IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 21, 2020, accepted January 6, 2021, date of publication January 11, 2021, date of current version February 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3050617

RDS-SLAM: Real-Time Dynamic SLAM Using Semantic Segmentation Methods

YUBAO LIU AND JUN MIURA, (Member, IEEE)
Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi 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 The scene rigidity is a strong assumption in typical visual Simultaneous Localization and Mapping (vSLAM) algorithms. Such strong assumption limits the usage of most vSLAM in dynamic real-world environments, which are the target of several relevant applications such as augmented reality, semantic mapping, unmanned autonomous vehicles, and service robotics. Many solutions are proposed that use different kinds of semantic segmentation methods (e.g., Mask R-CNN, SegNet) to detect dynamic objects and remove outliers. However, as far as we know, such kind of methods wait for the semantic results in the tracking thread in their architecture, and the processing time depends on the segmentation methods used. In this paper, we present RDS-SLAM, a real-time visual dynamic SLAM algorithm that is built on ORB-SLAM3 and adds a semantic thread and a semantic-based optimization thread for robust tracking and mapping in dynamic environments in real-time. These novel threads run in parallel with the others, and therefore the tracking thread does not need to wait for the semantic information anymore. Besides, we propose an algorithm to obtain as the latest semantic information as possible, thereby making it possible to use segmentation methods with different speeds in a uniform way. We update and propagate semantic information using the moving probability, which is saved in the map and used to remove outliers from tracking using a data association algorithm. Finally, we evaluate the tracking accuracy and real-time performance using the public TUM RGB-D datasets and Kinect camera in dynamic indoor scenarios.
Source code and demo: https://github.com/yubaoliu/RDS-SLAM.git

INDEX TERMS Dynamic SLAM, ORB SLAM, Mask R-CNN, SegNet, real-time.

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) [1] is a fundamental technique for many applications such as augmented reality (AR), robotics, and unmanned autonomous vehicles (UAV). Visual SLAM (vSLAM) [2] uses the camera as the input and is useful in scene understanding and decision making. However, the strong assumption of scene rigidity limits the use of most vSLAM in real-world environments. Dynamic objects will cause many bad or unstable data associations that accumulate drifts during the SLAM process. In Fig. 1, for example, assume m_1 is on a person and its position changes in the scene. The bad or unstable data associations (the red lines in Fig. 1) will lead to incorrect camera ego-motion estimation in dynamic environments. Usually, there are two basic requirements for vSLAM: robustness in tracking and real-time performance. Therefore, how to detect dynamic objects in the populated scene and prevent the tracking algorithm from using data associations related to such dynamic objects in real-time is the challenge to allow vSLAM to be deployed in the real world.

We classify the solutions into two classes: pure geometric-based [3]–[7] and semantic-based [8]–[13] methods. These geometric-based approaches cannot remove all potential dynamic objects, e.g., people who are sitting. Features on such objects are unreliable and also need to be removed from tracking and mapping. These semantic-based methods use semantic segmentation or object detection approaches to obtain pixel-wise masks or bounding box of potential dynamic objects. Sitting people can be detected and removed from tracking and mapping using the semantic information and a map of static objects can be built. Usually, in semantic-based methods, geometric check, such as Random Sample Consensus (RANSAC) [14] and multi-view geometry, are also used to remove outliers.

These semantic-based methods first detect or segment objects and then remove outliers from tracking. The tracking thread has to wait for semantic information before tracking

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

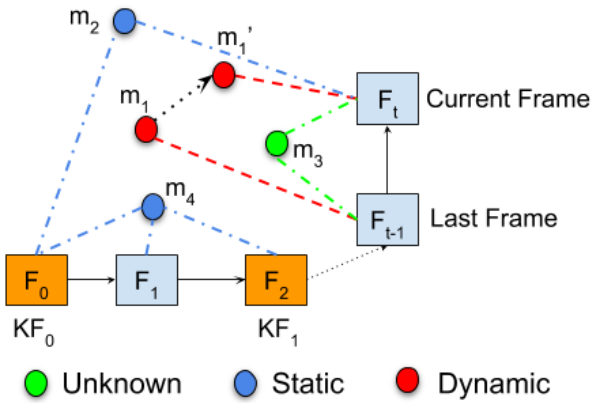


FIGURE 1. Example of data association in vSLAM under dynamic scene. F_t, ($t \geq 0$) is the frame and KF_t is the selected keyframe. m_i, $i \in \{0, 1, \ldots\}$ is the map point. Assume m_1 moved to new position m_1' because it belongs to a moving object. The red line indicates the unstable or bad data association.

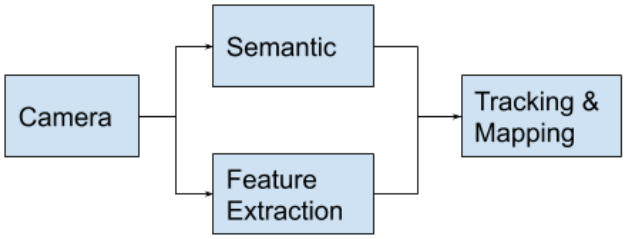


FIGURE 2. Blocked model. Semantic model can use different kinds of segmentation methods, e.g., Mask R-CNN and SegNet. Note that this is not exactly the same as the semantic-based methods mentioned [8]–[13]. The tracking process is blocked to wait for the results of semantic model.

(camera ego-motion estimation), which is called the *blocked model* in this paper (as shown in Fig. 2). Their processing speed is limited by the time-consuming of semantic segmentation methods used. For example, Mask R-CNN requires about 200ms [15] for segmenting one image and this will limit the real-time performance of the entire system.

Our main challenge is how to execute vSLAM in real-time under dynamic scenes with various pixel-wise semantic segmentation methods that ran at a different speed, such as SegNet and Mask R-CNN. We propose a semantic thread to wait for the semantic information. It runs in parallel with the tracking thread and the tracking thread does not need to wait for the segment result. Therefore, the tracking thread can execute in real-time. We call it a *non-blocked model* in this paper. Faster segmentation methods (e.g., SegNet) can update semantic information more frequently than slower methods (e.g., Mask R-CNN). Although we cannot control the segmentation speed, we can use a strategy to obtain as the latest semantic information as possible to remove outliers from the current frame.

Because the semantic thread runs in parallel with the tracking thread, we use the map points to save and share the semantic information. As shown in Fig. 1, we update and propagate semantic information using the moving probability and classify map points into three categories, static, dynamic, and unknown, according to the moving probability threshold. These classified map points will be used to select as stable data associations as possible in tracking.

The main contributions of this paper are:

(1) we propose a novel semantic-based real-time dynamic vSLAM algorithm, RDS-SLAM, which enables the tracking thread does not need to wait for the semantic results anymore. This method efficiently and effectively uses semantic segmentation results for dynamic object detection and outliers removing while keeping the algorithm's real-time nature.

(2) we propose a keyframe selection strategy that uses as the latest new semantic information as possible for outliers removal with any semantic segmentation methods with different speeds in a uniform way.

(3) We show the real-time performance of the proposed method is better than the existing similar methods using the TUM dataset.

The rest of the paper is structured as follows. Section II discusses related work. Section III describes a system overview. Sections IV, V, and VI detail the implementation of the proposed methods. Section VII shows experimental results, and section VIII presents the conclusions and discusses future work.

II. RELATED WORK

A. VISUAL SLAM

vSLAM [2] can be classified into feature-based methods and direct methods. Mur-Artal *et al.* presented ORB-SLAM2 [16], a complete SLAM system for monocular, stereo, and RGB-D cameras, which works in real-time on standard CPUs in a wide variety of environments. This system estimates the ego-motion of the camera by matching the corresponding ORB [17] features between the current frame and previous frames and has three parallel threads: tracking, local mapping, and loop closing. Carlos *et al.* proposed the latest version ORB-SLAM3 [18], mainly adding two novelties: 1) a feature-based tightly-integrated visual-inertial SLAM that fully relies on maximum-a-posteriori (MAP) estimation; 2) a multiple map system (ATLAS [19]) that relies on a new place recognition method with improved recall. In contrast to features-based methods,. For example, Kerl et al. proposed a dense visual SLAM method, DVO [20], for RGB-D cameras that minimizes both the photometric and the depth error over all pixels. However, none of the above methods can address the common problem of dynamic objects. Detecting and dealing with dynamic objects in a dynamic scene in real-time is a challenging task in vSLAM.

Our work follows the implementation of ORB-SLAM3 [18]. The concepts in ORB-SLAM3: keyframe, covisibility graph, ATLAS and Bundle adjustment (BA), are also used in our implementation.

1) KEYFRAME

Keyframes [18] is a subset of selected frames to avoid unnecessary redundancy in tracking and optimization. Each

keyframe stores 1) a rigid body transformation of the camera pose that transforms points from the world to the camera coordinate system; 2) ORB features, associated or not to a map point. In this paper, keyframes are selected by the same policy as ORB-SLAM3; a keyframe is selected if all the following conditions are met: 1) 20 frames have passed from the last global relocalization or the last keyframe insertion; 2) local mapping thread is idle; 3) current frame tracks at least 50 points or less than 90% points than reference keyframe.

2) COVISIBILITY GRAPH

Covisibility graph [16] is represented as an undirected weighted graph, in which each node is a keyframe and the edge holds the number of commonly observed map points.

