Received July 6, 2021, accepted July 24, 2021, date of publication July 26, 2021, date of current version August 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3100426

RDMO-SLAM: Real-Time Visual SLAM for Dynamic Environments Using Semantic Label Prediction With Optical Flow

YUBAO LIU AND JUN MIURA, (Member, IEEE)
Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi, Aichi 441-8580, Japan
Corresponding author: Yubao Liu (yubao.liu.ra@tut.jp)

This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 17H01799.

ABSTRACT Visual simultaneous localization and mapping (vSLAM) are considered a fundamental technology for augmented reality and intelligent mobile robots. However, rigid scene assumption is common in vSLAM, which limits the wide usage in populated real-world environments. Recently, with the widespread use of artificial neural networks, many solutions have tried to eliminate the influence of dynamic objects using semantic information provided by object detection or semantic segmentation. Mask R-CNN is popular in many applications, but is usually slow and limits the speed of vSLAM because it waits for the semantic results before camera ego-motion estimation. We had previously introduced a real-time vSLAM, RDS-SLAM, which isolates tracking and semantic segmentation by adding a semantic thread and moving probability estimation. However, Mask R-CNN only supplies a small amount of semantic information because only a few keyframes can be segmented within a short time. Therefore, in this study, we propose a novel vSLAM, RDMO-SLAM, which can leverage more semantic information while ensuring the real-time nature by adding semantic label prediction using dense optical flow. Besides, we also estimate the velocity of each landmark and use them as constraints to reduce the influence of dynamic objects in tracking. Demonstrations are presented, which compare the proposed method to comparable state-of-the-art approaches using dynamic sequences. We improved the real-time performance from 15 Hz (RDS-SLAM) to 30 Hz while keeping robust tracking in dynamic scenes.

INDEX TERMS Visual SLAM, semantic segmentation, real-time, dynamic environments, RDS-SLAM.

I. INTRODUCTION

Visual simultaneous localization and mapping (vSLAM) has been a hot research topic in computer vision, augmented reality (AR), unmanned autonomous vehicles, and robotics. vSLAM [1] is a fundamental technology for estimating the pose of sensors and reconstructing structures in an unknown environment using onboard sensors, such as mono, RGB-D, and stereo cameras. vSLAM can be classified into feature based approaches, such as ORB-SLAM [2] and RGB-D SLAM [3], and direct approaches, such as LSD-SLAM [4] and DSO [5]. As we know, there is usually a strong assumption in vSLAM, the rigid scene assumption. vSLAM assumes that the camera is the only moving object. However, the camera is not the only moving object in the real environment, and this assumption may result in unstable tracking or even tracking failure. For instance, humans are dynamic objects in indoor environments. The motion of features or points on the dynamic objects is unknown and cannot be accurately estimated. Dynamic objects may influence feature matching and BA (bundle adjustment) and eventually resulting in non-robust pose estimation and map building. As shown in Fig. 1, vSLAM by default cannot sense the motion of dynamic objects or dynamic features, e.g., m^j_{t-1} moved to m^j_t. Unfortunately, it still use the old map point m^j_{t-1} and its observed features $x_{t-1,j}$ and $x_{t,j}$ to estimate the pose using BA by minimizing the reprojection error for every selected features. The problem is that the newly observed feature, $x_{t,j}$, no longer matches m^j_{t-1} but matches a new landmark m^j_t that is unknown for vSLAM. This phenomenon can be somehow detected and reduced using geometric algorithms such as RANSAC (random sample consensus) [6] and BA if

The associate editor coordinating the review of this manuscript and approving it for publication was Ehsan Asadi.

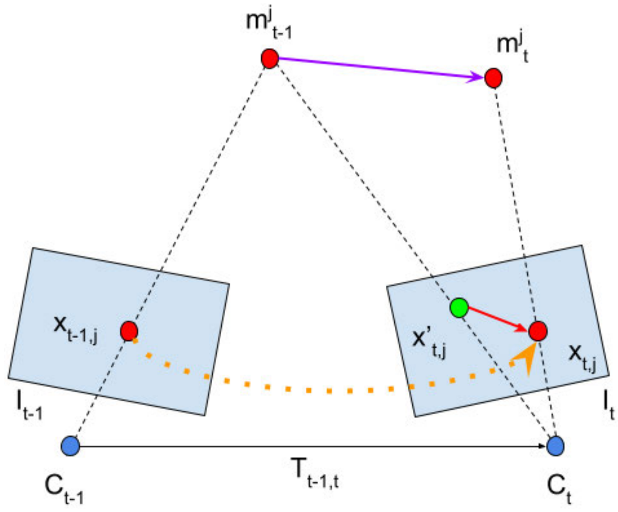


FIGURE 1. Rigid scene problem in vSLAM. The j-th map point m^j_{t-1} on a dynamic object matched with the feature $x_{t-1,j}$ in the previous image. Assume that this map point moved to a new position m^j_t and is observed as $x_{t,j}$ in the current image. $x'_{t,j}$ is the position if the feature is static. C_{t-1} and C_t are the camera centers of previous (I_{t-1}) and current (I_t) images.

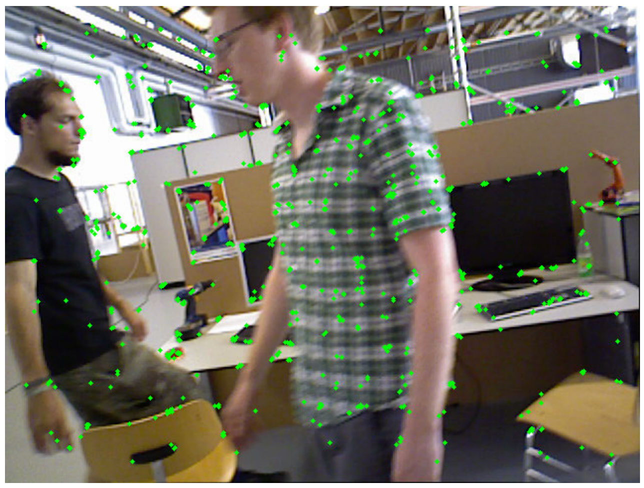


FIGURE 2. Example of dynamic environments (TUM).

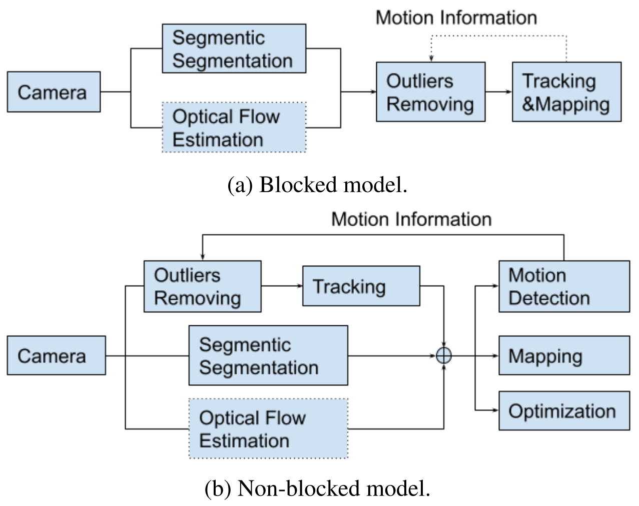


FIGURE 3. Blocked model vs non-blocked model. The optical flow model is optional. The semantic model can be Mask R-CNN, SegNet, SSD or others. Segmentation and optical flow can run in parallel for frames or keyframes. The feedback, motion information, is optional for the blocked model.

dynamic features move very fast. However, outliers cannot be efficiently detected if they move slowly or occupy a major part of the scene. As shown in Fig. 2, one person is walking slowly at the center of the image, and many features are detected on his T-shirts. vSLAM may mistakenly trust these features and result in non-robust pose estimation because these features are unstable.

In recent years, vSLAM is useful in scene understanding, semantic mapping, robot navigation, and decision making with the aid of object detection, convolutional neural network (CNN), deep learning, and machine learning. The benefits between these semantic applications/systems [7] and vSLAM are mutual. VSLAM can support pose estimation and map building for these semantic applications (*SLAM helps semantic*), whereas the semantic methods can be applied to improve tracking performance (*semantic helps SLAM*). Many studies try to eliminate or reduce the influence of dynamic objects using various segmentation methods. For example, Detect-SLAM [8] uses the object detection (SSD [9]) approach to improve tracking performance; similarly, DynaSLAM [10] and DM-SLAM [11] use Mask R-CNN [12], DS-SLAM [13] uses SegNet [14], and KMOP [15] uses Open-Pose [16] and k-means [17].

To the best of our knowledge, the execution speeds of these methods are limited by the semantic model used, e.g., Mask R-CNN, SSD, and OpenPose, because these methods need to wait for the *semantic result/information*, e.g., label, bounding box, before tracking. Such an architecture is called a *blocked model*, as shown in Fig. 3 (a). In our previous study, we proposed a novel real-time vSLAM architecture, RDS-SLAM [18], validated using both Mask R-CNN and SegNet using *non-blocked model*, as shown in Fig. 3 (b). RDS-SLAM can guarantee the speed of tracking free from speed limitation of semantic models using multi-thread and moving probability estimation of each map point. However, it has some shortcomings: a) Mask R-CNN is slow (approximately 200 ms) and cannot segment every keyframe to use more semantic information in a limited time. It cannot run stably at 30 Hz using the TUM [19] dataset because insufficient semantic information obtained at 30 Hz for some of the dynamic scenes of the TUM dataset, which are very short, only about half a minute; b) only predefined objects trained by Mask R-CNN are handled. In this study, we try to ensure real-time performance (30 Hz) while keeping robust tracking by exploiting optical flow.

One critical challenge of using semantic information is time complexity. Although there are some lightweight semantic segmentation models, e.g., SegNet, the total time of tracking one frame is still more than the original ORB-SLAM3.

Besides, sometimes a complex CNN architecture is required for robots to perform high-level tasks, e.g., human-robot interaction and semantic mapping. To acquire more semantic information in a limited time, we use dense optical flow for each pixel to predict the semantic label of Mask R-CNN. Another challenge is that only predefined objects trained by CNN are used to judged outliers. Optical flow can estimate the pattern of motion for every feature, including features on the undefined objects. The velocity of map points can be estimated by optical flow and used as a constraint to reduce the influence of outliers from the tracking process.

In this study, we propose a semantic label prediction algorithm to generate more semantic information for the keyframes that are not segmented. Real-time tracking under dynamic environments is achieved while keeping robust tracking using a heavy CNN architecture. Besides, the velocity of landmarks is estimated with the aid of scene flow and Kalman filter. These two constraints (the semantic label and velocity) can reduce the influence of dynamic objects in vSLAM.

The main contributions of this paper are as follows.

(1) We propose a novel semantic-based real-time vSLAM algorithm using Mask R-CNN and PWC-Net for dynamic environments, RDMO-SLAM, an extension of RDS-SLAM, which can achieve both good tracking performance and the real-time nature.

(2) We predict the semantic result of Mask R-CNN using optical flow to obtain more semantic information so that the tracking thread uses as much semantic information as possible.

(3) We demonstrate the real-time performance (30 Hz) under dynamic environments using the TUM dataset and an AR demo in the case of using a heavy CNN architecture, Mask R-CNN.

The rest of this article is structured as follows. Section II presents related work. Sections III - VIII detail the implementation of the proposed method. Section IX shows the experiment results. Finally, Section X presents conclusions and discusses future work.

