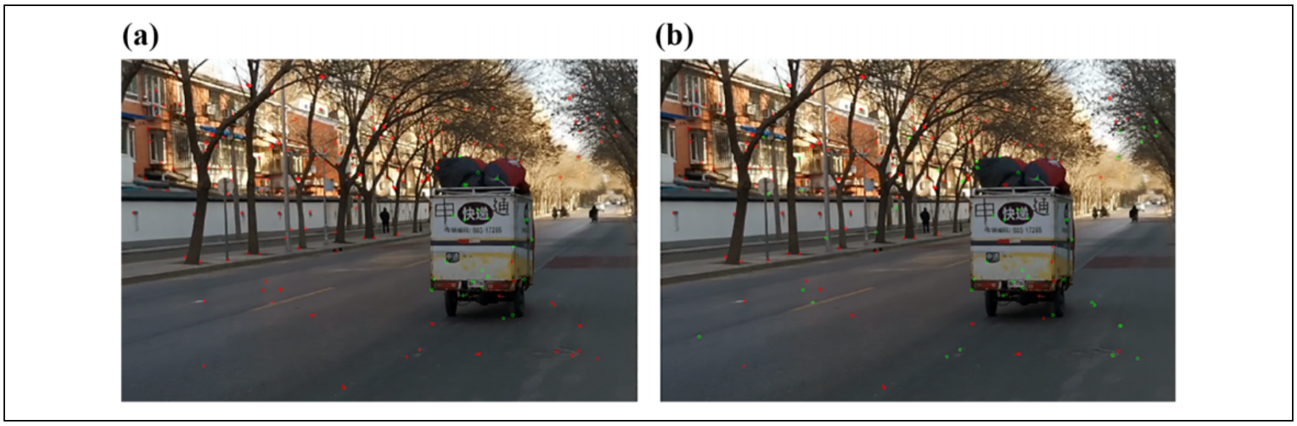

Figure 4. Comparison of the results of the standard RANSAC and our method in segmenting a tricycle in motion. The red points represent the background and the green points represent the moving tricycle. (a) Our method and (b) standard RANSAC.

Table 1. Localization error comparison of each frame for JD-SLAM, ORB-SLAM2, and Dyna-SLAM.

		Original ORB-SLAM2		JD-SLAM		Dyna-SLAM	
	Sequence	ATE (m)	RPE(deg/m)	ATE (m)	RPE(deg/m)	ATE (m)	RPE(deg/m)
TUM	fr2/desk_with_person	0.0344	0.0072	**0.0309**	**0.0071**	0.0365	0.0077
	fr3/sitting_xyz	0.0268	0.0079	**0.0223**	**0.0075**	0.0436	0.0085
	fr3/sitting_halfsphere	0.0217	0.0102	**0.0175**	**0.0823**	0.0554	0.0115
	fr3/walking_rpy	0.8814	0.0181	**0.0616**	0.0294	0.0665	**0.0127**
	fr3/walking_halfsphere	0.1406	0.0118	**0.1074**	**0.0092**	0.1269	0.0115
KITTI	01	497.8152	3.5468	**433.4713**	**2.4713**	458.7045	2.7549
	03	1.5664	0.0400	**1.3010**	**0.0245**	2.0246	0.0258
	04	0.9752	0.0564	**0.9288**	**0.0271**	0.9870	0.0301
	08	55.1200	0.3203	48.2557	0.3018	**44.6520**	**0.2664**
	09	44.6347	**0.4810**	**41.0825**	1.2035	47.7973	3.1072

*Bold data marks the best indicators in horizontal comparison.

