2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

SaD-SLAM: A Visual SLAM Based on Semantic and Depth Information

Xun Yuan , Song Chen

***Abstract*—Simultaneous Localization and Mapping (SLAM) is considered significant for intelligent mobile robot autonomous pathfinding. Over the past years, many successful SLAM systems have been developed and works satisfactorily in static environments. However, in some dynamic scenes with moving objects, the camera pose estimation error would be unacceptable, or the systems even lose their locations. In this paper, we present SaD-SLAM, a visual SLAM system that, building on ORB-SLAM2, achieves excellent performance in dynamic environments. With the help of semantic and depth information, we find out feature points that belong to movable objects. And we detect whether those feature points are keeping still at the moment. To make the system perform accurately and robustly in dynamic scenes, we use both feature points extracted from static objects and static feature points derived from movable objects to finetune the camera pose estimation. We evaluate our algorithm in TUM RGB-D datasets. The results demonstrate the absolute trajectory accuracy of SaD-SLAM can be improved significantly compared with the original ORB-SLAM2. We also compare our algorithm with DynaSLAM and DS-SLAM, which are designed to fit dynamic scenes.**

I. INTRODUCTION

The Simultaneous Localization and Mapping (SLAM) algorithm is essential for intelligent mobile robots. It can help a robot estimate its position and pose through the data collected by its sensors, such as monocular cameras, stereo cameras, RGB-D cameras and lidar, in an unknown environment, and then incrementally construct a map of surrounding environment based on its position and pose. The map can not only help a robot locate itself but also be essential for autonomous robot navigation, task planning, etc.

In recent years, visual SLAM, where the primary sensor is a camera, has been extensively investigated because of the relatively low price of hardware and plenty of useful information stored in images. Compared with a monocular camera and a stereo camera, an RGB-D camera can provide much more accurate depth information, which could contribute to improving the robustness and accuracy of visual SLAM.

In general, visual SLAM algorithms can be divided into two categories, direct methods [4, 5, 6] and feature-based methods [1, 7, 8]. Direct methods directly minimize the photometric error between two adjacent images to track the camera and can construct a relatively dense map. However, they are susceptible to brightness changes and lack of loop-closing to correct accumulative drift. Compared with feature-based methods, the pose estimation error of these algorithms is often relatively large. Meanwhile, feature-based methods rely on the feature points extracted from images to track the camera. With the help of feature points, these methods can relocate and correct accumulative drift. The framework of the feature-based visual SLAM algorithm is quite mature. The system often consists of front-end for tracking camera between two adjacent images, back-end for optimizing camera poses and map points' location among several frames, loop closure detection, and so forth.

Figure 1. Static feature points extracted from movable objects are shown in red and white points represent feature points extracted from static but movable objects like chairs. Green points and blue points represent feature points extracted from static objects which have appeared several times and only one time respectively.

However, the robustness of visual SLAM systems in dynamic environments is still a challenge. Although the above-mentioned methods perform satisfactorily in static scenes, their performance usually decreases significantly in a dynamic environment. Because these algorithms have a common assumption that the environment is static and rigid, any moving object, such as people, cars, or chairs that can be moved by people, appearing in the input frames can corrupt the quality of the pose estimation and even lead to system failure. Even if the above-mentioned movable objects are keeping still sometimes, they can lead to accuracy reduction of loop closure detection and relocation.

In this paper, we propose a feature-based visual SLAM algorithm, which is capable of working robustly in dynamic environments, based on ORB-SLAM2 [1]. The inputs of our system are RGB images, depth images from RGB-D camera, and semantic masks obtained through Mask_RCNN [9]. We find out dynamic feature points with the help of semantic and depth information to decrease the impact of dynamic objects. And we also detect the static feature points from movable objects and use them to finetune the camera pose estimation in front-end to make the tracking algorithm more robust and accurate. Examples of static feature points from movable objects are shown in Fig. 1. In the left image of Fig. 1, the

*This work was partially supported by the National Key R&D Program of China under grant No. 2019YFB2204800, National Natural Science Foundation of China (NSFC) under grant Nos. 61874102 and 61732020.

Xun Yuan is with University of Science and Technology of China, Hefei, China (e-mail: yx111@mail.ustc.edu.cn).

Song Chen is with University of Science and Technology of China, Hefei, China (corresponding author to provide phone: 05516360 2675; e-mail: songch@ustc.edu.cn).