II. RELATED WORK

In this section, we explore related works on vSLAM and state-of-the-art solutions to the rigid scene assumption in vSLAM under dynamic environments. We classify the methods into purely geometric, reconstruction, and semantic-based approaches. The semantic-based approaches leverage semantic information to detect and segment objects and remove outliers from tracking. Notably, these approaches may share some common ideas, such as the use of geometric checking.

A. VISUAL SLAM

Feature-based methods rely on salient point matching and can only perform a sparse reconstruction. Parallel tracking and mapping (PTAM) [20] that implements a keyframe-based monocular SLAM system on a cell phone is a promising platform for hand-held AR. ORB-SLAM [2], a monocular vSLAM that estimates camera ego-motion by matching ORB [21] features extends the versatility of PTAM to environments that are intractable for PTAM. Based on ORB-SLAM, ORB-SLAM2 [22], a complete SLAM system for monocular, stereo, and RGB-D camera was presented, which can work in real-time in various environments. Carlos *et al.* proposed the latest version of ORB-SLAM, ORB-SLAM3 [23], which tightly integrates visual and inertial information and adds a multiple map system (ATLAS [24]). Our previous work, a real-time dynamic SLAM using semantic segmentation methods (RDS-SLAM) [18], is implemented based on ORB-SLAM3. It adds a novel semantic tracking thread that leverages semantic information to improve tracking accuracy and retain the real-time property in dynamic environments. This study extends RDS-SLAM by adding two more threads (optical flow and velocity estimation) to generate more semantic information.

Apart from feature-based methods, many direct vSLAM approaches [4], [5], [25], [26], which can estimate, in principle, a completely dense reconstruction by the direct minimizing of the photometric error, have also been proposed. For example, Kerl *et al.* proposed a dense visual SLAM method, DVO (Dense Visual SLAM) [26], using an RGB-D camera, which minimizes both photometric and depth error over all pixels.

However, rigid scene assumption is a common problem for both the feature-based and the direct methods. Detecting and handling outliers in real-time is challenging in vSLAM. Although there are some strategies, such as selecting relative good features and RANSAC-based checking, in some vSLAMs, e.g., ORB-SLAM3. However, they are still not well suitable for dynamic environments.

B. GEOMETRIC-BASED APPROACHES

Li and Lee [27] proposed a depth edge-based RGB-D SLAM system for dynamic environments based on the frame-to-keyframe registration, which only uses weighted depth edge points. Sun *et al.* [28] proposed a novel online RGB-D data-based motion removal approach that uses optical flow. It is integrated with the front end of an RGB-D SLAM system, acting as a preprocessing stage to filter data associated with dynamic objects. They also proposed a monocular vSLAM algorithm [29] that uses optical flow to improve tracking performance with a monocular camera in dynamic environments. Also, they integrated their method into ORB-SLAM. However, their methods have some limitations. For example, the threshold used to distinguish dynamic points is set to a fixed value, which may not be an optimal value for some sequences. Kim *et al.* [30] proposed an IMU-based solution. They classified the features into dynamic and static using the IMU rotation component between two consecutive images. However, for many use cases, it is desirable to improve the accuracy of pose tracking and map building using only a single camera. Besides, IMU has drift and accumulated errors over time. Sun *et al.* [31] classified pixels using the segmentation of quantized depth images and calculated the difference in intensity between consecutive RGB images. Tan *et al.* [32] proposed a novel online keyframe representation and updating method to adaptively model the dynamic environments, where an appearance or a structure change could be effectively detected and handled. Although geometric-based methods can eliminate outliers to some extent, there is room for further optimization of tracking performance using semantic information.

C. RECONSTRUCTION-BASED APPROACHES

Visual odometry and scene flow (VO-SF) [33], an odometry-based method designed for dynamic scenes proposed by Jaimez *et al.*, combines visual odometry, k-means, and scene flow and reconstructs a 3D model of the rigid scene. Co-Fusion [34] proposed an approach to track and reconstruct multiple moving objects using SharpMask [35]. BaMVO [36] proposed a background model-based visual odometry. StaticFusion [37], a method for dense RGB-D SLAM proposed by Raluca *et al.*, tried to address the rigid scene assumption by jointly estimating the motion of an RGB-D camera and segmenting the scene into static and dynamic parts. StaticFusion is conceptually related to BaMVO but uses a frame-model alignment instead of a multi-frame strategy. Similar to ElasticFusion [38], camera tracking is performed by aligning incoming frames with a dense surfel-based model of the environment. A background model that fuses only the static elements by decoupling the static and dynamic parts is built. K-means is used to segment geometric clusters in StaticFusion, and it is difficult to find the optimal K value for a specific scene. This problem also exists in other k-means-based algorithms, such as KMOP-vSLAM [15]. Also, StaticFusion assumes each cluster is a rigid body to reduce the overall computational complexity, and then solves the static/dynamic segmentation problem cluster-wise as opposed to pixel-wise. Moving people are not rigid bodies in many cases. We do not use such an assumption because we focus on improving the tracking accuracy by eliminating outliers both on rigid objects and dynamic objects in real-time rather than focusing on building a conservative reconstruction of the static structures of the scene.

In this study, the camera pose is estimated using sparse ORB features because it is usually more lightweight than the dense RGB-D SLAM, and we do not build the dense surfel-based model.

D. SEMANTIC-BASED APPROACHES

We classify the semantic-based methods into the blocked and non-blocked models. As shown in Fig. 3, in the blocked model, the semantic information needs to be obtained before the tracking, which limits the real-time performance. We have proposed a non-blocked model in RDS-SLAM [18], where the tracking is not blocked to wait for semantic information, and the camera pose is optimized after obtaining semantic information.

1) BLOCKED MODEL-BASED APPROACHES

DynaSLAM [10], based on ORB-SLAM2 and Mask R-CNN has capabilities of dynamic object detection and background inpainting, which can detect dynamic objects either by multi-view geometry, deep learning, or both and then reconstructs frame backgrounds occluded by dynamic objects using a rigid scene map. DP-SLAM [39] combines the results of geometry constraints and Mask R-CNN to track the dynamic key points in a Bayesian probability estimation framework. DP-SLAM was integrated into the front-end of the ORB-SLAM2 to inpaint frame background occluded by the detected dynamic objects. KMOP-vSLAM [15], also implemented on ORB-SLAM2, has capabilities of unsupervised learning segmentation (k-means [17]) and human detection (OpenPose [16]) for robust tracking in dynamic environments. Outliers belonging to dynamic objects are detected and eliminated from tracking. One problem is that the number of clusters of k-means is given manually and it may not be optimal for the current environment. DS-SLAM [13], based on ORB-SLAM2 and SegNet [14], uses a moving consistency check to reduce the impact of dynamic objects by assuming that feature points on people are most likely to be outliers. Detect-SLAM [8], based on ORB-SLAM2 and SSD [9], classifies keypoints into four states: low-confidence static, high-confidence static, low-confidence dynamic, and high-confidence dynamic. It only detects keyframes to save time and then insert the keyframes into the local map and update the moving probability into the local map. DM-SLAM [11], also based on ORB-SLAM2, employs Mask R-CNN, optical flow, and epipolar constraints to judge outliers. It uses features in dynamic objects if they are not moving very fast to reduce the feature-scarce cases that may happen by eliminating the features on dynamic objects. Fan *et al.* [40] proposed a novel semantic SLAM system with a more accurate point cloud map in dynamic environments by exploiting ORB-SLAM2 and BlizNet [41].

Most existing algorithms operating in complex dynamic environments simplify problems by eliminating dynamic objects from tracking or tracking them separately. However, VDO-SLAM [42] presented a novel formulation to model dynamic scenes in a unified estimation framework over robot poses, static and dynamic 3D points, and object motions. DynaSLAM II [43] is a similar work with VOD-SLAM that tracks multiple rigid objects such as cars and bicycles. According to the data provided in their papers, DynaSLAM II is a little faster and more robust than DVO-SLAM if not considering the time complexity of semantic segmentation. However, these methods only work for rigid objects and neither of them is suitable for indoor dynamic environments where people are the major dynamic objects. For example, In the dynamic scene of the TUM [19] dataset, people change their shape sometimes by standing or sitting. Besides, these methods are not real-time because the semantic segmentation and optical flow information need to be prepared beforehand.

All the methods that use the blocked model wait for the semantic results of each frame or keyframe before estimating the camera pose, thereby resulting in their processing speed being limited by the segmentation method used. To further clarify this, we compared the tracking performance and time complexity with state-of-the-art works.

2) NON-BLOCKED MODEL-BASED APPROACHES

Our previous work, RDS-SLAM [18] implemented based on ORB-SLAM3, adds a novel semantic tracking thread that segments objects with the aid of Mask R-CNN or SegNet, and then uses the semantic information to update and propagate the moving probability of map points in ATLAS [24]. We follow the ideas of RDS-SLAM and mainly solve the insufficient semantic information problem when using Mask R-CNN. In this study, the concepts of frames, keyframes, BA, and global maps are derived from RDS-SLAM and ORB-SLAM3.

III. RIGID SCENE ASSUMPTION PROBLEM

As shown in Fig. 1, given a 3D point in world coordinate $m_{t-1}^{j} = (x, y, z)^T \in \mathbb{R}^3$ at time $t - 1$, the reprojection error of the predicted and the observed pixels at time t is defined as follows:

$$e_{t,j}(\xi) = x_{t,j} - \pi(T_t^w(\xi), m_{t-1}^j), \tag{1}$$

where, $x_{t,j} \in \mathbb{R}^2$ is the observed feature point; $T_{tw}(\xi) = exp(\xi^\wedge) \in SE(3)$ is the pose of camera t under the world coordinate with $exp(.)$ as a mapping from $se(3)$ to $SE(3)$; $\xi \in \mathbb{R}^6$ is a 6D vector (3 for position and 3 for rotation), which is the target variable to be solved and optimized; π is a project function that projects a map point from the 3D space to the 2D image plane. In a static scene, $x_{t,j}$ should be in the position of $x'_{t,j}$ or very near position (influenced by the noise). vSLAM performs camera ego-motion estimation by minimizing the reprojection error using the matched feature and landmark pairs. In practice, usually, BA is used to find an optimal solution using the error term Eq. (1) and the following cost function:

$$C = \sum_{t,j} \rho_h(e_{t,j}(\xi)^T \Omega_{t,j}^{-1} e_{t,j}(\xi)), \tag{2}$$

where a robust Huber kernel ρ_h is used to reduce the influence of spurious matching. For example, in ORB-SLAM3, g2o [44] is used to solve this BA problem. However, in dynamic environments, the observed and predicted positions may be different due to the movement of dynamic objects. For example, the old map point m_{t-1}^j moves to a new position/point m_t^j. By default, the traditional vSLAM cannot detect the motion and still use the old map point m_{t-1}^j to estimate the camera motion. If the motion of objects is considered, the error term should be defined as follows:

$$\begin{aligned} e_{t,j}(\xi) &= x_{t,j} - \pi(T_t^w(\xi), m_t^j) \qquad (3)\\ &= x_{t,j} - \pi(T_t^w(\xi), H_{t-1}^t m_{t-1}^j), \qquad (4) \end{aligned}$$

where, H_{t-1}^t is the motion of the landmark m_{t-1}^j from the previous time $t - 1$ to the current time. BA cannot optimize the camera pose correctly in dynamic environments due to the unknown motion H_{t-1}^t of landmarks. Usually, this operation will cause a non-robust camera pose estimation or tracking failure due to the large reprojection error.