3) ATLAS

The Atlas [19] is a multi-map representation that handles an unlimited number of sub-maps. Two kinds of maps, active map and non-active map, are managed in the atlas. When the camera tracking is considered lost and relocalization was failed for a few frames, the active map becomes a non-active map, and a new map will be initialized. In the atlas, keyframes and map points are managed using the covisibility graph and the spanning tree.

4) BUNDLE ADJUSTMENT (BA)

BA [21] is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter estimates. Local BA is used in the local mapping thread to optimize only the camera pose. Loop closing launches a thread to perform full BA after the pose-graph optimization to jointly optimize the camera pose and the corresponding landmarks.

B. GEOMETRIC-BASED SOLUTIONS

Li *et al.* [5] proposed a real-time depth edge-based RGB-D SLAM system for dynamic environments based on the frame-to-keyframe registration. They only use depth edge points which have an associated weight indicating its probability of belonging to a dynamic object. Sun *et al.* [6] classify pixels using the segmentation of the quantized depth image and calculate the difference in intensity between consecutive RGB images. Tan *et al.* [3] propose a novel online keyframe representation and updating method to adaptively model the dynamic environments. The camera pose can reliably be estimated even in challenging situations using a novel prior-based adaptive RANSAC algorithm to efficiently remove outliers.

Although the geometric-based vSLAM solution in dynamic environments can restrict the effect of the dynamic objects to some extent, there are some limitations: 1) they cannot detect the potential dynamic objects that temporarily keep static; 2) lack of semantic information. We cannot judge dynamic objects using priori knowledge of the scene.

C. SEMANTIC-BASED SOLUTIONS

DS-SLAM [10], implemented on ORB-SLAM2 [16], combines a semantic segmentation network (SegNet [22]) with a moving consistency check to reduce the impact of dynamic objects and produce a dense semantic octree map [23]. DS-SLAM assumes that the feature points on the people are most likely to be outliers. If a person is determined to be static, then matching points on the person can also be used to predict the pose of the camera.

DynaSLAM [9], also built on ORB-SLAM2, is robust in dynamic scenarios for monocular, stereo, and RGB-D datasets, by adding the capabilities of dynamic object detection and background inpainting. It can detect the moving objects either by multi-view geometry, deep learning, or both and inpaint the frame background that has been occluded by dynamic objects using a static map of the scene. It uses Mask R-CNN to segment out all the priori dynamic objects, such as people or vehicles. DynaSLAM II [24] tightly integrates the multi-object tracking capability. But this method only works for rigid objects. However, in the dynamic scene of TUM [25] dataset, people change their shape by sometimes standing and sometimes sitting.

Detect-SLAM [12], also built on ORB-SLAM2, integrates visual SLAM with single-shot multi-box detector (SSD) [26] to make the two functions mutually beneficial. They call the probability of a feature point belonging to a moving object the moving probability. They distinguish keypoints into four states, high-confidence static, low-confidence static, low-confidence dynamic, and high-confidence dynamic. Considering the delay of detection and the spatio-temporal consistency of successive frames, they only use the color images of keyframes to detect using SSD, meanwhile propagating probability frame-by-frame in the tracking thread. Once the detection result is obtained, they insert the keyframe into the local map and update the moving probability on the local map. Then they update the moving probability of 3D points in the local map that matched with the keyframe.

DM-SLAM [11] combines Mask R-CNN, optical flow, and epipolar constraint to judge outliers. The Ego-motion Estimation module estimates the initial pose of the camera, similar to the Low-cost tracking module in DynaSLAM. DM-SLAM also uses features in priori dynamic objects, if they are not moving heavily, to reduce the feature-less case caused by removing all priori dynamic objects.

Fan *et al.* [8] proposed a novel semantic SLAM system with a more accurate point cloud map in dynamic environments and they use BlizNet [27] to obtain the masks and bounding boxes of the dynamic objects in the image.

All these methods use the blocked model. They wait for the semantic results of every frame or keyframe before estimating the camera pose. As a result, their processing speed are limited by the specific CNN models they used. In this paper, we propose RDS-SLAM that uses the non-blocked model and shows its real-time performance by comparing it with those methods.

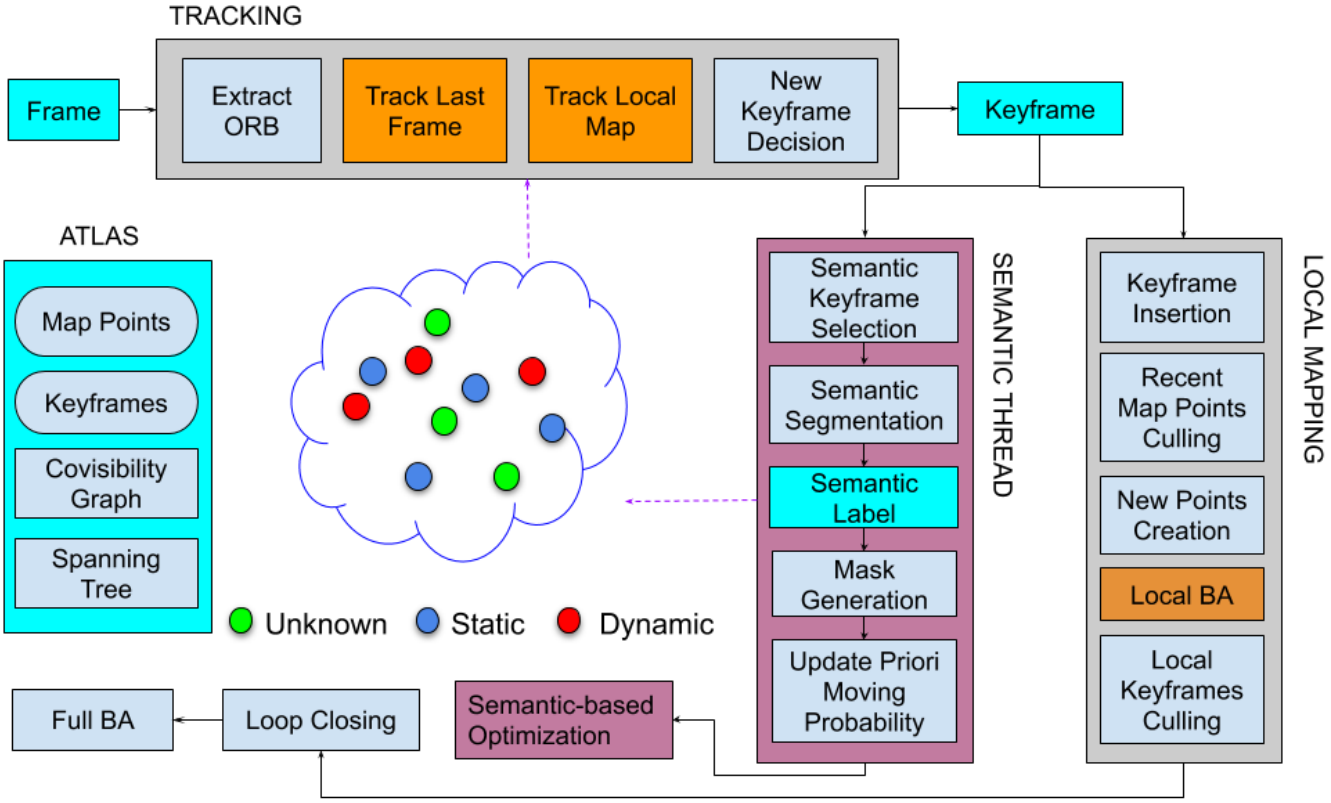


FIGURE 3. System architecture. Models with orange color are modified blocks based on ORB-SLAM3. Models with magenta color are newly added features. Blocks in blue are important data structures.

III. SYSTEM OVERVIEW

Each frame will first pass through the tracking thread. The initial camera pose is estimated for the current frame after being tracked with the last frame and further optimized by being tracked with the local map. Then, keyframes are selected and they are useful in semantic tracking, semantic-based optimization, and local mapping thread. We modify several models in the tracking and the local mapping threads to remove outliers from camera ego-motion estimation using the semantic information. In the tracking thread, we propose a data association algorithm to use as the features on static objects as possible.

The semantic thread runs in parallel with the others, so as not to block the tracking thread and saves the semantic information into the atlas. Semantic labels are used to generate the mask image of the priori dynamic objects. The moving probability of the map points matched with features in the keyframes is updated using the semantic information. Finally, the camera pose is optimized using the semantic information in the atlas.

We will introduce the new features and modified models in the following sections. We skip the detailed explanations of the modules that are the same as those of ORB-SLAM3.

IV. SEMANTIC THREAD

The semantic thread is responsible for generating semantic information and updating it into the atlas map. Before we introduce the detailed implementation of the semantic thread, we use a simple example to explain the general flow, as shown in Fig. 4. Assume the keyframes are selected every two frames. The keyframes are selected by the ORB-SLAM3 and we inserted them into a keyframe list KF sequentially. Assume, at time $t = 12$, KF_2-KF_6 are inside KF. The next step is to select keyframes from KF to request semantic labels from the semantic server. We call this process as *semantic keyframe selection* process in this paper. We take one keyframe from the head of KF (KF_2) and one from the back of KF (KF_6) to request the semantic labels. Then, we calculate the mask of the priori dynamic objects using semantic labels S_2 and S_6. Next, we update the moving probability of map points stored in the atlas. The moving probability will be used later to remove outliers from the tracking thread.