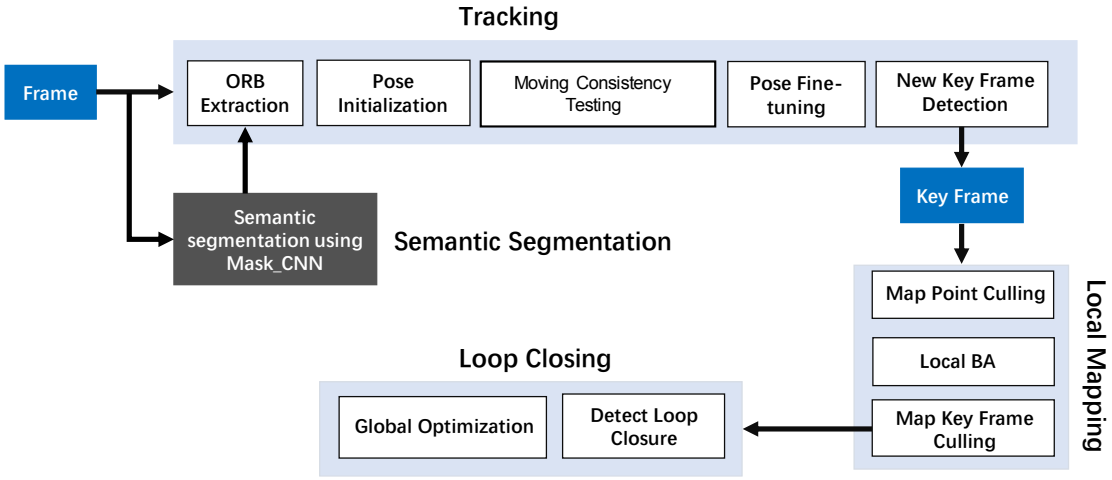

Figure 2. The framework of SaD-SLAM. Loop closing, that searches for loops and perform a graph optimization if a loop is detected, is the same as ORB-SLAM2. Our main contributions are in tracking and local mapping threads. Similar with ORB-SLAM2 three threads that run in parallel: tracking, local mapping, and loop closing. And semantic segmentation is processed offline.

person on the left is standing up, whose upper body is moving while the lower body is still keeping. Our algorithm accurately detects the feature points on his lower body as static points. Meanwhile, the people on the right is sitting still so that our algorithm detects the feature points on his whole body as static points. In the right image of Fig. 1, two people are moving, and our algorithm discards all the feature points extracted from them.

The main contributions of this paper can be summarized in two aspects:

1. A feature-based RGB-D SLAM algorithm based on ORB-SLAM2 is proposed, which can perform satisfactorily in a dynamic environment. Our method could significantly improve the pose estimation accuracy and run much more stably in highly dynamic scenes compared with ORB-SLAM2.

2. We test epipolar constraints between feature points in a current frame and a previous frame and then find out the static feature points from dynamic objects and static but movable objects like chairs to help to improve the accuracy and robustness of camera pose estimation.

The experimental results in section IV demonstrate that the absolute trajectory accuracy of SaD-SLAM can be improved significantly compared with original ORB-SLAM2. And our algorithm performs more robustly than DynaSLAM [2] and DS-SLAM [3].

The remaining of this paper is organized as follows. Section II provides an overview of various accomplishments in respect of visual SLAM in dynamic environments. Section III describes the proposed algorithm in detail. Subsequently, we evaluate our algorithm and compare SaD-SLAM against related works on the TUM RGB-D dataset [14] in section IV. Finally, a brief conclusion is given in section V.

II. Related Works

As for feature-based visual SLAM algorithm, because of the assumption of static and rigid environment, moving objects may contribute plenty of unreliable feature points to the system, which often results in the increasing of pose estimation error. ORB-SLAM2 [1] tries to alleviate the influence caused by unreliable feature points by using robust cost functions, but the estimated trajectory error in dynamic environments is still unacceptable. Therefore, it's essential for feature-based visual SLAM to discard unreliable feature points.

Some relevant works try to address this issue. Tan et al. [10] reliably detect changed feature points by projecting them from the keyframes to current frame for appearance and structure comparison. And the appearance change due to occlusions also can be reliably detected and handled. Kundu et al. [11] use multi-view geometric and epipolar constraints to classify a pixel as moving or static. A Bayesian framework is used to assign a probability that the pixel is stationary or dynamic based on the above geometric properties, and the probabilities are updated when the pixels are tracked in subsequent images. Lin and Wang [12] detect moving objects by examining monocular SLAM results under different hypotheses. Two local monocular SLAMs are initialized under two hypotheses when a new feature is extracted. One is local monocular SLAM without adding this new feature, and the other is local monocular SLAM under the assumption that this new feature is stationary. The differences between the two results are temporally integrated using binary Bayes filter. After a fixed number of updates, this new feature can be classified as static if the log odds ratio is larger than a predetermined threshold.