To the best of our knowledge, there are two kinds of solutions. One solution [42], [43] jointly optimizes the motion H of the object and the camera pose $T(\xi)$ using multi-object tracking by assuming the object is rigid and the points on it have the same or consistent motion. It has been reported that this assumption works in outdoor where the distances of objects are relatively large. However, this assumption does not hold for non-rigid objects, e.g., people in indoor environments. Besides, such methods are offline or not real-time because they use the blocked model.

Another solution is to detect outliers and remove them from tracking, which is widely employed in the geometric and semantic-based approaches. In this case, the cost function is defined as follows:

$$C = \sum_{t,j} W_j \rho_h(e_{t,j}(\xi)^T \Omega_{t,j}^{-1} e_{t,j}(\xi)). \tag{5}$$

In some studies, W_j is assigned to 0 and 1 for dynamic and static points, respectively. In RDS-SLAM, we set W_j to the static probability (1 - moving probability) of each matched landmark in BA. We do not delete the dynamic map points because they are useful to reduce the situation that too few matches are left by directly deleting the outliers from the map. We keep updating the moving probability of each map point over time and reduce the influence of dynamic objects in tracking. Usually, semantic-based approaches can achieve more robust tracking performance using high-level semantic information. In this study, we also use this idea and focus on the indoor environment where people are the main dynamic objects.

IV. SYSTEM OVERVIEW

Fig. 4 shows the architecture of RDMO-SLAM, which is implemented based on ORB-SLAM3 and RDS-SLAM. There are four threads in ORB-SLAM3: tracking, local mapping, loop closing, and full BA. In RDS-SLAM, we add a semantic thread to request the semantic information and update the moving probability of map points into ATLAS. We classify these landmarks into three subsets, unknown, static, and dynamic according to their moving probability, and then use as many static ones as possible in the tracking thread. In this study, we follow the basic idea of RDS-SLAM, add two new threads, optical flow, and velocity estimation threads, and modify some modules of RDS-SLAM and ORB-SLAM3.

The tracking thread aims to estimate the initial camera pose via feature matching and select keyframes used by the local mapping thread to update the map and further optimize the pose estimation via BA. In the semantic thread, first, we request semantic labels of selected keyframes, then generate mask images of predefined dynamic objects, and finally calculate as well as update the moving probability of

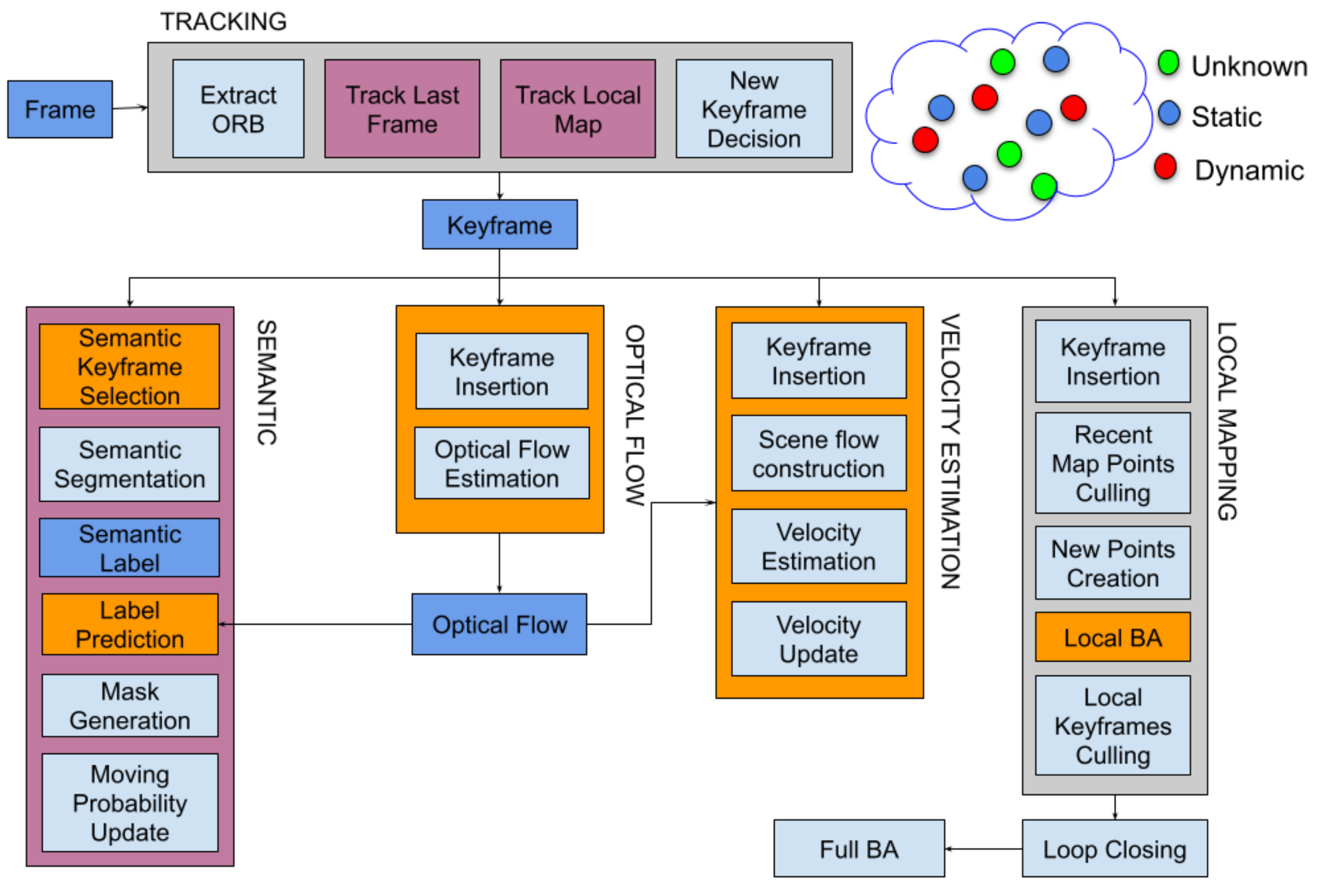


FIGURE 4. System architecture. Models with orange color are the ones that are modified from RDS-SLAM or new blocks. Models with magenta color are derived from RDS-SLAM but different from ORB-SLAM3. Blocks in blue are important data structures.

map points in the global map using semantic information. Different from RDS-SLAM, we add a new module called *Label Prediction*, which is designed to predict semantic labels using optical flow while waiting for the semantic result. In the optical flow thread, we estimate the dense optical flow for each keyframe and use the optical flow to predict the semantic label and estimate the scene flow of landmarks. The velocity estimation thread aims to calculate and update the velocity of map points using the scene flow of map points. The velocity of landmarks is used as another constraint to filter bad data associations from tracking. Finally, this semantic information expressed by the moving probability and the velocity of landmarks is used to filter the outliers.

V. OPTICAL FLOW THREAD

Optical flow estimation is a basic computer vision problem and has many applications, such as autonomous driving, multi-object tracking, and vSLAM. PWC-Net [45] is a compact but effective CNN model for optical flow estimations. It adopts a fast, scalable, and end-to-end trainable CNN framework [46]. It is designed using well-established principles, pyramidal processing, warping, and the use of a cost volume. It warps the CNN features of the second image toward the first image. Then uses the warped features and the features of the first image to construct a cost volume, which is processed by a CNN to estimate the optical flow. PWC-Net is more lightweight and easier to train than the recent FlowNet2 [47] model and can run at about 35 fps [45] on Sintel [48] resolution (1024×436).

Each pixel of optical flow result stores two float values $F = (f_x, f_y) \in \mathbb{R}^2$, which indicate the displacement of each pixel between a previous and current image. Formally, for each pixel (x_1, y_1) in the previous image, the corresponding pixel in the current image is given by:

$$(x_2, y_2)^T = (x_1 + f_x, y_1 + f_y)^T. \tag{6}$$

A ROS version of PWC-Net[1] (Caffe models[2]) is used to predict the optical flow for each pixel of consecutive keyframes. The input is the consecutive two RGB images, and the output is the optical flow, as shown in Fig. 5. Optical flow can only detect the motion part of the body, e.g., hand and leg. However, the unstable features on the static parts of the

[1]https://github.com/ActiveIntelligentSystemsLab/pwc_net_ros
[2]https://github.com/NVlabs/PWC-Net/blob/master/Caffe/model/pwc_net.caffemodel

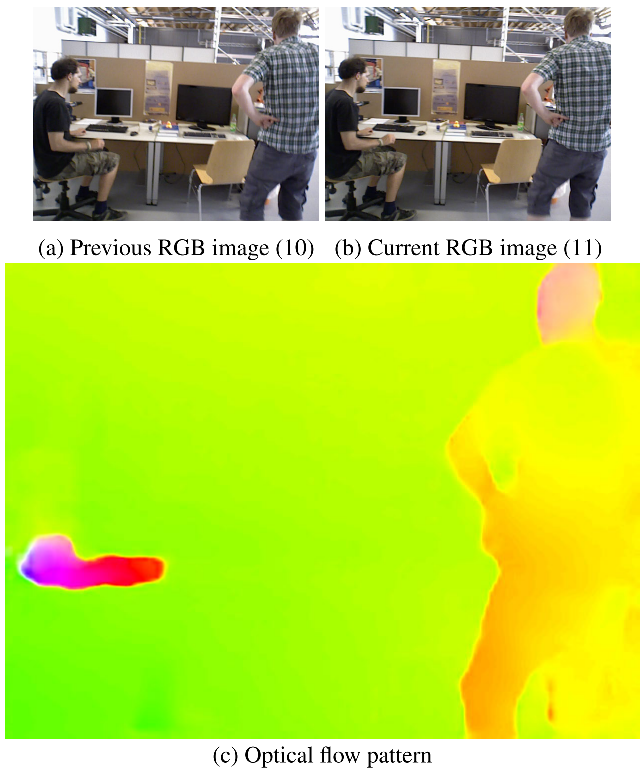


FIGURE 5. Optical flow estimation example using the TUM dataset. (a) and (b) are images from consecutive keyframes, and (c) is their optical flow visualized using HSV color constructed by flow direction and flow magnitude.

body cannot be detected. This problem can be solved together using semantic segmentation.

Later, we use the result of optical flow to predict the semantic label of keyframes in the semantic thread to increase the speed of semantic information generation. Besides, the result is also used by velocity thread to calculate the velocity of map points.

VI. SEMANTIC THREAD

This thread aims to provide semantic information and use them to update the moving probability of map points. Fig. 4 (semantic modules) shows the general flow. First, we select one keyframe to request a semantic label using Mask R-CNN. However, it requires a very long time (about 200ms) to obtain the semantic result/label. To obtain more semantic information, we propose an algorithm to predict the semantic label of the keyframes using the previous obtained semantic label and optical flow patterns of the reference keyframes, while waiting for the result of the current semantic request. After obtaining the semantic label, we generate a mask of dynamic objects, which will be used to update the moving probability. We will explain in detail in the following sub-sections.