Alg. 1 shows the detailed implementation of the semantic thread. The first step is to select semantic keyframes from keyframe list KF (Line 2). Next, we request semantic labels from the semantic model and return the semantic labels SLs (Line 3). Lines 4-8 are to save and process the semantic results for each item returned. Line 6 is to generate the mask image of dynamic objects and Line 7 updates the moving probability stored in the atlas. We will introduce each submodule of the semantic thread sequentially (see Fig. 3).

A. SEMANTIC KEYFRAME SELECTION ALGORITHM

The semantic keyframe selection algorithm is to select keyframes for requesting the semantic labels later. We need

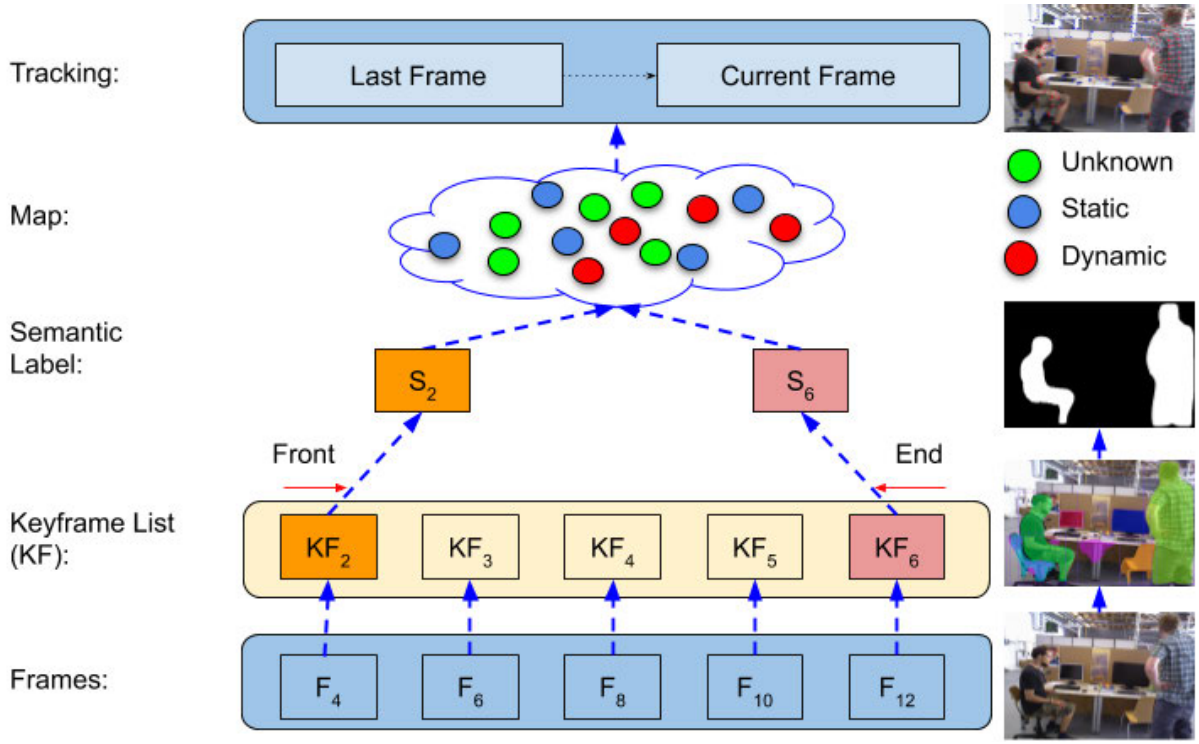


FIGURE 4. Semantic tracking example. Assume keyframes KF_n is selected every two frames F_n and inserted into keyframe list KF. We choose keyframes from KF to request semantic labels S_n. Then we update the moving probability into the atlas using the mask image of dynamic objects that reproduced from the semantic label. Blue circles stand for the static map points and red circles for the dynamic map points. Others marked in green are unknown.

Algorithm 1 Semantic Tracking Thread

Require: KeyFrame list: KF
1: **while** not_request_finish() **do**
2: SK = semantic_keyframe_selection(KF)
3: SLs = request_segmentation(SK)
4: **for** $i = 0; i < SLs.size(); i++$ **do**
5: KeyFrame kf = SR[i]
6: kf->mask = GenerateMaskImage(SLs[i])
7: kf->UpdatePrioriMovingProbability()
8: **end for**
9: **end while**

to keep the real-time performance while using different kinds of semantic segmentation methods. However, some of them, such as Mask R-CNN, are time-consuming and the current frame in tracking may not obtain the new semantic information if we segment every keyframe sequentially.

To evaluate the distance quantitatively, we define the *semantic delay* that is the distance between the latest frame id which has the semantic label (S_t) that holds the latest semantic information and the current frame (F_t) id, as follows:

$$d = FrameID(F_t) - FrameID(S_t). \quad (1)$$

Fig. 5 shows the semantic delay for several cases. The general idea is to segment each frame or keyframe sequentially, according to the time sequence as shown in Fig. 5 (a). We call this kind of model the *sequential segmentation model*. However, this will monotonically increase the time delay when using time-consuming segmentation methods as shown as the blue line in Fig. 6. For instance, at time $t = 10$ (F_{10}), the semantic model completed the segmentation of KF_0 (F_0) and the semantic delay is $d = 10$. Similarly, at time 40 (F_{40}), the semantic delay becomes 34. That is, the last frame that has semantic information is 34 frames behind the current frame. The current frame cannot obtain the latest semantic information.

To shorten the distance, supposed that we segment two frames sequentially at the same time (Fig. 5 (b)). Then, the delay becomes $12-2 = 10$ if KF_0 and KF_1 are segmented at the same time. The delay still grows linearly as shown as the red line in Fig. 6.

To further shorten the semantic delay, we use a *bi-directional model*. We do not segment keyframes sequentially. Instead, we do semantic segmentation using keyframes both from the front and back of the list to use as the latest semantic information as possible, as shown in Fig. 5 (c) and as the yellow line in Fig. 6. The semantic delay becomes a constant value. In practice, the delay in the bidirectional model is not always 10. The distance is influenced by the segmentation method used, the frequency of keyframe selection, and the processing speed of the related threads.

The left side of Fig. 7 indicates a semantic keyframe selection example and the right side of Fig. 7 shows the timeline of requesting semantic information from the semantic model/server. We take both keyframes from the head and

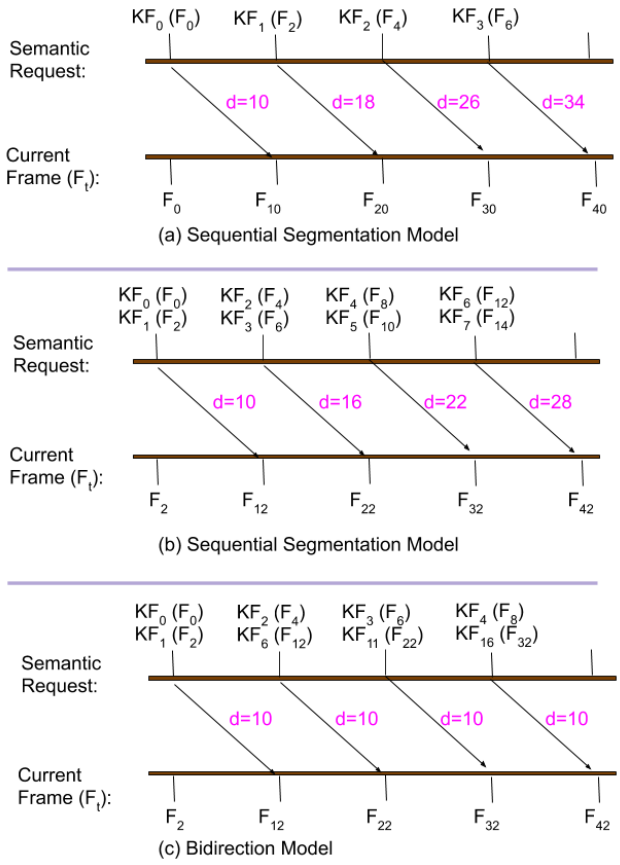


FIGURE 5. Bi-direction model vs sequential model. Assume we use Mask R-CNN (200ms) and ORB-SLAM3 (20ms), and the keyframe is selected every two frames. About 200/20 = 10 frames delay while waiting for the semantic result.

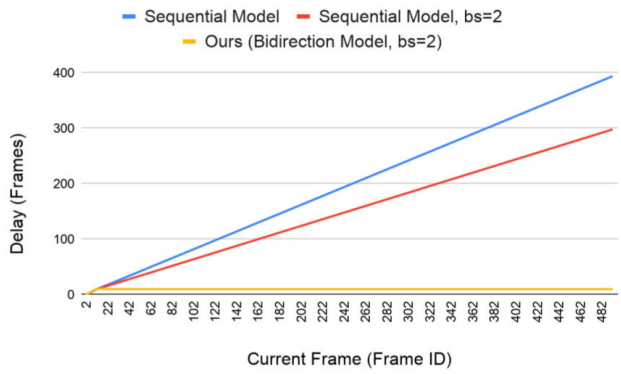


FIGURE 6. Semantic delay of sequential model vs bi-direction model.

back of KF to request the semantic label. (Round 1) At time $t = 2$, two keyframes KF_0 and KF_1 are selected. Segmentation finished at $t = 12$. By this time, new keyframes are selected and then inserted into KF (see Round 2). Then we take two elements KF_2 from the front and KF_6 from this back to request the semantic label. At the time $t = 22$, we received the semantic result and continue the next round (Round 3).