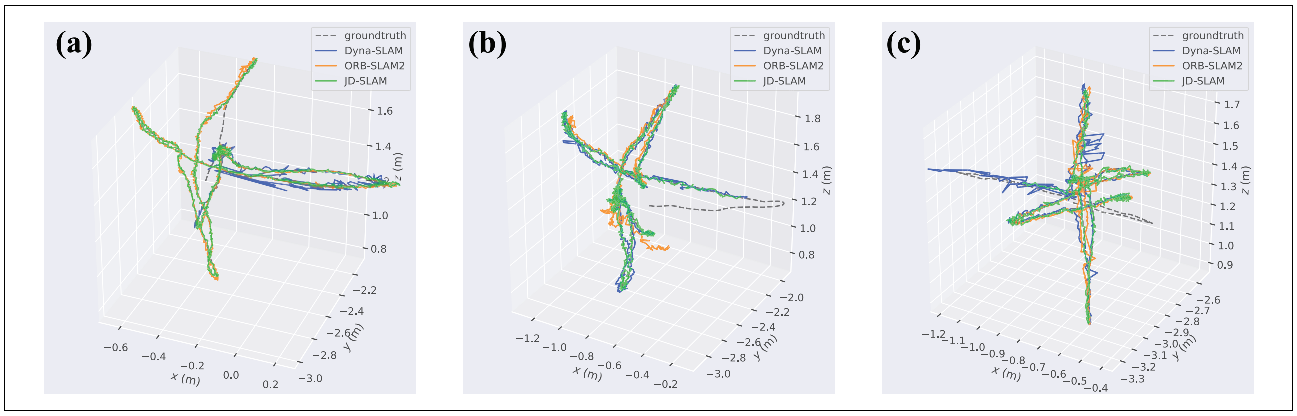

Figure 5. Estimated trajectories by JD-SLAM, ORB-SLAM2, and Dyna-SLAM on TUM RGB-D data set. (a) Sitting halfsphere; (b) walking halfsphere; (c) walking xyz.

avoid such problem. But on those sequences with complicated road conditions and numerous passing vehicles such as sequence 08, our method may be confused due to the interlacing of bounding boxes, while Dyna-SLAM uses Mask-RCNN to accurately segment each vehicle and achieves higher accuracy than our method. But our method still performs better than the original ORB-SLAM2.

Figure 5 shows the estimated trajectories by our method, ORB-SLAM2, and Dyna-SLAM on TUM RGB-D data set. It can be seen clearly that our estimated trajectories are

Table 2. Time consumption of each module of JD-SLAM.

Thread	Mapping	Tracking	
Function	Object detection	3D segmentation	Pose estimation
s_xyz	220 ± 20 ms	45 ± 5 ms	20 ± 2ms
w_halfsphere	190 ± 20 ms	40 ± 5 ms	22 ± 2 ms

Table 3. Tracking time cost of each frame on the TUM RGB-D dataset.

	Tracking time cost (ms)
ORB-SLAM2	35 ± 10
JD-SLAM	60 ± 10
Dyna-SIAM	2000 ± 500

much closer to the ground truth. Above quantitative comparison indicate that our method yields significant boosts in terms of robustness and accuracy, especially in the sequences of high-dynamic environments. That benefits from the object detection combined with the subsequent points cloud segmentation, which eliminate the possible dynamic data points and give an accurate pose estimation. Meanwhile, the motion consistency test by the optical flow and the observation quality check further provides reliable filtering for data.

Evaluation of real-time performance

Running time is an important factor of the performance of the online system. We test the time cost of camera pose tracking. Table 2 shows the time cost of the core modules of 3D segmentation in JD-SLAM. As a comparison item, Dyna-SLAM often takes more than 2 s just in the 2D instance segmentation step. And Table 3 shows the time cost comparisons of our camera pose tracking method with the ordinary ORB-SLAM2 and Dyna-SLAM on the fr3/walking–halfsphere.

In the experiment, the operating speed of JD-SLAM basically maintains at 22 ± 5 fps (on a single-core CPU), which achieves real-time performance in most of the test sequences. Since the JD-SLAM avoids the 2D segmentation, and the adopted network has a small amount of parameters, it gains a huge advantage in speed over the Dyna-SLAM. Also, that makes JD-SLAM suitable for running on small devices with limited computing power.

Conclusion

In this article, we present JD-SLAM, a novel method to improve the performance of the monocular SLAM system in a dynamic environment while focusing on the efficiency-accuracy balance of the system. To make better use of semantic information and geometric constraint information, instead of directly segmenting 2D images, we cluster the point clouds in space with respect to the motion distribution and the object detection result. To this end, we propose a hybrid framework based on an improved weighted-RANSAC solver, which makes estimating the camera pose and solving the object motion into a whole. At the same time, the weight distribution based on the motion probability and tracking quality and the introduction of the local optimization module in RANSAC could further improve the robustness of the system. Experiments show that the proposed system has superior performances in terms of accuracy and speed than state-of-the-art methods. And it is proved that the system can run in real time on CPU with a single core.

In the future, we will try to use the results of 3D point clouds segmentation to optimize the performance of the target detection network and explore the possibility of performing SLAM and objection tracking at the same time.