A. SEMANTIC KEYFRAME SELECTION

The *semantic delay* [18] between the semantic and tracking threads will increase over time if all keyframes are segmented

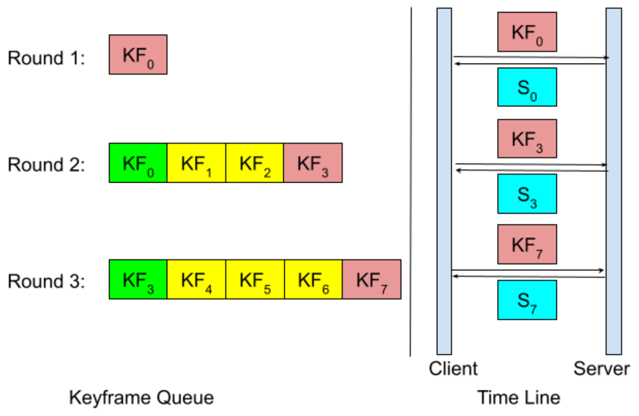


FIGURE 6. Semantic timeline. The left side is the contents inside the keyframe queue *KF*, and the right side is the timeline of requesting semantic labels. $S_{(.)}$ is the semantic label returned from the semantic server. The keyframes in yellow are the ones that need to predict. The keyframes in pink are the ones that request the semantic label from the semantic server, and those in green are the ones that have obtained semantic results in the previous rounds.

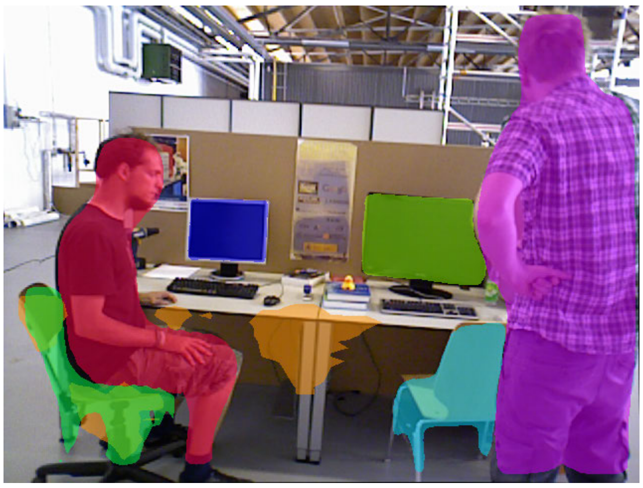


FIGURE 7. Semantic segmentation result.

sequentially. The tracking thread cannot obtain the latest and enough semantic information in real-time. To decrease the semantic delay, only the keyframes from the front and back of the keyframe queue *KF* are selected to request semantic labels in RDS-SLAM. However, this will cause many keyframes not able to obtain semantic results. In other words, not all the keyframes can have the chance to get a semantic result. This may result in non-robust or unstable tracking under complex environments. RDS-SLAM has been only evaluated at 15 Hz rather than 30 Hz in TUM dataset because adequate semantic information cannot be obtained within a short time using Mask R-CNN. To handle this drawback, in this study, we try to ensure that almost all keyframes can obtain semantic labels. We always select the latest keyframe from the back of the *KF* queue to request semantic results.

Fig. 6 shows an example of keyframe selection policy. In round 1, we select the first keyframe KF_0 to request

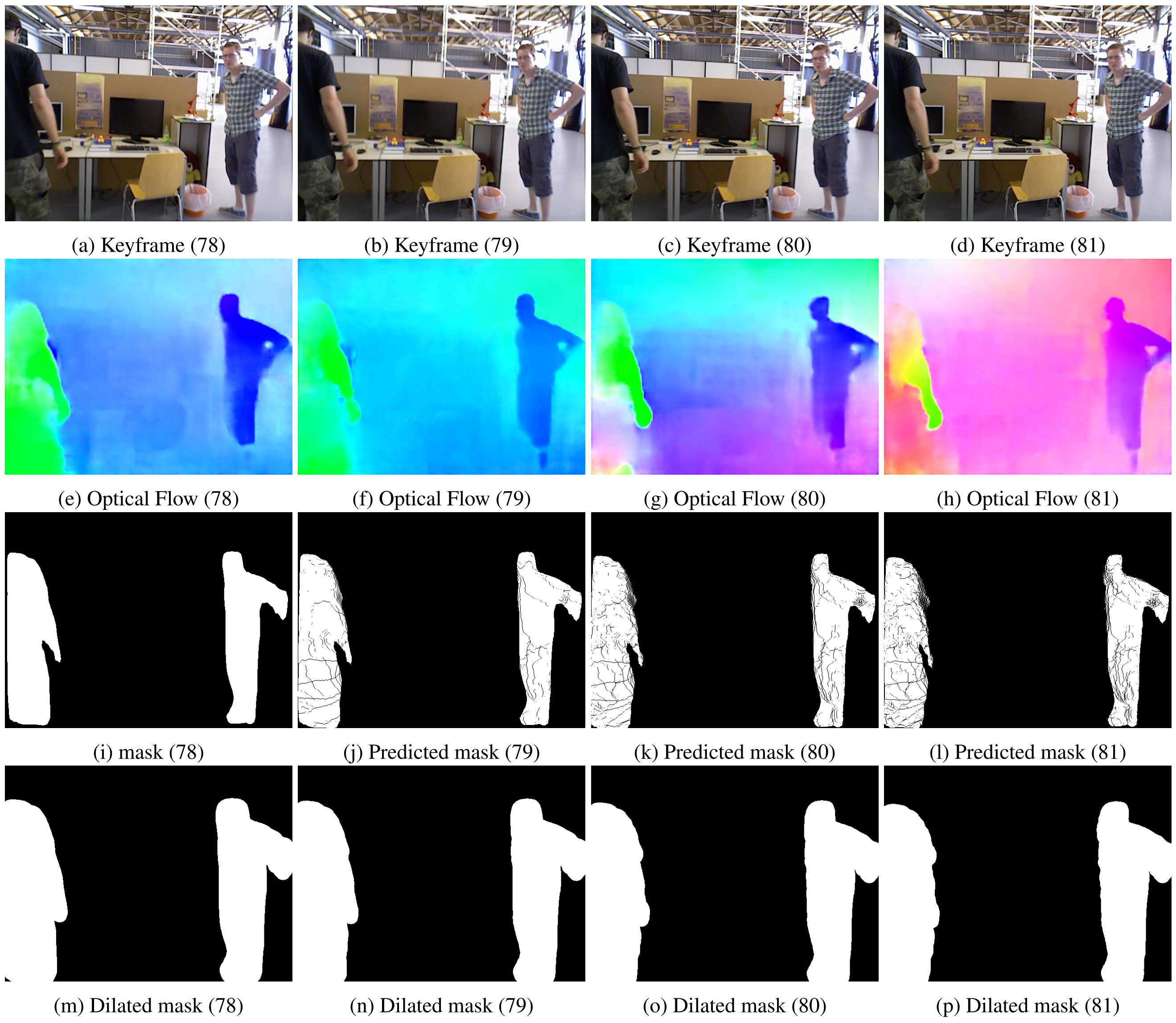


FIGURE 8. Predicted mask. (a) is the reference keyframe, and (b)-(d) are the keyframes that need to predict.

a semantic label from a semantic server (Mask R-CNN). In round 2, we select the latest keyframe (KF_3) in the queue to request semantic labels. We predict the semantic label for others ($KF_1 - KF_2$). Similarly, in the next round, we take the element KF_7 from the back of the queue to request and predict the others sequentially (KF_4 - KF_6).

B. SEMANTIC SEGMENTATION

We use Mask R-CNN[3] trained with the MS COCO [49] dataset as the semantic server. Fig. 7 shows an example of a semantic segmentation result. However, the semantic segmentation result is not always correct, and the edge of the object is difficult to classify. Besides, only pretrained objects can be segmented. Therefore, we dilate the mask to cover the features on the boundary and use the velocity of map points as another constraint to remedy this insufficiency.

[3]https://github.com/matterport/Mask_RCNN

C. SEMANTIC LABEL PREDICTION

To ensure that more keyframes can obtain the semantic label, we predict the semantic labels for not-yet-segmented keyframes using optical flow. As shown in Fig. 6 (round 2), KF_0 is the reference keyframe that has already been segmented, KF_3 the current request, and $KF_1 - KF_2$ are the predicted ones using the reference keyframe. Given a reference keyframe label $I_r(x_r, y_r)$ and the corresponding optical flow vector (f_x, f_y), the predicted label $I_p(x_p, y_p)$ is calculated as follows:

$$I_p(x_p, y_p) = I_p(x_r + f_x, y_r + f_y) = I_r(x_r, y_r). \quad (7)$$

Fig. 8 shows an example of semantic label prediction. From the label of the reference keyframe (a) and the optical flow (f-h), we predict semantic labels of the subsequent keyframes and generate their mask images, as shown in (j-l).

D. SEMANTIC MASK GENERATION

We generate mask images of predefined dynamic objects such as persons and animals by applying dilation operation to the predicted semantic labels to fill the holes and expand object boundaries. As shown in Fig. 9, since the features around the boundary of dynamic objects can also be the outliers, they will be covered after dilating the mask. The noise or holes on the predicted labels can also be smoothened, as shown in Figs. 8 (n-p).

E. MOVING PROBABILITY UPDATE

We define the moving probability $p(m_t^j)$, $m_t^j \in M$ where $M = \{static\ (s), dynamic\ (d))\}$, for a map point j that matches with features in the keyframe, as shown in Fig. 10 [18]. We omit the superscript j in the following derivation. We update the moving probability in the semantic thread using Bayesian filter [50] as follows:

$$\begin{aligned} bel(m_t) &= p(m_t|z_{1:t}, m_0) \\ &= \eta p(z_t|m_t) \int p(m_t|m_{t-1}) bel(m_{t-1}) dm_{t-1}, \end{aligned} \tag{8}$$

where $\eta = 1/(bel(m_t = d) + bel(m_t = s))$ and $p(m_0) = 0.5$ is the initial probability and $p(z_t|m_t)$ is the observation likelihood, which is set according to the semantic label. It is reasonable to assume that the current observation is independent of the previous ones. Thus, we define the observation model as follows:

$$p(z_t = d|m_t = d) = \alpha, \tag{9}$$
$$p(z_t = s|m_t = s) = \beta, \tag{10}$$

where α is given a fixed value in RDS-SLAM. Usually, the segmentation accuracy is influenced by the camera rotation around the optical axis, as shown in Fig. 11, if CNN is not trained using enough data for such cases.

The rotation of the camera is presented as follows:

$$R(r) = rot(T(\xi)) = exp(r^\wedge), \tag{11}$$

where, r is the Euler angle (roll, pitch, yaw) and $r^\wedge \in so(3)$. We heuristically adjust the reliability of semantic segmentation (α) according to the roll component and set α to a small value when the rotation is huge. We set α according to the roll component when it is greater than a threshold γ.

$$\alpha = \begin{cases} max\{min\{0.9, \frac{1}{exp^{(||roll(r)||-2)}}\}, 0.1\} & ||roll(r)|| > \gamma \\ 0.9 & others. \end{cases} \tag{12}$$

In our experiment, γ is set to 1.5 to omit the relatively small camera rotation and β to 0.9 by assuming the observation is fairly robust for static objects.