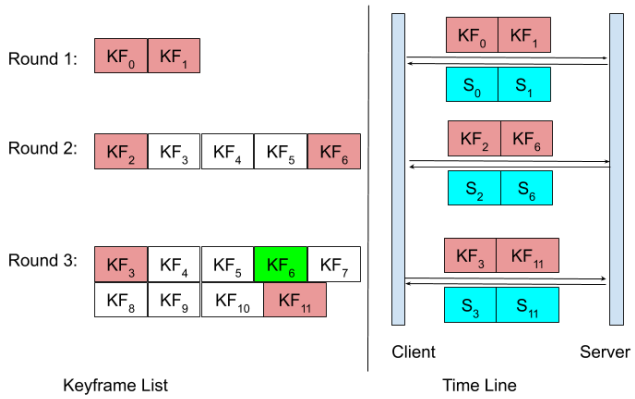


FIGURE 7. Semantic time line. The left side is the contents inside the keyframe list *KF* and right side is the time line of requesting semantic label. Keyframe in green color means this item has already obtained the semantic information in the previous round.

We can obtain relatively new information if we segment the keyframe at the tail of the KF list. Then why do we also need to segment the keyframe that in front of the list? Different from the blocked model, there is no semantic information for the first few frames (about 10 frames if use Mask R-CNN) in our method. Since the processing speed of the tracking thread is usually faster than the semantic thread, vSLAM may have already accumulated large errors because of the dynamic objects. Therefore, we need to correct these drift errors using the semantic information by popping out and feeding the keyframes in the front of the KF list sequentially to the semantic-based optimization thread to correct/optimize the camera poses.

B. SEMANTIC SEGMENTATION

In our experiment, we use two models with different speeds, Mask R-CNN (slower) and SegNet (faster), as shown in Fig. 8. Mask R-CNN [15] is trained with the MS COCO [28], which has both pixel-wise semantic segmentation results and instance labels. We implemented it based on the TensorFlow version of Matterport.[1] SegNet [22] implemented using Caffe,[2] is trained with the PASCAL VOC [29] 2012 dataset, where 20 classes are offered. We did not refine the network using the TUM dataset because SLAM usually runs in an unknown environment.

C. SEMANTIC MASK GENERATION

We merge all the binary mask images of instance segmentation results into one mask image that is used to generate the mask image (Fig. 8) of people. Then we calculate the priori moving probability of map points using the mask. In practice, since the segmentation on object boundaries are sometimes unreliable, the features on the boundaries cannot be detected if directly apply the mask image, as shown in Fig. 9 (a).

[1]https://github.com/matterport/Mask_RCNN
[2]https://github.com/alexgkendall/SegNet-Tutorial

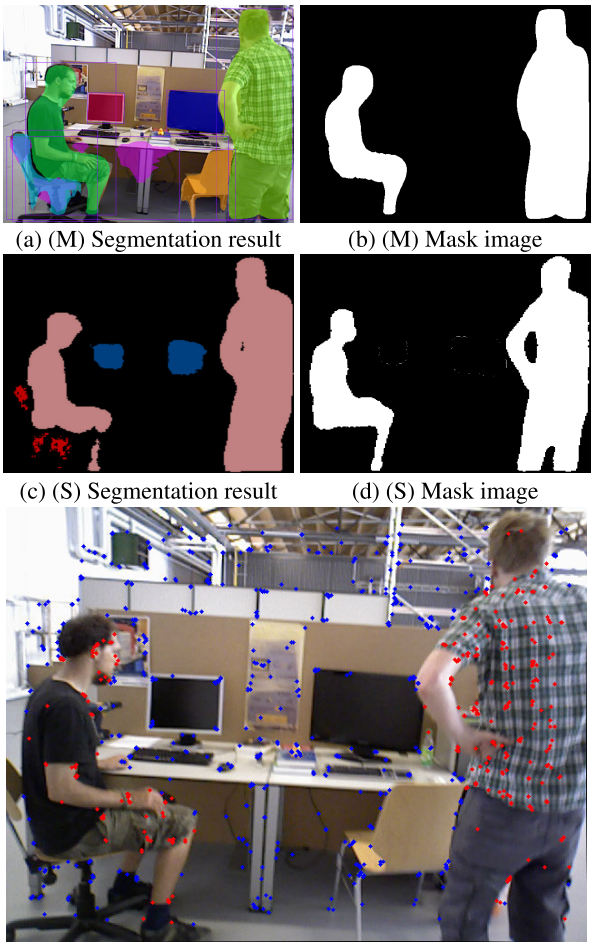


FIGURE 8. Semantic information. "M" stands for Mask R-CNN and "S" for "SegNet". (e) shows the outliers that marked in red color, which are detected using the mask image.

FIGURE 9. Mask dilation. Remove outliers on the edge of dynamic objects.

Therefore, we dilate the mask using a morphological filter to include the edge of dynamic objects, as shown in Fig. 9 (b).

D. MOVING PROBABILITY UPDATE

In order not to wait for the semantic information in the tracking thread, we isolate the semantic segmentation from tracking. We use the moving probability to convey semantic information from semantic thread to tracking thread. The moving probability is used to detect and remove outliers from tracking.

FIGURE 10. Segmentation failure case. Some features on the body on the person (a) cannot be identified as outliers using unsound mask (c) generated by semantic result (b). Therefore, those features are wrongly labeled as static in this frame.

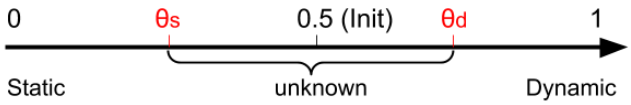


FIGURE 11. Moving probability. θ_s is the static threshold and θ_d is the dynamic threshold value.

1) DEFINITION OF MOVING PROBABILITY

As we know, vSLAM is usually running in an unknown environment, the semantic result is not always robust if the CNN network is not well trained or refined according to the current environment (Fig. 10). To detect outliers, it is more reasonable to consider the spatio-temporal consistency of frames, rather than just use the semantic result of one frame. Therefore, we use the moving probability to leverage the semantic information of successive keyframes.

We define the moving probability ($p(m_t^i)$, $m_t^i \in M$) of each map point i at the current time as shown in Fig. 11. The status of the map point is more likely dynamic if its moving probability is closer to one. The more static the map point is if it is more closer to zero. To simplify, we abbreviate the moving probability of map point i at time t ($p(m_t^i)$) to $p(m_t)$. Each map point has two status (M), dynamic and static, and the initial probability (initial belief) is set to 0.5 ($bel(m_0)$).

$$M = \{static(s), dynamic(d)\}$$
$$bel(m_0 = d) = bel(m_0 = s) = 0.5$$

2) DEFINITION OF OBSERVED MOVING PROBABILITY

Considering the fact that the semantic segmentation is not 100% accurate, we define the observe moving probability as:

$$\begin{aligned} p(z_t = d|m_t = d) &= \alpha, \\ p(z_t = s|m_t = d) &= 1 - \alpha, \\ p(z_t = s|m_t = s) &= \beta, \ and \\ p(z_t = d|m_t = s) &= 1 - \beta. \end{aligned}$$

The values α and β are manually given and it is related to the accuracy of semantic segmentation. In the experiment, we set α and β to 0.9 by supping the semantic segmentation is fairly reliable.

3) MOVING PROBABILITY UPDATE

The moving probability of the current time $bel(m_t)$ is predicted based on the observation $z_{1:t}$ (semantic segmentation) and initial status m_0. We formulate the moving probability updating problem as a Bayesian filter [30] problem:

$$\begin{aligned} bel(m_t) &= p(m_t|z_{1:t}, m_0) \\ &= \eta p(z_t|m_t, z_{1:t-1}, m_0)p(m_t|z_{1:t-1}, m_0) \\ &= \eta p(z_t|m_t)p(m_t|z_{1:t-1}, m_0) \\ &= \eta p(z_t|m_t)\overline{bel}(m_t) \end{aligned} \tag{2}$$

In Eq. 2 exploits Bayes rule and the conditional independence that the current observation z_t only relies on the current status m_t. η is a constant. The prediction $\overline{bel}(m_t)$ is calculated by:

$$\begin{aligned} \overline{bel}(m_t) &= \int p(m_t|m_{t-1}, z_{1:t-1})p(m_{t-1}|z_{1:t-1})dm_{t-1} \\ &= \int p(m_t|m_{t-1})bel(m_{t-1})dm_{t-1} \end{aligned} \tag{3}$$

In Eq. (3), we exploit the assumption that our state is complete. This implies if we know the previous state m_{t-1}, past measurements convey no information regarding the state m_t. We assume the state transition probability $p(m_t = d|m_{t-1} = s) = 0$ and $p(m_t = d|m_{t-1} = d) = 1$ because we cannot detect the suddenly change of objects. η is calculated by $(bel(m_t = d)+bel(m_t = s))/2$. The probability of map points belonging to dynamic is calculated by:

$$\begin{aligned} &\overline{bel}(m_t = d) \\ &= p(m_t = d|m_{t-1} = d)bel(m_{t-1} = d) \end{aligned} \tag{4}$$

4) JUDGEMENT OF STATIC AND DYNAMIC POINTS

Whether a point is dynamic or static is judged using predefined probability thresholds, θ_d and θ_s (see Fig. 11). They are set to 0.6 and 0.4 respectively in the experiment.

$$Status(m_t^i) = \begin{cases} dynamic & p(m_t) > \theta_d \\ static & p(m_t) < \theta_s \\ unknown & others \end{cases} \tag{5}$$

V. TRACKING THREAD

The tracking thread runs in real-time and tends to accumulate the drift error due to the incorrect or unstable data associations of 3D map points and 2D features in each frame caused by dynamic objects. We modify the Track Last Frame model and Track Local Map model of ORB-SLAM3 tracking thread to remove outliers (see Fig. 3). We propose a data association algorithm that uses as good data associations as possible using the moving probability stored in the atlas.