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 by the National Natural Science Foundation of China No. 61901436

ORCID iDs

Yujia Zhai https://orcid.org/0000-0001-9729-484X
Baoli Lu https://orcid.org/0000-0001-8294-3421

References

1. He K, Gkioxari G, Dollár P, et al. Mask R-CNN. In: *Proceedings of the IEEE international conference on computer vision, 2017*, Venice, Italy, 22–29 October 2017, pp. 2961–2969.
2. Long J, Shelhamer E and Darrell T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition, 2015*, Boston, MA, USA, 7–12 June 2015, pp. 3431–3440.
3. Bescos B, Fácil JM, Civera J, et al. DynaSLAM: tracking, mapping, and inpainting in dynamic scenes. *IEEE Robot Autom Lett* 2018; 3(4): 4076–4083.
4. Runz M, Buffier M and Agapito L. Maskfusion: real-time recognition, tracking and reconstruction of multiple moving objects. In: *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Munich, Germany, 16–20 October 2018, pp. 10–20. IEEE.

5. Zhang T, Zhang H, Li Y, et al. FlowFusion: dynamic dense RGB-D SLAM based on optical flow. *arXiv preprint* arXiv: 2003.05102. 2020 Mar 11.
6. Yu C, Liu Z, Liu XJ, et al. DS-SLAM: a semantic visual SLAM towards dynamic environments. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Piscataway, NJ: IEEE, 1 October 2018, pp. 1168–1174.
7. Salas-Moreno RF, Newcombe RA, Strasdat H, et al. Slam++: simultaneous localisation and mapping at the level of objects. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Portland, OR, USA, 23–28 June 2013, pp. 1352–1359.
8. Liu H, Zhang G and Bao H. Robust keyframe-based monocular SLAM for augmented reality. In: *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Merida, Mexico, 19–23 September 2016, pp. 1–10. IEEE.
9. Zhang J, Henein M, Mahony R, et al. VDO-SLAM: a visual dynamic object-aware SLAM system. *arXiv preprint* arXiv: 2005.11052. 2020 May 22.
10. 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.
11. Saputra MR, Markham A and Trigoni N. Visual SLAM and structure from motion in dynamic environments: a survey. *ACM Comput Surv* 2018; 51(2): 1–36.
12. Zhong F, Wang S, Zhang Z, et al. Detect-slam: making object detection and slam mutually beneficial. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Tahoe, NV, USA, 12–15 March 2018, pp. 1001–1010. IEEE.
13. Kaneko M, Iwami K, Ogawa T, et al. Mask-SLAM: robust feature-based monocular SLAM by masking using semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2018*, Salt Lake City, Utah, USA, 18–22 June 2018, pp. 258–266.
14. Zhang J, Gui M, Wang Q, et al. Hierarchical topic model based object association for semantic SLAM. *IEEE Trans Vis Comput Graph* 2019; 25: 3052–3062.
15. Zhou W, Peng X, Wang H, et al. Nonparametric statistical and clustering based RGB-D dense visual odometry in a dynamic environment. *3D Res* 2019; 10(2): 11.
16. Sünderhauf N, Pham TT, Latif Y, et al. Meaningful maps with object-oriented semantic mapping. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Vancouver, BC, Canada, 24–28 September 2017, pp. 5079–5085. IEEE.
17. Pham TT, Eich M, Reid I, et al. Geometrically consistent plane extraction for dense indoor 3D maps segmentation. In: *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Daejeon, South Korea, 9–14 October 2016, pp. 4199–4204. IEEE.
18. Horn BK and Schunck BG. Determining optical flow. In: Pearson JJ (ed) *Techniques and Applications of Image Understanding*. Bellingham, WA: International Society for Optics and Photonics, 12 November 1981, Vol. 281, pp. 319–331.
19. Dosovitskiy A, Fischer P, Ilg E, et al. Flownet: learning optical flow with convolutional networks. In: *Proceedings of the IEEE international conference on computer vision*, Santiago, Chile, 7-13 December 2015, pp. 2758–2766.
20. Tan M, Pang R and Le QV. EfficientDet: scalable and efficient object detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Seattle, WA, USA, 13–19 June 2020, pp. 10781–10790.
21. Lin TY, Maire M, Belongie S, et al. Microsoft COCO: common objects in context. In: *European conference on computer vision*, Zurich, Switzerland, 6–12 September 2014, pp. 740–755. Cham: Springer.
22. Lebeda K, Matas J and Chum O. Fixing the locally optimized RANSAC-full experimental evaluation. In: *British machine vision conference*, Surrey, UK, 3–7 September, 2012, Vol. 2.
23. Lucas BD and Kanade T. An iterative image registration technique with an application to stereo vision. 1981.
24. Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems. In: *2012 IEEE/RSJ international conference on intelligent robots and systems*, Vilamoura, Portugal, 7–12 October 2012, pp. 573–580. IEEE.
25. Geiger A, Lenz P and Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, 16–21 June 2012, pp. 3354–3361. IEEE.
26. Zhang Z and Scaramuzza D. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In: *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Madrid, Spain, 1–5 October 2018, pp. 7244–7251. IEEE.