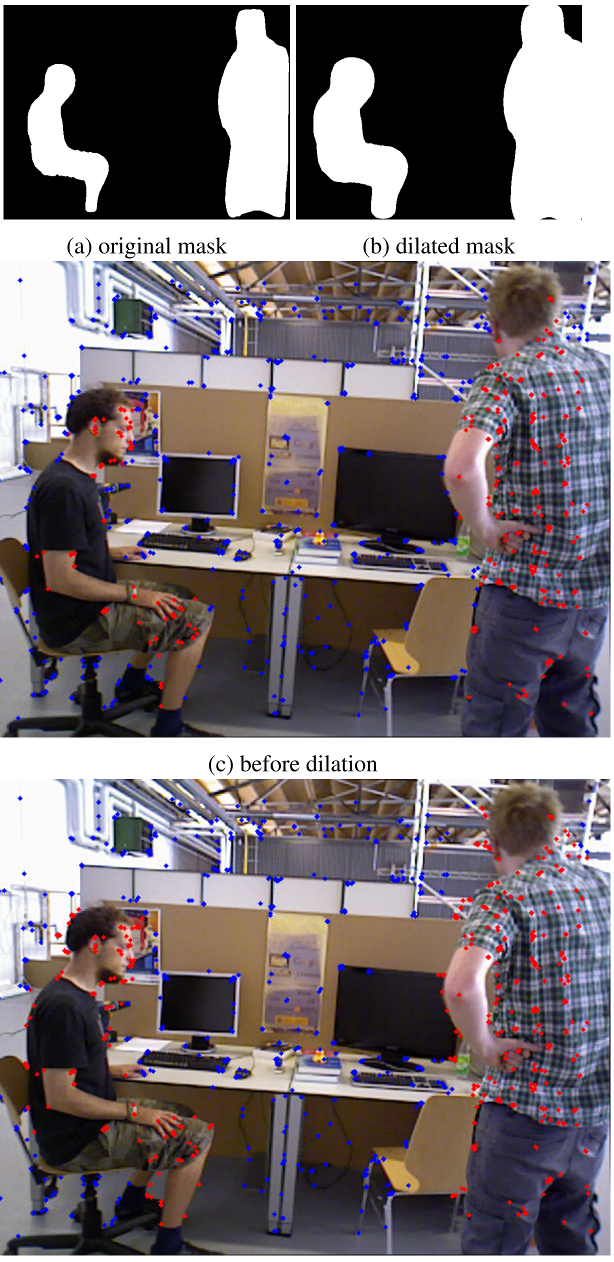


FIGURE 9. Dilation example. Features in red are outliers after dilation operation, and in blue are the observed static features.

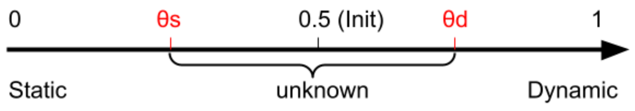


FIGURE 10. Moving probability. θ_s and θ_d are threshold values.

F. ALGORITHM IMPLEMENTATION

Alg. 1 shows the detailed implementation of semantic thread. To maintain the information exchange of optical flow and semantic segmentation threads, checking functions "IsOpticalFlowReady()" and "IsSemanticReady()" are used

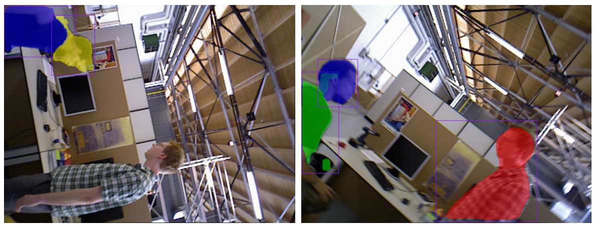

(a) (735) w/rpy (b) (766) w/rpy

FIGURE 11. Segmentation accuracy is not correct in the case of a large camera rotation. (a) the right person is not segmented, and the head of the left person is wrongly segmented. (b) the head of the left person is wrongly segmented.

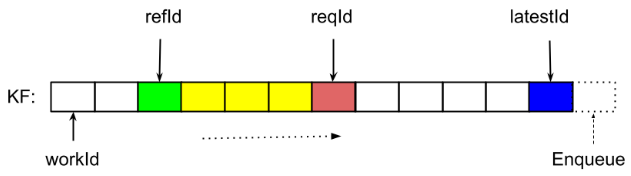


FIGURE 12. Semantic prediction algorithm. The *workId* is the current working pointer that walks through every keyframe sequentially. The *refId* is the id of the reference keyframe, and *reqId* is the id of the last semantic request. *lstestId* is the id of the latest keyframe by now.

respectively to sync the data flow. To handle each keyframe incrementally, we designed some indicators/pointers to control the flow of the algorithm, as shown in Fig. 12. The keyframes in yellow that need to predict are located between the reference keyframe (*refId*) and the last semantic request keyframe (*reqId*). First, we take out the current keyframe (line 5) and the latest keyframe (lines 6-7) from the back of the *KF* queue. Then, we segment the first few keyframes (*initNum*), as shown in Alg. 1 (lines 8-17) considering some datasets are short. The tracking is blocked to wait for the semantic results for these keyframes. Besides, it will consume more time (>300ms in our experiment) to segment the first image due to the GPU initialization. Therefore, we suggest waiting for the segmentation result of the first few keyframes. In the experiment, *initNum* is set to 1 when evaluating the TUM dataset and 0 for a real camera. Lines 21-25 are to select the latest keyframe to request the semantic label non-blocked when 1) the last request *reqKF* has already obtained the semantic label and 2) there are new elements in the *KF* queue waiting to segment. Lines 26-39 are to predict the semantic labels using the selected reference keyframe that have already obtained the semantic result (lines 26-28), and the keyframes that need to predict have already got the optical flow (lines 31-33). The keyframe is predicted when the semantic segmentation processing speed is slower than the new keyframe enqueuing speed. This algorithm predicts semantic labels while waiting for the semantic label. We wait for the segmentation result (lines 18-20) when 1) no new enqueued keyframe exist, or 2) all the keyframes before the last request *reqId* are already handled.

We update the semantic information after the semantic label is obtained either by semantic segmentation or

Algorithm 1 Semantic Thread

```
Require: vector<Keyframe*> KF
    Keyframe *requestKF, *workKF, *latestKF
    int workId, latestId, refId, reqId = 0
    int initNum = 1
    thread* segmentThread
 1: while notRequestFinish() do
 2:   if KF.size() < 1 + workId then
 3:     continue
 4:   end if
 5:   workKF = KF[workId]
 6:   latestKF = KF.back()
 7:   latestId = latestKF->id
 8:   if workId < initNum then
 9:     reqKF = workKF
10:     reqId = workId
11:     segment(reqKF)
12:     updateSemantic(reqKF)
13:     refKF = reqKF
14:     refId = reqId
15:     workId++
16:     continue
17:   end if
18:   if (refId >= reqId) || (latestId == reqId) || (workId >= reqId) then
19:     segmentThread->join()
20:   end if
21:   if reqKF->isSemanticReady() && (workId > reqId) && (latestId > reqId) then
22:     reqKF = latestKF
23:     reqId = latestId
24:     segmentThread = new thread(&segment, reqKF)
25:   end if
26:   if workKF->isSemanticReady() then
27:     refKF = workKF
28:     refId = workId
29:   else
30:     if (refId<reqId) && (workId>refId) && (workId-refId == 1) then
31:       while !workKF->IsOpticalFlowReady() do
32:         sleep(1)
33:       end while
34:     end if
35:     workKF->label = predictLabel(refKF)
36:     updateSemantic(workKF)
37:     refKF = workKF
38:     refId = workId
39:   end if
40:   workId++
41: end while
```

prediction, as shown in Alg. 2. Similarly to RDS-SLAM, we generate the mask images of dynamic objects and update the moving probability of map points using the generated mask.

Algorithm 2 Update Semantic

```
Require: Keyframe* pKF
1: pKF->mask = generateMask(pKF->label)
2: pKF->informSemanticReady()
3: pKF->updateMovingProbability()
```

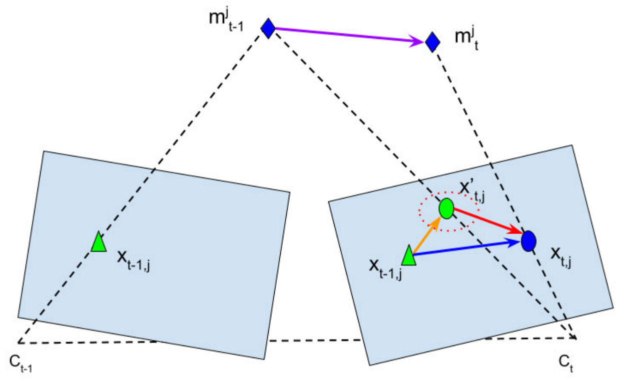


FIGURE 13. Optical and scene flows. The blue vector is the original optical flow vector, orange is the rigid flow, and red is the non-rigid flow. The purple vector is a scene flow vector.

VII. VELOCITY ESTIMATION THREAD

Semantic segmentation can only handle predefined dynamic objects, and the segmentation is not always accurate. In this study, we add velocity constraints for objects to further reduce the influence of outliers.

As shown in Fig. 13, given a pixel $x_{t,j}$ in the previous image, we can estimate the corresponding pixel in the next image using

$$x_{t,j} = x_{t-1,j} + F_{x_{t,j}}, \tag{13}$$

where $F_{x_{t,j}}$ is an optical flow vector shown in Fig. 13 (blue vector) and $x'_{t,j}$ is the estimated point using the camera motion assuming the camera is the only moving object. The motion of the camera needs to be subtracted from the optical flow. Then, the sparse scene flow of landmarks is calculated by

$$\begin{aligned} s &= m_t - m_{t-1} & (14)\\ &= \pi^{-1}(x_{t,j}, D(x_{t,j}), T^w_t(\xi)) \\ &\quad - \pi^{-1}(x_{t-1,j}, D(x_{t-1,j}), T^w_{t-1}(\xi)), & (15) \end{aligned}$$

where π^{-1} is a function that back project one 2D point in the image to the 3D world space using the camera pose and depth image.

The velocity of each map point is calculated by:

$$z_t = \frac{\|s\|}{\Delta t}, \tag{16}$$

where Δt is the time difference between consecutive keyframes.

Some velocities are very large due to inaccurate camera pose estimation, inaccurate depth data, and wrongly matched feature points. Besides, the features are extracted from different pyramid layers of the image. This also results in inaccurate or wrong velocity estimation. We update the velocity using Kalman Filter [50] by:

$$\bar{v}_t = v_{t-1}, \tag{17}$$

$$v_t = \bar{v}_t + K_t(z_t - \bar{v}_t), \tag{18}$$

where K_t is the Kalman gain and z_t is the newly calculated velocity. We assume the map points move at a constant speed. The predicted velocity $\bar{v}_t$ is equal to the previous speed. Ideally, the speed of static map points should be nearly zero.

We use the velocity of map points as another constraint to further filter outliers. As we knew, it is difficult to find an optimal threshold to judge outliers. Not enough features may be left if the threshold is set too small. In our view, there is no close form to decide the optimal value for all the frames or scenes. In our experiment, we set a large value to remove only obvious outliers with very large velocity.

VIII. TRACKING

To let vSLAM run in real-time, we separated the semantic thread and the velocity estimation thread from the tracking thread, so as not to block the tracking. The moving probability and the velocity of landmarks are stored in the map. We use them as constraints to filter outliers from camera ego-motion estimation.

As shown in Fig. 10, we judge the status of objects using

$$Status(m_t) = \begin{cases} dynamic & bel(m_t) > \theta_d \\ static & bel(m_t) < \theta_s \\ unknown & others. \end{cases} \tag{19}$$

This is used as a constraint to select relatively good data associations (see robust data association algorithm in [18]) and reduce the influence of dynamic objects in tracking for every frame. In the experiment, θ_d is set to 0.6 and θ_s to 0.4.