A. TRACK LAST FRAME

Alg. 2 shows the data association algorithm in tracking last frame model. For each feature i in the last frame, we first get their matched map point m (Line 2). Next, we find the matched feature in the current frame by comparing the descriptor distance of ORB features (Line 3). After that,

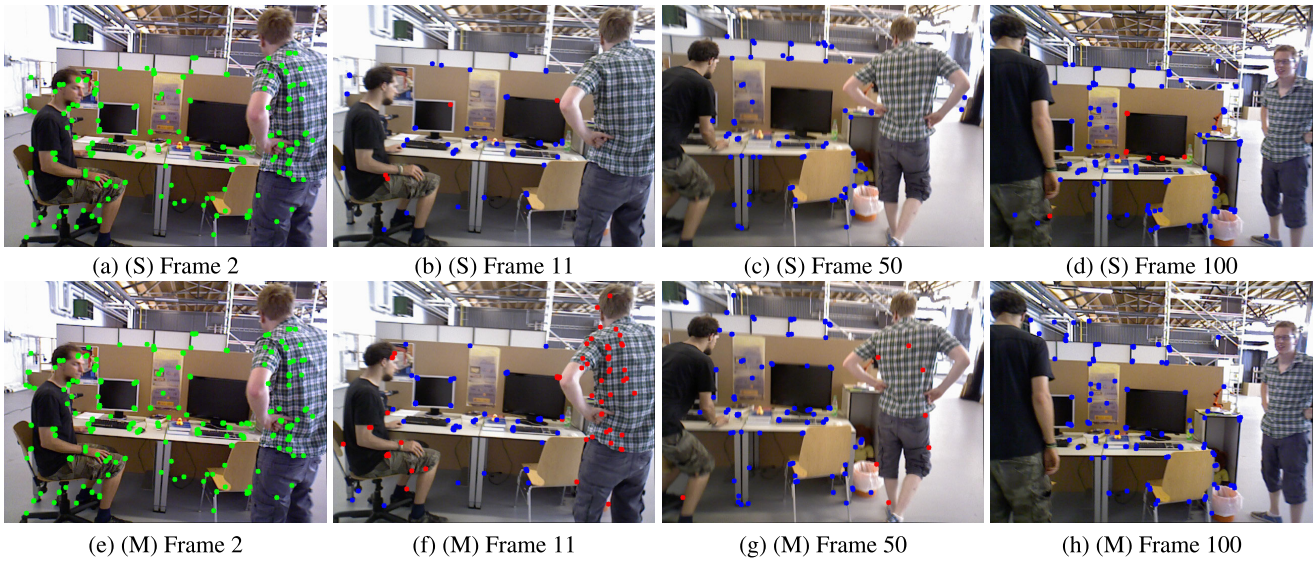
Algorithm 2 Robust Data Association Algorithm

Require: Current Frame: F_t
Last Frame: F_{t-1}
Unknown subset: Unknown<FeatureId, MapPoint*>
Static subset: Static<FeatureId, MapPoint*>
Threshold: θ_d, θ_s, $\tau = 20$

```
1:  for i = 0; i < F_{t-1}.Features.size(); i++ do
2:    MapPoint* m = F_{t-1}.MapPoints[i]
3:    f = FindMatchedFeatures(F_t, m)
4:    if p(m) > θ_d then
5:      continue
6:    end if
7:    if p(m) < θ_s then
8:      Static.insert(f, m)
9:    end if
10:   if θ_d ≤ p(m) ≤ θ_s then
11:     Unknown.insert(f, m)
12:   end if
13: end for
14: for it = Static.begin(); it! = Static.end();it ++ do
15:   F_t.MapPoints[it->first] = it->second;
16: end for
17: if Static.size()<τ then
18:   for it = Unknown.begin(); it!=Unknown.end();it ++ do
19:     F_t.MapPoints[it->first] = it->second;
20:   end for
21: end if
```

in order to remove the bad influence from dynamic map points, we skip those map points that have higher moving probability (Lines 4-6). Then, there are two kinds of map points left, static and unknown map points. We want to use only the static map points as far as we can. Therefore, we classify the remaining map points into two subsets: static subset and unknown subset, according to their moving probability (Lines 7-12). Finally, we use the selected relative good matches. We first use all the good data stored in static subset (Lines 14-16). If the size of these data is not enough (less than the threshold $\tau = 20$, the value used in ORB-SLAM3), we also use the data in unknown subset (Lines 17-21).

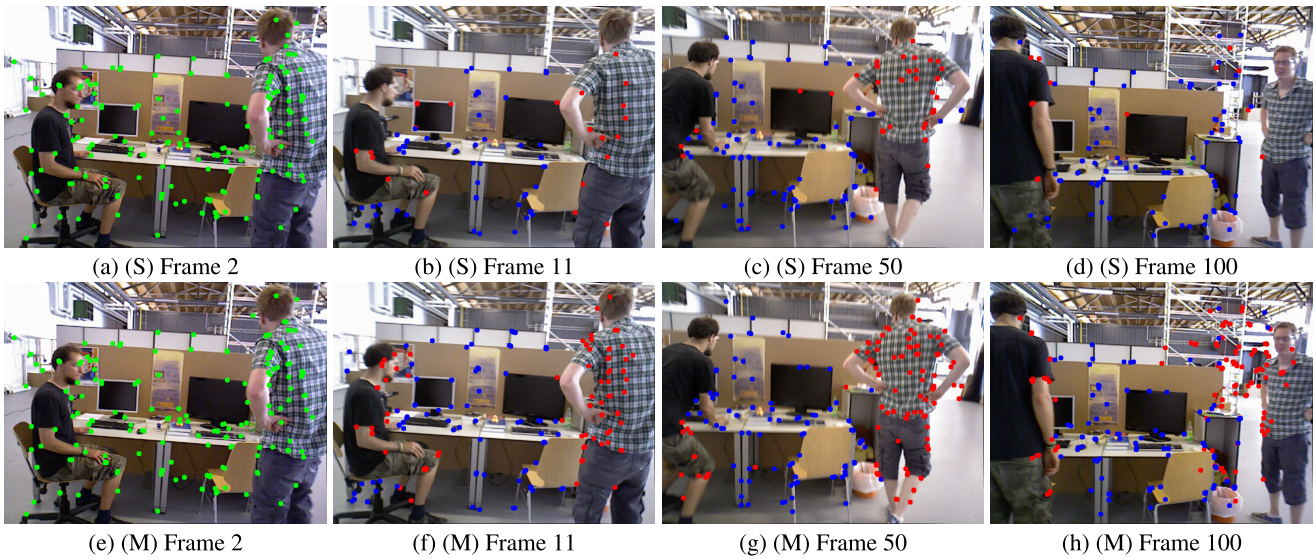
We try to exclude outliers from tracking using the moving probability stored in the atlas. How well the outliers are removed will have a great influence on the tracking accuracy. We show the results of a few frames in Fig. 12. All the features in the first few frames are in green color because no semantic information can be used and the moving probability of all map points is 0.5, the initial value. The features in red belong to dynamic objects and they are hard to match with the last frame than static features (blue features). The green features are almost disappeared because the map points obtained the semantic information over time. We only use features in the static subset if its size number is enough to estimate camera ego-motion.



(a) (S) Frame 2 (b) (S) Frame 11 (c) (S) Frame 50 (d) (S) Frame 100

(e) (M) Frame 2 (f) (M) Frame 11 (g) (M) Frame 50 (h) (M) Frame 100

FIGURE 12. Results after tracking last frame. "M" stands for Mask R-CNN an "S" for SegNet. The features in red color are not used in tracking. Blue features belong to the static subset and green features belong to the unknown subset.



(a) (S) Frame 2 (b) (S) Frame 11 (c) (S) Frame 50 (d) (S) Frame 100

(e) (M) Frame 2 (f) (M) Frame 11 (g) (M) Frame 50 (h) (M) Frame 100

FIGURE 13. Results after tracking local map. "M" stands for Mask R-CNN and "S" for SegNet.