This module is to estimate the initial camera pose by matching the features between the previous frame and the current frame. Similar to RDS-SLAM, we use the moving probability as the constraint as defined in Eq. (19). First, we use features in the static subset. If the matched feature pairs are not enough, we use the features in the unknown subset. If they are still not enough, the features in the dynamic feature subset can also be used such as the ones of a person who is sitting. In the experiment, the dynamic feature subset was not used when evaluating the TUM dataset. The estimated initial pose may not be very reliable; however, it is further optimized via tracking local map and BA.

BA is used in the local mapping thread (local BA), the loop closing thread, and the full BA thread. We use moving probability and velocity constraints to filter outliers from them.

IX. EXPERIMENTAL RESULTS

We demonstrated the real-time performance and the tracking accuracy by comparing with state-of-the-art vSLAMs using the indoor dynamic scenes of the TUM dataset.

Our system was evaluated using GeForce RTX 2080Ti GPU, Cuda 11.1, and an RGB-D camera (Kinect V2). We also showed a demo of AR using a Kinect v2 camera in the real environment.

A. TRACKING ACCURACY EVALUATION

There are both high and low dynamic office sequences in the TUM RGB-D dataset [19] recorded with a Microsoft Kinect sensor at full frame rate (30 Hz). RGB (640×480), depth images, together with ground-truth trajectory, recorded by a high-accuracy motion capture system. There is a low degree of motion in the sequence named fr3/sitting_* (s/*), where two people are sitting in front of a desk while speaking with gestures. Two people are walking both in the background and foreground, and they sometimes sit down in front of the desk in the sequences named fr3/walking_* (w/*). There are four types of camera motions, 1) half-sphere (half): moved on a small half-sphere of approximately one-meter diameter, 2) xyz: manually moved along three directions (xyz) while keeping the same orientation, 3) rpy: rotated along the principle axes (roll-pitch-yaw) at the same position, 4) static: kept in a place manually.

In this stdudy, we compared the trajectories of our proposal with state-of-the-art vSLAM algorithms, as shown in Fig. 14, using their source codes when possible, ORB-SLAM3,[4] DS-SLAM,[5] Dyna-SLAM,[6] KMOP-vSLAM [15], and RDS-SLAM[7] using only an RGB-D camera (no IMU).

We evaluated the tracking performance using absolute trajectory error (ATE) and relative pose error (RPE) [19]. The root means squared error (RMSE) and standard deviation (S.D) are used as the error metrics. Given the estimated trajectory: $P_1, \dots, P_n \in SE(3)$, ground truth trajectory $Q_1, \dots, Q_n \in SE(3)$, and a fixed time interval Δ. The RPE at time i is defined as follows:

$$R_i = (Q^{-1} Q_{i+\Delta})^{-1} (P_i^{-1} P_{i+\Delta}). \tag{20}$$

The RMSE of RPE over all time is defined as follows:

$$RMSE(R_{1:n}) = \frac{1}{n} \sum_{\Delta=1}^{n} (\frac{1}{m} \sum_{i=1}^{m} \|trans(R_i)\|^2)^{\frac{1}{2}}. \tag{21}$$

The ATE error is defined as follows:

$$A_i = Q_i^{-1} S P_i, \tag{22}$$

where $S \in Sim(3)$, which corresponds to the least squares solution that maps the estimated trajectory onto the ground truth trajectory. The RMSE of ATE over all time indices is defined as follows:

$$RMSE(A_{i:n}, \Delta) = (\frac{1}{n} \sum_{i=1}^{n} \|trans(A_i)\|^2)^{\frac{1}{2}}. \tag{23}$$

[4]https://github.com/UZ-SLAMLab/ORB_SLAM3.git
[5]https://github.com/ivipsourcecode/DS-SLAM.git
[6]https://github.com/BertaBescos/DynaSLAM
[7]https://github.com/yubaoliu/RDS-SLAM

We compared the tracing performance with counterpart state-of-the-art vSLAMs: ORB-SLAM3 [23], KMOP [15], Detect-SLAM [8], VO-SF [33], Elastic Fusion [38], CO-Fusion [34], Static Fusion [37], DP-SLAM [39], DynaSLAM [10], SLAM-PCD [40], DM-SLAM [11], and RDS-SLAM [18], using, when possible, results published in the original papers, as shown in Tab. 1, Tab. 2 and Tab. 3. We achieved a similar tracking performance with state-of-the-art semantic-based methods in dynamic environments using a heavy segmentation method, Mask R-CNN.

We achieved similar tracking performance compared with the methods that use the blocked model. However, these methods cannot achieve good real-time performance. The proposed method can run the Mask R-CNN version vSLAM in real-time while keeping the robust tracking. We will demonstrate the real-time performance later.

B. OUTLIERS REMOVING USING TUM DATASET

We qualitatively checked the feature classification performance by evaluating the TUM dataset. The features can be classified into three subsets according to the moving probability (Eq. (19)), as shown in Fig. 15. The static features are mostly distributed on static objects, and the unstable features (green and red) are mostly on the moving people. In the tracking thread, we try to use as static features as we can. An example is shown in Fig. 16, wherein only selected good static features are used in the initial camera pose estimation stage in the tracking.

C. AR DEMO

We qualitatively evaluated our system using an AR demo, as shown in Fig. 17, where a virtual cube is put on the desk. One person is sometimes sitting down and standing up, and sometimes the person occupies half of the camera view. The tracking is very unstable or even tracking lost in the situation such as Figs. 17 (b-d) when using the original ORB-SLAM. In this demo, the position of the virtual object is somehow influenced by the person due to the occlusion (e.g., Figs. 17 (b) and (d)); however, it recovers to its original position after the person leaves (Fig. 17 (e)). We also try to disturb the tracking by moving the keyboard (Fig. 17 (f)) and moving the hand (Figs. 17 (g) and (h)). Tracking in Figs. 17 (f-h) is not influenced by the hands because features on the hands are detected and removed using semantic and motion information.

D. VELOCITY CONSTRAINT VS SEMANTIC INFORMATION

We have evaluated the ATE of TUM only using velocity constraint or semantic information, as shown in Tab. 4. The tracking performance is much better than that of ORB-SLAM3 with the velocity constraint. This constraint can filter the landmarks (matched with features) that have large velocities on the objects, and it is a little faster than Mask R-CNN segmentation. We also evaluated the tracking performance that only uses segmentation. The performance may be not good if the camera rotates and translates rapidly because it

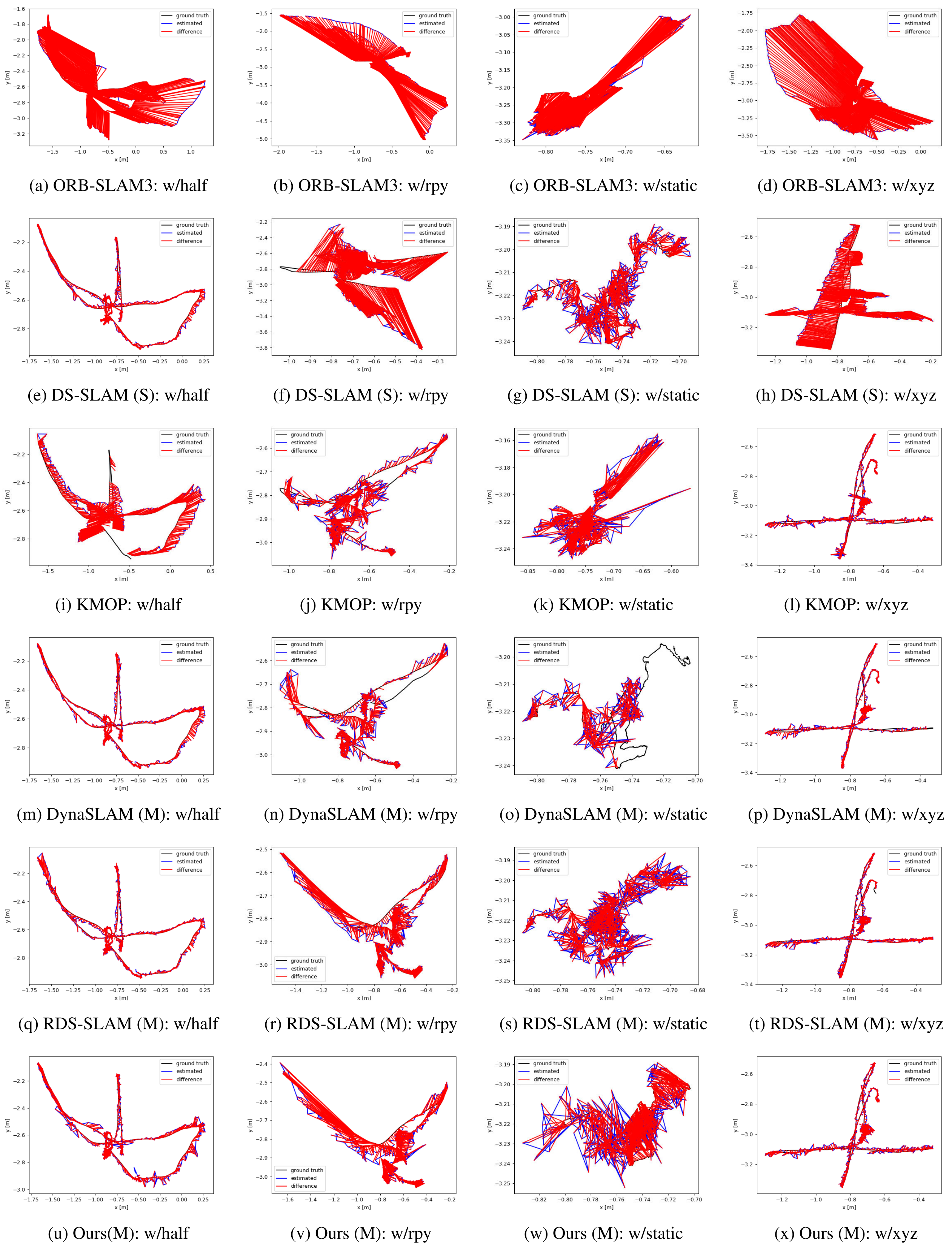


FIGURE 14. Trajectory comparing frame by brame. "M" stands for "Mask R-CNN" and "S" for "SegNet". RDS-SLAM is executed in 15 Hz and ours in 30 Hz.

TABLE 1. Evaluation of absolute trajectory error (ATE) of TUM (m). k-means (K), SegNet (S), Mask R-CNN (M), SharpMask (SM), OpnePose (O) are segmentation or detection methods. RDS-SLAM is evaluated in 15 Hz and ours is evaluated in 30 Hz.