B. TRACK LOCAL MAP

The basic idea of the data association algorithm in the Tracking Local Map model is similar with Alg. 2. The difference is that here we use all the map points in the local map to find good data association. The data association result after tracking local map is shown in Fig. 13. More map points are used to match in this model than the tracking last frame. The features on the people are almost successfully detected or not matched/used.

VI. OPTIMIZATION

A. SEMANTIC-BASED OPTIMIZATION

We optimize the camera pose using the keyframes given by the semantic keyframe selection algorithm. Considering that the tracking thread runs very fast than the semantic thread, drifts have already accumulated to some extent with the influence of dynamic objects. Therefore, we try to correct the camera pose using semantic information. We modify the error term used in ORB-SLAM3 by using the moving probability of map points for weighting, as shown below. In the experience, we only use the matched static map points for optimization.

Assume $X_j^w \in \mathbb{R}^3$ is the 3D pose of a map point j in the world coordinate system. The i-th keyframe pose in the world coordinate is $T_i^w \in SE(3)$. The camera pose T_i^w is optimized by minimizing the reprojection error concerning the matched keypoint $x_{ij} \in \mathbb{R}^2$ of the map point. The error term for the observation of a map point j in a keyframe i is:

$$e(i,j) = (x_{ij} - \pi_i(T_i^w, X_j^w))(1 - p(m^j)), \tag{6}$$

where π_i is the projection function that projects a 3D map point into a 2D pixel point in the keyframe i. The larger the moving probability is, the smaller contribution to the error. The cost function to be optimized is:

$$C = \sum_{i,j} \rho(e_{i,j}^T \Omega_{i,j}^{-1} e_{i,j}) \tag{7}$$

where ρ is the Huber robust cost function and $\Omega_{i,j}^{-1}$ is the covariance matrix.

B. BUNDLE ADJUSTMENT IN LOCAL MAPPING THREAD

We modify the local BA model to reduce the influence of dynamic map points using semantic information. What we modified are: 1) the error term, in which the moving probability is used, as shown in Eq. 6; 2) only keyframes that already obtained semantic information are used for BA.

VII. EXPERIMENTAL RESULTS

We evaluate the tracking accuracy using TUM [25] indoor dataset and demonstrate the real-time performance by comparing with state-of-the-art vSLAMs methods using, when possible, the results in the original papers.

A. SYSTEM SETUP

Our system is evaluated using GeForce RTX 2080Ti GPU, Cuda 11.1, and docker. [3] Docker is used to deploy different kinds of semantic segmentation methods on the same machine. We also use Kinect v2 [4] camera to evaluate in real environment.

B. TRACKING ACCURACY EVALUATION

The proposed method was compared against the ORB-SLAM3 and similar semantic-based algorithms to quantify the tracking performance of our proposal in dynamic scenarios.

The TUM RGB-D dataset contains color and depth images along the ground-truth trajectory of the sensor. In the sequence named "*fr3/walking_**" (labeled as f3/w/*), two people walk through an office. This is intended to evaluate the robustness of vSLAM in the case of quickly moving dynamic objects in large parts of a visible scene. Four types of camera motion are included in *walking* data sequences 1) "*xyz*", the Asus Xtion camera is manually moved along three directions (xyz); 2) "*static*", where the camera is kept in place manually; 3) "*halfsphere*", where the camera is moved on a small half-sphere of approximately one-meter diameter; 4) "*rpy*", where the camera is rotated along the principal axes *(roll-pitch-yaw)*. In the experiment, the person is dealt with as the only priori dynamic object in the TUM dataset.

We compared the trajectory of camera with ORB-SLAM3,[5] DS-SLAM,[6] and DynaSLAM. [7] Fig. 14 compares the obtained trajectories using their source codes and therefore the trajectories are not exactly the same as the ones in their original paper. We evaluated our system using both Mask R-CNN (M) and SegNet (S). The trajectory of DynaSLAM that use Mask R-CNN is very similar with our Mask R-CNN version as shown in Fig. 14 (m-p) and Fig. 14 (q-t). The performance of our SegNet version (Fig. 14 (i and j)) is similar to the DS-SLAM (Fig. 14 (e and f)).

The error in the estimated trajectory was calculated by comparing it with the ground truth, using two prominent measurements: absolute trajectory error (ATE) and relative pose error (RPE) [25], which are well-suited for measuring the performance of the vSLAM. The root mean squared error (RMSE), and the standard deviation (S.D.) of ATE and RPE are compared. Each sequence was run at least five times as dynamic objects are prone to increase the non-deterministic effect. We compared our method with ORB-SLAM3 [18], DS-SLAM [10], DynaSLAM [9], SLAM-PCD [8], DM-SLAM [11], and Detect-SLAM [12]. The comparison results are summarized in Tables 1, 2, and 3. DynaSLAM reported they obtained the best performance using the combination of Mask R-CNN and geometric model. In this paper, we mainly focus on the time cost problem caused by semantic segmentation. Contrary to the very heavy geometric model that DynaSLAM used, we only use the very light geometric check, such as RANSAC, photometric error to deal with the outliers that not rely on the priori dynamic objects.

Our proposal outperforms the original ORB-SLAM3 (RGB-D mode only without IMU) and obtains similar performance with DynaSLAM, SLAM-PCD, and DM-SLAM, in which the tracking error is already very small. Different from them, we use the non-blocked model. The first few frames do not have any semantic information. The number of keyframes that have a semantic label is smaller than suing the blocked model because the processing speed of the tracking thread is much faster than the semantic segmentation (especially for the heavy model, Mask R-CNN). However, we achieved a similar tracking performance using less semantic information.

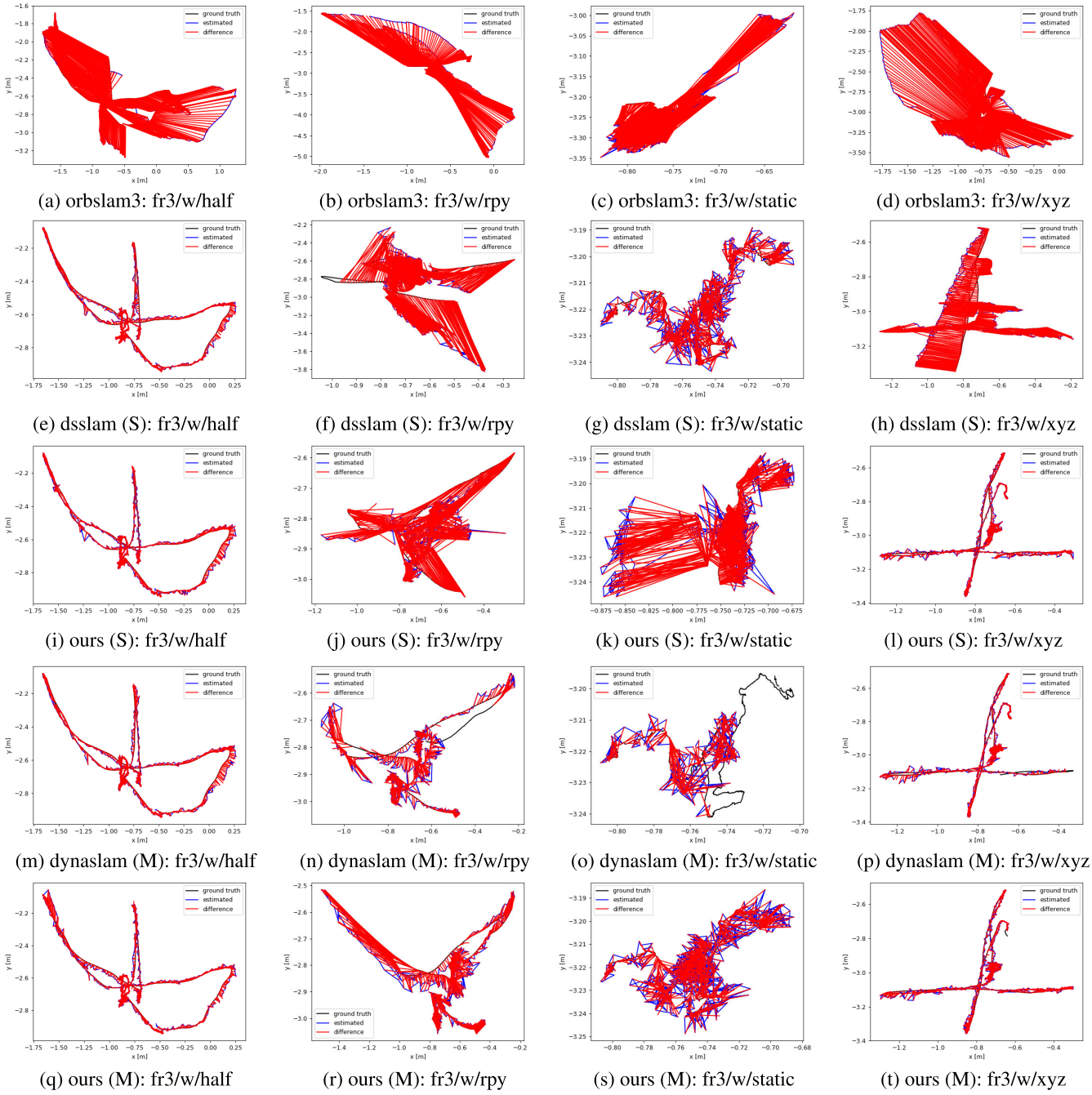
C. REAL ENVIRONMENT EVALUATION

We test our system using Kinect2 RGB-D camera, as shown in Fig. 15. All the features are in initial status when in the first few frames because they have not yet obtained any semantic information. The static features will be increasingly detected over time and used to estimate camera pose. The features on the person is detected and excluded from tracking. The algorithm runs in around 30HZ, as shown in Table 4.