Seq.	ORB SLAM3		KMOP (K+O)	Detect-SLAM (SSD)	VO-SF (K)	Elastic Fusion	CO-Fusion (SM)	Static Fusion (K)	DP-SLAM (M)		DS-SLAM (S)		DynaSLAM (M)		SLAM-PCD (CNN)		DM-SLAM (M)		RDS-SLAM (M)		Ours (M)	
	RMSE	S.D.	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.
w/half	0.6572	0.3124	0.176	0.0514	0.739	0.638	0.803	0.391	0.0254	0.0129	0.0303	0.0159	0.0296	0.0157	**0.0241**	0.0122	0.0274	0.0137	0.0259	0.0141	0.0304	0.0141
w/rpy	1.0197	0.5122	0.049	0.2959	-	-	-	-	**0.0356**	0.0218	0.4442	0.2350	0.0354	0.019	0.0453	0.0316	0.0328	0.0194	0.1468	0.1051	0.1283	0.1047
w/static	0.3614	0.1522	0.032	-	0.327	0.293	0.551	0.014	0.0079	0.0037	0.0081	0.0033	**0.0068**	0.0032	0.0077	0.0039	0.0079	0.0040	0.0815	0.0224	0.0126	0.0071
w/xyz	0.9178	0.4859	0.019	0.0241	0.874	0.906	0.696	0.127	**0.0141**	0.0073	0.0247	0.0161	0.0164	0.0086	0.0157	0.0084	0.0148	0.0072	0.0213	0.0127	0.0226	0.0137
s/static	0.0090	0.0043	-	-	0.029	0.008	0.011	0.013	**0.0059**	0.0029	0.0065	0.0033	0.0108	0.0056	0.0080	0.0037	0.0063	0.0032	0.0088	0.0043	0.0066	0.0033

TABLE 2. Evaluation of translational relative pose error (RPE) (m) of TUM.

Seq.	ORB SLAM3		KMOP (K+O)	VO-SF (K)	Elastic Fusion	CO-Fusion	BaMVO	Static Fusion (K)	DP-SLAM (M)		DS-SLAM (S)		DynaSLAM (M)		SLAM-PCD (CNN)		RDS-SLAM (M)		Ours (M)	
	RMSE	S.D.	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.
w/half	0.3262	0.2625	0.07	0.335	0.205	0.40	0.173	0.207	**0.0142**	0.0082	0.0297	0.0152	0.0284	0.0149	0.0274	0.0140	0.0282	0.0155	0.0294	0.013
w/rpy	0.4368	0.3197	0.065	-	-	-	-	-	**0.0225**	0.0150	0.1503	0.1168	0.0448	0.0262	0.0616	0.0357	0.1114	0.092	0.1396	0.1176
w/static	0.7800	0.7563	0.033	0.101	0.26	0.224	0.133	0.013	**0.0066**	0.0038	0.0102	0.0038	0.0089	0.0044	0.0102	0.0049	0.0419	0.0348	0.016	0.009
w/xyz	0.4258	0.3063	0.026	0.277	0.24	0.329	0.232	0.121	**0.0114**	0.0063	0.0333	0.0229	0.0217	0.0119	0.0204	0.0107	0.0281	0.0167	0.0299	0.0188
s/static	0.0102	0.0049	-	0.024	0.009	**0.0011**	0.024	0.011	0.0054	0.0027	0.0078	0.0038	0.0126	0.0067	0.0087	0.0038	0.0107	0.005	0.009	0.004

TABLE 3. Evaluation of rotational pose error (RPE) (m) of TUM.

Seq.	ORB SLAM3		KMOP (O+K)	VO-SF (K)	Elastic Fusion	CO-Fusion (SM)	BaMVO	Static Fusion (K)	DP-SLAM (M)		DS-SLAM (S)		DynaSLAM (M)		SLAM-PCD (CNN)		RDS-SLAM (M)		Ours (M)	
	RMSE	S.D.	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.
w/half	7.2352	5.9487	1.595	0.0669	0.0641	0.1302	0.0428	0.0504	**0.0106**	0.0059	0.8142	0.4101	0.7842	0.4012	0.7440	0.3459	0.8216	0.4347	0.7915	0.3782
w/rpy	8.7683	6.4583	1.105	-	-	-	-	-	**0.0128**	0.0082	3.0042	2.3065	0.9894	0.5701	1.3831	0.8318	9.3192	8.572	2.5472	2.0607
w/static	6.0054	5.5995	0.627	0.0168	0.0477	0.0401	**0.0208**	0.0038	0.0044	0.0023	0.2690	0.1215	0.2612	0.1259	0.2631	0.1119	1.1686	0.9917	0.3385	0.1612
w/xyz	7.8974	5.5917	0.689	0.0511	0.0479	0.0555	0.0439	0.0266	**0.0093**	0.0065	0.8266	0.2826	0.6284	0.3848	0.6227	0.3807	0.7236	0.4435	0.799	0.5502
s/static	0.3007	0.1300	-	0.0071	**0.003**	0.0044	0.0069	0.0043	0.0040	0.0021	0.2735	0.1215	0.3416	0.1642	0.2782	0.1210	0.3091	0.1325	0.291	0.133

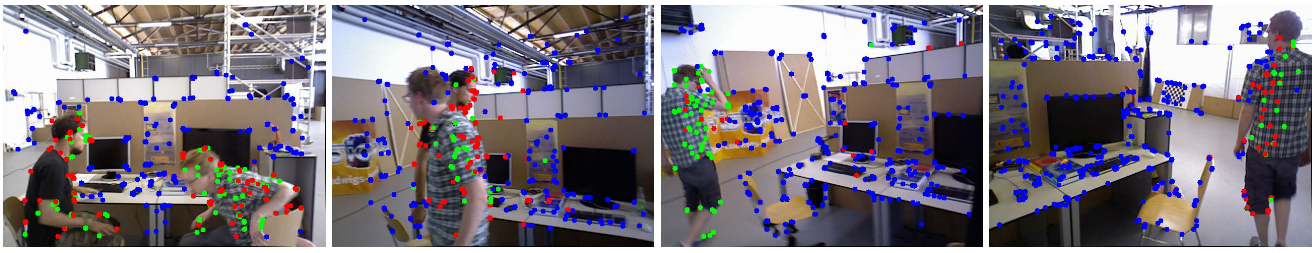

(a) Frame 2 (b) Frame 50 (c) Frame 100 (d) Frame 205

FIGURE 15. Classify objects according to the moving probability (w/half). Green features are unknown and red ones are dynamic, and blue ones are static.

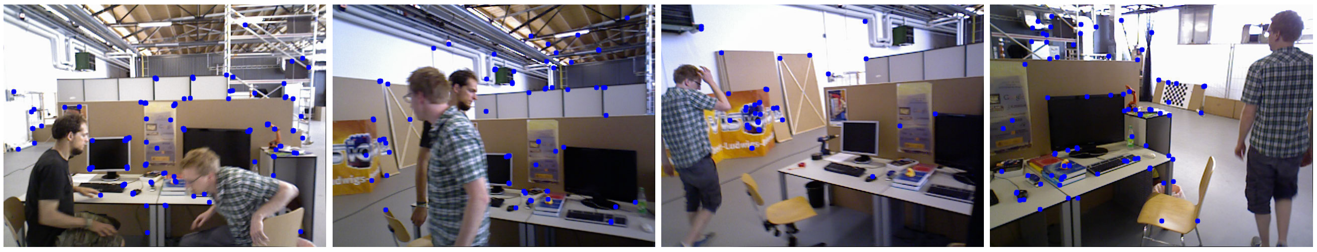

(a) Frame 2 (b) Frame 50 (c) Frame 100 (d) Frame 205

FIGURE 16. Use robust features in tracking (w/half).

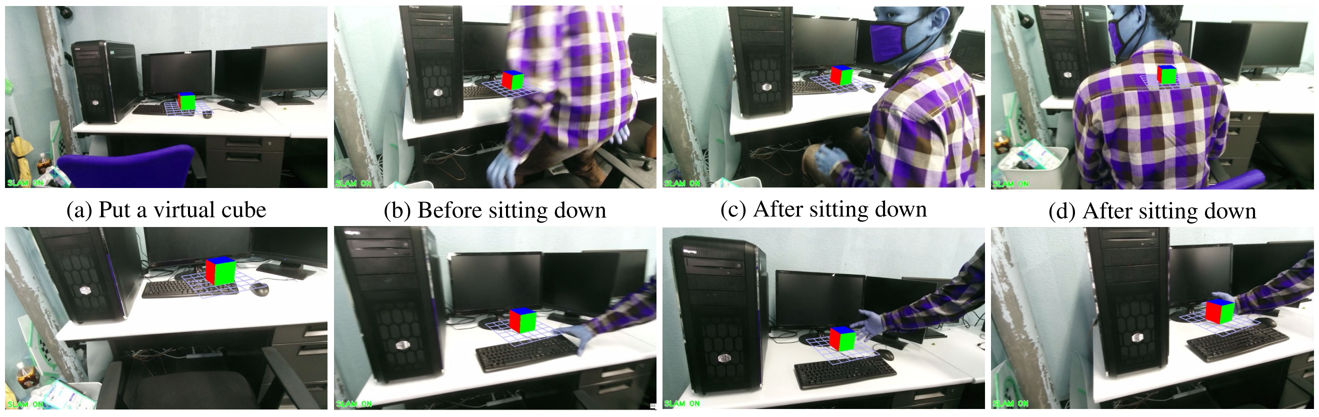
does not have enough time to obtain semantic information. That is why the tracking performance is a little lower in w/rpy and w/half. This problem can be solved by combining velocity constraints and semantic information. The tracking performances for other scenarios are very similar in the case of only using semantic and using both, especially in the standard deviation.

TABLE 4. Evaluation of absolute trajectory error (ATE) of TUM (m) with or without velocity and semantic mask. "V" means only use velocity and "M" only use the semantic mask.

Seq.	ORB-SLAM3		Ours (V)		Ours (M)		Ours (V+M)	
	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.
w/half	7.2352	5.9487	0.6314	0.3518	0.1204	0.0952	**0.0304**	0.0141
w/rpy	8.7683	6.4583	1.0978	0.5768	0.2708	0.2039	**0.1283**	0.1047
w/static	6.0054	5.5995	0.3867	0.1756	**0.0124**	0.0077	0.0126	0.0071
w/xyz	7.8974	5.5917	0.6479	0.3363	**0.0164**	0.0085	0.0226	0.0137
s/static	0.3007	0.1300	0.0089	0.004	**0.0064**	0.003	0.0066	0.0033

E. VELOCITY CONSTRAINT THRESHOLD

It is challenging to decide the threshold of the velocity to support robust tracking. We analyzed the landmark distribution that matched with features on the keyframes in the terms of the velocity. As shown in Fig. 18, the velocity of about a half of landmarks is less than 2.0 in TUM w/xyz. We use the



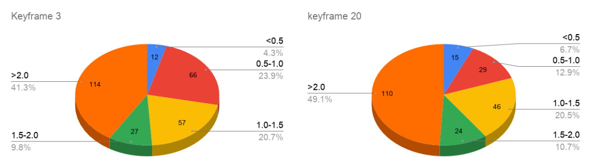
(a) Put a virtual cube (b) Before sitting down (c) After sitting down (d) After sitting down

(e) Person left (f) Move the keyboard (g) Disturb the cube (far) (h) Disturb the cube (near)

FIGURE 17. **AR demo.**

TABLE 5. **The execution time comparison of the TUM dataset. We use the data in their original paper as possible. If not provided, we approximate the processing time.**

Method	Model	GPU	Semantic	Segmentation/Detection Time (ms)	Time of Other Models Related to Tracking (ms)		Total Time for Each Frame (ms)
ORB-SLAM3	-	-	-	-	-	-	22 - 30
KMOP-vSLAM	blocked	GeForce GTX 1080 Ti	Open Pose k-means	45.89 95.29	Geometric Constraints Moving Detection Camera Ego-motion	221.636	257.819
DP-SLAM	blocked	GeForce MX150	Mask R-CNN	200	Geometric Constraints Background Inpainting Moving probability Updating	- - -	>200
Detect-SLAM	blocked	GTX960M	SSD	310	Progation Updating	20 10	>310
DS-SLAM	blocked	P4000	SegNet	37.57330	ORB feature extraction Moving consistency check	9.375046 29.50869	>65
DynaSLAM	blocked	Tesla M40 GPU	Mask R-CNN	195	Multi-view Geometry Background Inpainting	235.98 (w/rpy) 183.56 (w/rpy)	>300
DM-SLAM	blocked	GeForce GTX 1080 Ti	Mask R-CNN	201.02	Ego-motion Dynamic Point Detection	3.16 40.64	>201
RDS-SLAM (TUM)	non-blocked	GeForce RTX 2080Ti	Mask R-CNN	200	Mask Generation Update Moving Probability Semantic-based Optimization	5.42 0.17 0.54	50 - 65 (15 Hz)
Ours	non-blocked	GeForce RTX 2080Ti	Mask R-CNN	200	Optical flow estimation Update Moving Probability Mask Generation Velocity Estimation Label Prediction	54 0.14 6.04 2.54 1.56	**22-35 (30 Hz)**



(a) Keyframe 3 (b) Keyframe 20

FIGURE 18. **Landmark distribution according to the velocity range (w/xyz).**

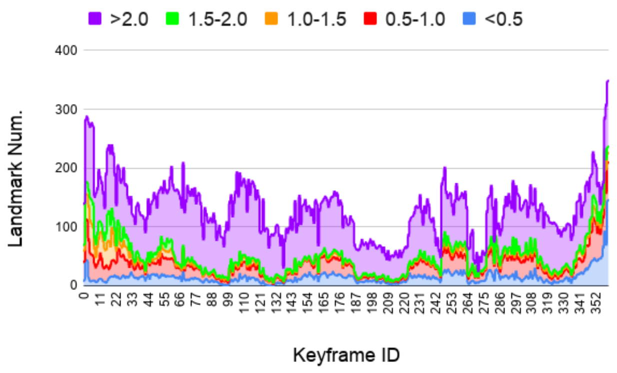


FIGURE 19. **The number of landmarks in the different velocity ranges (w/xyz).**

landmarks that have a relatively small velocity to optimize the camera pose in BA. A very small number of landmarks will be left when setting the threshold too small, and too much noise data are used when setting it too large. We suggest setting the velocity threshold to 1.0-2.0 (see Fig. 19 (orange, red, and green lines)) because the number of landmarks used is reasonable in the optimization. To avoid the tracking loss due to the few landmarks, we do not use this constraint in the "track last frame" and "track local map" models in the tracking thread. We only use this constraint in the local BA where many landmarks are used together for optimization.

F. TIMING ANALYSIS

Tab. 5 shows the comparison result of the real-time performance. We compared the time required for the original ORB-SLAM3 (RGB-D camera only), blocked model-based solutions (e.g., DP-SLAM, Detect-SLAM, DS-SLAM, DynaSLAM, DM-SLAM), and non-blocked model-based solutions (e.g., RDS-SLAM). The time required for the blocked model is limited by the time-consuming semantic segmentation, which significantly lowers their real-time performance. Our previous study, RDS-SLAM only can evaluate the TUM dataset at 15 Hz because the TUM dataset is usually short (about half a minute) and Mask R-CNN only can segment very few keyframes, which results in inadequate semantic information when running at 30 Hz. We mitigate this limitation via predicting the semantic label, which enables almost all the keyframes to obtain semantic results even when executing at 30 Hz.

The tracking performance may be influenced by the hardware configuration because the speed of Mask R-CNN and PWC-Net rely on the GPU. However, the time required for tracking each frame is not influenced due to the non-blocked architecture.

X. CONCLUSION

We proposed RDMO-SLAM, a novel real-time vSLAM for the real environment exploiting RDS-SLAM, Mask R-CNN, and dense optical flow. To overcome the problem of inadequate semantic information obtained within a short time due to the slow speed of Mask R-CNN segmentation, we predict semantic labels using optical flow so that almost all the keyframes can acquire the semantic information. To reduce the influence of dynamic objects untrained by semantic segmentation models, we add a velocity constraint by estimating the velocity of landmarks using optical flow. The tracking and real-time performances are evaluated using the dynamic scenes of the TUM RGB-D dataset and compared with counterpart state-of-the-art vSLAMs with similar motivation. As a result, our proposal that uses a non-blocked model can maintain real-time nature (30 Hz) even with a very heavy segmentation method. In future works, we will 1) consider the outdoor environment and 2) build a static dense map without dynamic objects.

REFERENCES

[1] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: A survey from 2010 to 2016," *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, no. 1, pp. 1–11, Dec. 2017.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[3] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3D mapping with an RGB-D camera," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, Sep. 2014.

[4] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep. 2014, pp. 834–849.

[5] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2016.

[6] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[7] A. J. Davison, "FutureMapping: The computational structure of spatial AI systems," 2018, *arXiv:1803.11288*. [Online]. Available: http://arxiv.org/abs/1803.11288

[8] F. Zhong, S. Wang, Z. Zhang, C. Chen, and Y. Wang, "Detect-SLAM: Making object detection and SLAM mutually beneficial," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 1001–1010.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, in Lecture Notes in Computer Science, vol. 9905. Amsterdam, The Netherlands: Springer, Oct. 2016, pp. 21–37.

[10] B. Bescos, J. M. Fácil, J. Civera, and J. L. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018.

[11] J. Cheng, Z. Wang, H. Zhou, L. Li, and J. Yao, "DM-SLAM: A feature-based SLAM system for rigid dynamic scenes," *ISPRS Int. J. Geo-Inf.*, vol. 9, no. 4, pp. 1–18, 2020.

[12] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[13] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "DS-SLAM: A semantic visual SLAM towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1168–1174.

[14] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[15] Y. Liu and J. Miura, "KMOP-vSLAM: Dynamic visual SLAM for RGB-D cameras using K-means and OpenPose," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2021, pp. 415–420.

[16] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1302–1310.

[17] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, 1967, pp. 281–297.

[18] Y. Liu and J. Miura, "RDS-SLAM: Real-time dynamic SLAM using semantic segmentation methods," *IEEE Access*, vol. 9, pp. 23772–23785, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9318990/

[19] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.

[20] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proc. 8th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2009, pp. 83–86.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[22] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[23] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM," *IEEE Trans. Robot.*, pp. 1–17, Jul. 2020.

[24] R. Elvira, J. D. Tardos, and J. M. M. Montiel, "ORBSLAM-atlas: A robust and accurate multi-map system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 6253–6259.

[25] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[26] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 2100–2106.

[27] S. Li and D. Lee, "RGB-D SLAM in dynamic environments using static point weighting," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2263–2270, Oct. 2017.

[28] Y. Sun, M. Liu, and M. Q.-H. Meng, "Motion removal for reliable RGB-D SLAM in dynamic environments," *Robot. Auton. Syst.*, vol. 108, pp. 115–128, Oct. 2018.

[29] J. Cheng, Y. Sun, and M. Q.-H. Meng, "Improving monocular visual SLAM in dynamic environments: An optical-flow-based approach," *Adv. Robot.*, vol. 33, no. 12, pp. 576–589, Jun. 2019.

[30] D.-H. Kim, S.-B. Han, and J.-H. Kim, "Visual odometry algorithm using an RGB-D sensor and IMU in a highly dynamic environment," in *Robot Intelligence Technology and Applications*, vol. 345. Cham, Switzerland: Springer, 2015, pp. 11–26, doi: 10.1007/978-3-319-16841-8_2.

[31] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robot. Auton. Syst.*, vol. 89, pp. 110–122, Mar. 2017.

[32] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular SLAM in dynamic environments," in *Proc. IEEE Int. Symp. Mixed Augment. Reality (ISMAR)*, Adelaide, SA, Australia, Oct. 2013, pp. 209–218.

[33] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast odometry and scene flow from RGB-D cameras based on geometric clustering," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3992–3999.
[34] M. Runz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 4471–4478.
[35] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *Proc. ECCV* in Lecture Notes in Computer Science, Amsterdam, The Netherlands, vol. 9905, Oct. 2016, pp. 75–91.
[36] D.-H. Kim and J.-H. Kim, "Effective background model-based RGB-D dense visual odometry in a dynamic environment," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1565–1573, Dec. 2016.
[37] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "Static-Fusion: Background reconstruction for dense RGB-D SLAM in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 3849–3856.
[38] T. Whelan, S. Leutenegger, R. Salas Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Robotics: Science and Systems XI*, Los Altos, CA, USA: Robotics: Science and Systems Foundation, Jul. 2015.
[39] A. Li, J. Wang, M. Xu, and Z. Chen, "DP-SLAM: A visual SLAM with moving probability towards dynamic environments," *Inf. Sci.*, vol. 556, pp. 128–142, May 2021.
[40] Y. Fan, Q. Zhang, S. Liu, Y. Tang, X. Jing, J. Yao, and H. Han, "Semantic SLAM with more accurate point cloud map in dynamic environments," *IEEE Access*, vol. 8, pp. 112237–112252, 2020.
[41] N. Dvornik, K. Shmelkov, J. Mairal, and C. Schmid, "BlitzNet: A real-time deep network for scene understanding," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4174–4182.
[42] J. Zhang, M. Henein, R. Mahony, and V. Ila, "VDO-SLAM: A visual dynamic object-aware SLAM system," *ArXiv*, vol. abs/2005.11052, May 2020.
[43] B. Bescos, C. Campos, J. D. Tardos, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5191–5198, Jul. 2021.
[44] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 30, no. 12, pp. 3607–3613, May 2011.
[45] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost, volume," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8934–8943.
[46] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
[47] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1647–1655.
[48] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)* (Lecture Notes in Computer Science), vol. 7577, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Germany: Springer, Oct. 2012, pp. 611–625.
[49] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision—ECCV* (Lecture Notes in Computer Science), vol. 8693. Cham, Switzerland: Springer, Cham, 2014, pp. 740–755.
[50] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. Cambridge, MA, USA: MIT Press, 2005.

YUBAO LIU received the bachelor's degree in computer science from Qufu Normal University, Shandong, China, in 2012, and the master's degree in computer science from Capital Normal University, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with Toyohashi University of Technology, Toyohashi, Aichi, Japan. In 2015, he joined Intel Research Center, Beijing, and transferred to iSoftStone, Beijing, in 2016, as a Senior Software Engineer, working on computer vision and AR. His research interests include pattern recognition and SLAM for AR as well as smart robotics.

JUN MIURA (Member, IEEE) received the B.Eng. degree in mechanical engineering, and the M.Eng. and Dr.Eng. degrees in information engineering from The University of Tokyo, Tokyo, Japan, in 1984, 1986, and 1989, respectively. In 1989, he joined the Department of Computer-Controlled Mechanical Systems, Osaka University, Suita, Japan. From March 1994 to February 1995, he was a Visiting Scientist with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. Since April 2007, he has been a Professor with the Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi, Japan. He has published over 220 articles in international journals and conferences in the areas of intelligent robotics, mobile service robots, robot vision, and artificial intelligence. He has received several awards, including the Best Paper Award from the Robotics Society of Japan, in 1997, the Best Paper Award Finalist at ICRA-1995, and the Best Service Robotics Paper Award Finalist at ICRA-2013.