D. EXECUTION TIME

Tab. 4 compares the execution time of vSLAM algorithms. In the blocked model, the tracking thread needs to wait for the semantic label. The speed of the other methods is related to the semantic segmentation methods used. The heavy the

[3] https://docs.docker.com/
[4] https://github.com/code-iai/iai_kinect2
[5] https://github.com/UZ-SLAMLab/ORB_SLAM3
[6] https://github.com/ivipsourcecode/DS-SLAM
[7] https://github.com/BertaBescos/DynaSLAM



(a) orbslam3: fr3/w/half (b) orbslam3: fr3/w/rpy (c) orbslam3: fr3/w/static (d) orbslam3: fr3/w/xyz

(e) dsslam (S): fr3/w/half (f) dsslam (S): fr3/w/rpy (g) dsslam (S): fr3/w/static (h) dsslam (S): fr3/w/xyz

(i) ours (S): fr3/w/half (j) ours (S): fr3/w/rpy (k) ours (S): fr3/w/static (l) ours (S): fr3/w/xyz

(m) dynaslam (M): fr3/w/half (n) dynaslam (M): fr3/w/rpy (o) dynaslam (M): fr3/w/static (p) dynaslam (M): fr3/w/xyz

(q) ours (M): fr3/w/half (r) ours (M): fr3/w/rpy (s) ours (M): fr3/w/static (t) ours (M): fr3/w/xyz

FIGURE 14. Trajectory comparing frame by brame. "M" stands for "Mask R-CNN" and "S" for "SegNet".

TABLE 1. Results of absolute trajectory error of TUM (m). Ours (1) and (3) are evaluated results only using keyframes.

Seq.	ORB SLAM3		DS-SLAM		DynaSLAM)		SLAM-PCD		DM-SLAM		Detect-SLAM	Ours (1) KeyFrame (Mask R-CNN)		Ours (2) All (Mask R-CNN)		Ours (3) KeyFrame (SegNet)		Ours (4) All (SegNet)	
	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.
w/half	0.6572	0.3124	0.0303	0.0159	0.0296	0.0157	**0.0241**	0.0122	0.0274	0.0137	0.0514	0.0306	0.0171	0.0259	0.0141	0.0291	0.0143	0.0807	0.0454
w/rpy	1.0197	0.5122	0.4442	0.2350	0.0354	0.019	0.0453	0.0316	0.0328	0.0194	0.2959	0.0587	0.0380	0.1468	0.1051	**0.0128**	0.0081	0.1604	0.0873
w/static	0.3614	0.1522	0.0081	0.0033	**0.0068**	0.0032	0.0077	0.0039	0.0079	0.0040	-	0.0720	0.0343	0.0815	0.0224	0.0215	0.0104	0.0206	0.012
w/xyz	0.9178	0.4859	0.0247	0.0161	0.0164	0.0086	0.0157	0.0084	**0.0148**	0.0072	0.0241	0.0240	0.0139	0.0213	0.0127	0.0565	0.0184	0.0571	0.0229
s/static	0.0090	0.0043	0.0065	0.0033	0.0108	0.0056	0.0080	0.0037	**0.0063**	0.0032	-	0.0084	0.0043	0.0088	0.0043	0.0039	0.0017	0.0084	0.0043

semantic model used, the higher the total time consuming is. Although DynaSLAM achieved good tracking performance, the processing time is long due to Mask R-CNN. As we known, DynaSLAM is not a real-time algorithm. DS-SLAM is the second fastest algorithm because it uses a lightweight semantic segmentation method, SegNet. However, the

TABLE 2. Results of translational relative pose error (RPE) (m). Ours (1) and (3) are evaluated results only using keyframes.

Seq.	ORB SLAM3		DS-SLAM		DynaSLAM		SLAM-PCD		Ours (1) KeyFrame (Mask R-CNN)		Ours (2) All (Mask R-CNN)		Ours (3) KeyFrame (SegNet)		Ours (4) All (SegNet)	
	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.
w/half	0.3262	0.2625	0.0297	0.0152	0.0284	0.0149	0.0274	0.0140	0.0332	0.0208	0.0282	0.0155	**0.0274**	0.0140	0.0482	0.036
w/rpy	0.4368	0.3197	0.1503	0.1168	0.0448	0.0262	0.0616	0.0357	0.0700	0.0488	0.1114	0.0920	**0.0245**	0.0122	0.1320	0.1067
w/static	0.7800	0.7563	0.0102	0.0038	**0.0089**	0.0044	0.0102	0.0049	0.0529	0.0444	0.0419	0.0348	0.0221	0.0149	0.0221	0.0149
w/xyz	0.4258	0.3063	0.0333	0.0229	0.0217	0.0119	**0.0204**	0.0107	0.0299	0.4943	0.0281	0.0167	0.0269	0.0163	0.0426	0.0317
s/static	0.0102	0.0049	0.0078	0.0038	0.0126	0.0067	0.0087	0.0038	0.0097	0.0052	0.0107	0.0050	**0.0050**	0.0026	0.0123	0.0070

TABLE 3. Results of rotational relative pose error (RPE) (m). Ours (1) and (3) are evaluated results only using keyframes.

Seq.	ORB SLAM3		DS-SLAM		DynaSLAM		SLAM-PCD		Ours (1) KeyFrame (Mask R-CNN)		Ours (2) All (Mask R-CNN)		Ours (3) KeyFrame (SegNet)		Ours (4) All (SegNet)	
	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.	RMSE	S.D.
w/half	7.2352	5.9487	0.8142	0.4101	0.7842	0.4012	0.7440	0.3459	0.8194	0.4858	0.8216	0.4347	**0.7332**	0.3712	1.8828	1.5250
w/rpy	8.7683	6.4583	3.0042	2.3065	0.9894	0.5701	1.3831	0.8318	1.4736	1.0262	9.3192	8.5720	**0.4973**	0.2586	13.1693	12.0103
w/static	6.0054	5.5995	0.2690	0.1215	**0.2612**	0.1259	0.2631	0.1119	1.4966	1.2839	1.1686	0.9917	0.4944	0.3112	0.4944	0.3112
w/xyz	7.8974	5.5917	0.8266	0.2826	0.6284	0.3848	**0.6227**	0.3807	0.7739	0.4943	0.7236	0.4435	0.7768	0.4886	0.9222	0.6509
s/static	0.3007	0.1300	0.2735	0.1215	0.3416	0.1642	0.2782	0.1210	0.3217	0.1522	0.3091	0.1325	**0.1520**	0.0821	0.3338	0.1706

TABLE 4. The execution time comparison of TUM Dataset. We use the data in their original paper as possible. If not provide in their papers, we approximate the processing time.

Method	GPU	Semantic	Segmentation/Detection Time (ms)	Time of Other Models Related to Tracking (ms)		Track Each Frame (ms)
ORB-SLAM3	-	-	-	-		22 - 30
Detect-SLAM	Nvidia GPU GTX960M	SSD	310	Propagation: Updating:	20 10	> 310
DS-SLAM	P4000	SegNet	37.57330	ORB feature extraction: Moving consistency check:	9.375046 29.50869	> 65
DynaSLAM	Nvidia Tesla M40 GPU	Mask R-CNN	195	Multi-view Geometry: Background Inpainting:	235.98 (w/rpy) 183.56 (w/rpy)	> 300
DM-SLAM	GeForce GTX 1080 Ti	Mask R-CNN	201.02	Ego-motion: Dynamic Point Detection:	3.16 40.64	> 201
Ours (1) TUM	GeForce RTX 2080Ti	Mask R-CNN	200	Mask Generation: Update Moving Probability: Semantic-based Optimization:	5.42 0.17 0.54	50 - 65 (15HZ)
Ours (2) TUM	GeForce RTX 2080Ti	SegNet	30	Mask Generation: Update Moving Probability: Semantic-based Optimization:	5.04 0.17 0.50	50 - 65 (15HZ)
Ours (3) Kinect v2	GeForce RTX 2080Ti	Mask R-CNN	200	Mask Generation: Update Moving Probability: Semantic-based Optimization:	9.01 0.15 0.38	22 - 30 (30HZ)
Ours (4) Kinect v2	GeForce RTX 2080Ti	SegNet	30	Mask Generation: Update Moving Probability: Semantic-based Optimization:	9.05 0.18 0.45	22 - 30 (30HZ)

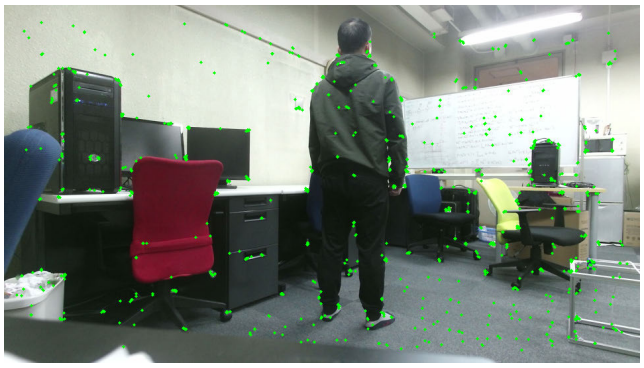
architecture used is also a blocked model. The execution time will increase if a more time-consuming method is used. Our method uses the non-blocked model and runs almost at a constant speed regardless of the segmentation methods.

We evaluate the error metric of TUM dataset using 15HZ by manually adding some time delay in the tracking thread because TUM dataset is very short. Very small semantic information can be obtained in this short time. We compare the time and the number of keyframes that obtained semantic label (Semantic keyframe Number) in Tab. 5. We only compared the Mask R-CNN version because SegNet is faster and it can segment almost all the keyframes in each dataset. We assume the time cost of Mask R-CNN is 0.2s for segmenting each frame. The total time of running the fr3/w/xyz dataset is about 57.3s for 15HZ, however, only 28.3s for 30HZ. In this short time, the number of semantic keyframes in 30HZ (143) is two times smaller than 15HZ (286). Usually, the more keyframes are segmented, the better tracking accuracy can be achieved. This depends on the specific application and the segmentation methods used.

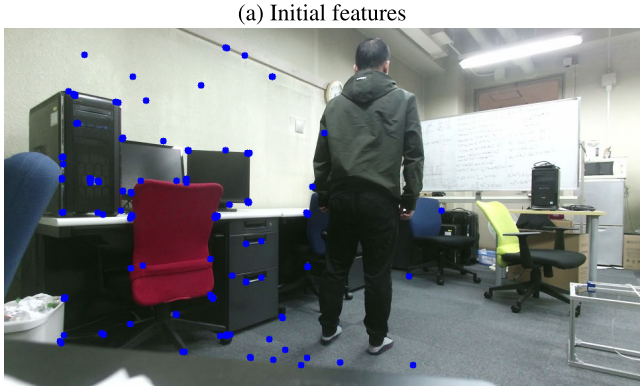
TABLE 5. Semantic keyframe number comparison (Mask R-CNN).

	Total Frames	15HZ		30HZ	
		Total Time (s)	Semantic KeyFrame Num.	Total Time (s)	Semantic KeyFrame Num.
walk_xyz	859	57.3	286	28.6	143
walk_rpy	910	60.7	303	30.3	151
walk_half	1067	71.1	355	35.6	178
walk_sit	707	47.1	235	23.6	118

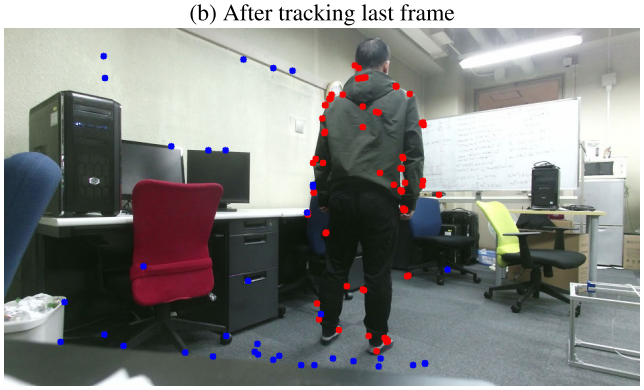
In the bi-direction model, we selected two keyframes at the same time. We offered two strategies to segment them: 1) infer images at the same time as a batch on the same GPU, 2) infer images on the same GPU sequentially (one by one).



(a) Initial features



(b) After tracking last frame



(c) After tracking local map

FIGURE 15. Result of real environment. The green features are in initial status and their moving probability is 0.5. The blue features are static features and the red are outliers. (a) is the original detected ORB features. (b) is the output after tracking last frame process and (c) is the result after tracking local map process.

We suggest using (1) if the GPU can infer a batch of images at the same time. Our Mask R-CNN version uses (1) because we found we need 0.3s-0.4s in case (1) and 0.2s in case (2). Our SegNet version is evaluated using the strategy (2) because SegNet is very fast and can be segmented sequentially.

E. SEMANTIC DELAY EVALUATION

We have analyzed the semantic delay by assuming the keyframe is selected every two frames (see Fig. 6). In experiment, we follow the keyframe selection policy used in ORB SLAM3 and we compared the semantic delay of Mask R-CNN case and SegNet case using the TUM dataset, as shown in Fig. 16. The semantic delay is influenced by these

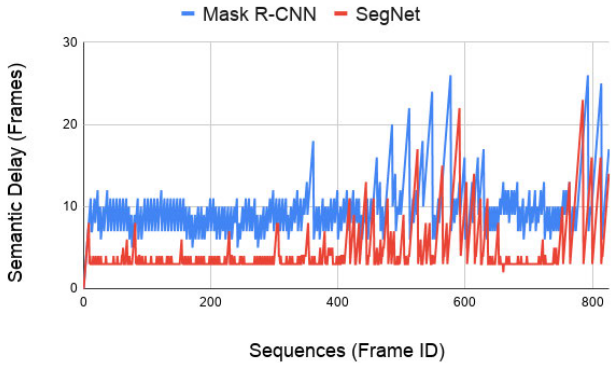


FIGURE 16. Semantic Delay of TUM w/xyz Dataset. The average value of Mask R-CNN case is 10 and SegNet is 5.

factors: 1) the segmentation speed, 2) the keyframe selection policy, 3) the undetermined influence caused by the different running speed of multiple threads (e.g., Loop Closing thread), 3) the hardware configures. In the fr3/w/xyz dataset, the camera sometimes moves very slow and sometimes moves forward or backward. As a result, this will change the keyframe selection frequency and cause the variance of semantic delay.

VIII. CONCLUSION

A novel vSLAM system, semantic-based real-time visual SLAM (RDS-SLAM) for dynamic environment using an RGB-D camera is presented. We modify ORB-SLAM3 and add a semantic tracking thread and a semantic-based optimization thread to remove the influence of dynamic objects using semantic information. These new threads run in parallel with the tracking thread and therefore, the tracking thread is not blocked to wait for semantic information. We proposed a keyframe selection strategy for semantic segmentation to obtain as the latest semantic information as possible that can deal with segmentation methods with different speeds. We update and propagate semantic information using the moving probability which is used to detect and remove outliers from tracking using a data association algorithm. We evaluated the tracking performance and the processing time using the TUM dataset. The comparison against state-of-the-art vSLAMs shows that our method achieved good tracking performance and can track each frame in real-time. The fastest speed of the system is about 30HZ, which is similar to the tracking speed of ORB-SLAM3. In future work, we will try to 1) deploy our system on a real robot, 2) extend our system to the stereo camera and mono camera systems, and 3) build a semantic map.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7747236/

[2] 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, doi: 10.1186/s41074-017-0027-2.

[3] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular SLAM in dynamic environments," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2013, pp. 209–218. [Online]. Available: http://ieeexplore.ieee.org/

[4] W. Dai, Y. Zhang, P. Li, and Z. Fang, "RGB-D SLAM in dynamic environments using points correlations," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2263–2270, Nov. 2018. [Online]. Available: https://arxiv.org/pdf/1811.03217v1.pdf

[5] 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.

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

[7] 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 3*, vol. 345. New York, NY, USA: Springer-Verlag, 2015, pp. 11–26. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-16841-8_2

[8] 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.

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

[10] 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.

[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] 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.

[13] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robot. Auto. Syst.*, vol. 117, pp. 1–16, Jul. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889018308029

[14] 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.

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

[16] 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.

[17] 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.

[18] C. Campos, R. Elvira, J. J. Gómez Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," 2020, *arXiv:2007.11898*. [Online]. Available: http://arxiv.org/abs/2007.11898

[19] 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.

[20] 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.

[21] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—A modern synthesis," in *Proc. Int. Workshop Vis. Algorithms*, 2000, pp. 298–372. [Online]. Available: http://link.springer.com/10.1007/3-540-44480-7_21

[22] 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.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.

[24] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," 2020, *arXiv:2010.07820*. [Online]. Available: http://arxiv.org/abs/2010.07820

[25] 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. [Online]. Available: http://vision.in.tum.de/data/datasets/

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision—ECCV 2016* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence: Lecture Notes in Bioinformatics), vol. 9905. Cham, Switzerland: Springer, 2016, pp. 21–37. [Online]. Available: https://github.com/weiliu89/caffe/tree/ssd

[27] 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.

[28] 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 2014* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8693. Cham, Switzerland: Springer, 2014, pp. 740–755.

[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010. [Online]. Available: http://link.springer.com/10.1007/s11263-009-0275-4

[30] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2012, p. 2012. [Online]. Available: https://mitpress-mit-edu.proxy.library.uu.nl/books/probabilistic-robotics%0Ahttp://mitpress.mit.edu/books/probabilistic-robotics

YUBAO LIU received the bachelor's degree in computer science from Qufu Normal University, Qufu, 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 the Toyohashi University of Technology, Toyohashi, Japan. In 2015, he joined Intel Research Center, Beijing, and he 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 and 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. Since April 2007, he has been a Professor with the Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi, Japan. From March 1994 to February 1995, he was a Visiting Scientist with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA. He has published over 220 articles in international journal and conferences in the areas of intelligent robotics, mobile service robots, robot vision, and artificial intelligence. He 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.