The methods mentioned above mainly rely on geometric features without priori semantic information of the environment. In recent years, with the development of machine learning, semantic segmentation through convolutional neural networks (CNNs) has achieved significant improvement. Some advanced CNN frameworks, such as SegNet [13] and Mask-RCNN [9], can perform semantic segmentation in pixel-level quite accurately. Semantic information obtained through CNN can provide SLAM systems with a prior knowledge about which object is dynamic and which object could be moved. And feature points

extracted from dynamic objects have a high possibility to be moving and unreliable.

Some relevant works fit visual SLAM systems into dynamic environments based on the semantic information obtained through CNN. Berta et al. [2] detect the moving objects by multi-view geometry and deep learning. Their work performs pretty well on TUM RGB-D dataset. But although some feature points extracted from dynamic objects are keeping static, they still discard those feature points, which could result in missing many reliable feature points. Chao et al. [3] check moving consistency of feature points by epipolar constraint. If there are above a certain number of dynamic points produced by moving consistency check fall in the contours of a segmented object, then this object is determined to be moving, and all the feature points located in the object's contour will be removed. But they have an assumption that all objects are rigid and for example, people are non-rigid objects. Also, it's not easy for a system to distinguish moving points from random error and pose estimation error between two adjacent frames. In this paper, we propose a reliable method to detect static feature points from movable objects.

III. System Description

In this section, SaD-SLAM will be introduced in detail. First of all, the framework of SaD-SLAM is presented. Secondly, we give a method to obtain semantic masks of the input frames. Subsequently, we explain our three-stage algorithm, including pose initialization, moving consistency testing, and pose fine-tuning, which detects whether a feature point is moving or not with the help of semantic masks and depth information. Our system can exploit more reliable feature points to optimize camera pose estimation in a dynamic environment. Finally, we modify ORB-SLAM2's local mapping and local bundle adjustment to fit our algorithm.

All the stages are described in-depth in the next subsections (III-A to III-F).

A. Framework of SaD-SLAM

ORB-SLAM2 is one of the most successful feature-based visual SLAM systems which performs satisfactorily in most public datasets. Therefore, we adopt ORB-SLAM2 as the backbone of SaD-SLAM. The framework of SaD-SLAM is shown in Fig. 2.

Like ORB-SLAM2, three threads run in parallel in SaD-SLAM: tracking, local mapping and local closing. The task of detecting whether the feature points are moving and fine-tuning pose estimation is done in the tracking thread. And this task can be divided into three phases: pose initialization, moving consistency testing and pose fine-tuning. The semantic masks are gained through Mask R-CNN offline.

B. Semantic Segmentation

For detecting dynamic and static objects, we adopt Mask R-CNN, which is the state-of-the-art for object instance segmentation and could produce both pixel-wise semantic segmentation of images and the instance labels. We use the PyTorch implementation by facebookreaserch. Our Mask R-CNN uses X-101-FPN as the backbone. And we directly utilize the parameters that are pretrained on imagenet dataset [17] without any fine-tuning.

The input of Mask R-CNN is an original RGB image. And the output of the network, assuming that the input is a RGB image of size m×n×3, is a matrix of size m×n×l, where l is the number of objects in the image. For each output channel $i \in l$, a binary mask of an object is obtained. We mainly pick out the dynamic objects and the relatively big static but movable objects. As for TUM RGB-D dataset, they are people and chairs. Feature points belong to people are labeled as dynamic points, and feature points belong to chairs are labeled as static but movable points. Other feature points are labeled as static points. By combining all the channels into one, we can obtain the segmentation of people and chairs in one image of the scene. When one pixel belongs to both people and chair in different output channels, we judge the pixel belonging to people.

C. Pose Initialization

After semantic segmentation, we obtain a semantic mask of movable objects like people and chairs. But some small items like pencils and books which could be moved by the people around have not been segmented out because it's hard for a CNN to segment small things out and the accuracy is far from acceptable. What's more, we could find them out in an easy and relatively accurate way.

Taking TUM RGB-D dataset for example, whenever we extract a feature point from input frame which does not belong to people and chair, we check the points 5 pixels away and 10 pixels away around the feature point in the semantic mask. If one of the points belongs to people or chair and its depth is similar to the feature point's, we could judge the feature point as dynamic or static but movable. As described above, the dynamic has a higher priority than static but movable. In this way, we can detect the item held in the human's hand easily without the help of CNN. What's more, semantic segmentation through CNN is prone to errors at the boundary of an object, and the above-method can make up for these errors. As a result, we need CNN to segment fewer classes which may help to make the CNN smaller, faster and more accurate.

At the current stage, we minimize the reprojection error to optimize the camera pose estimation with static and static but movable feature points because static but movable feature points most likely keep still or move slowly. And in this way, we get initialized camera pose which will be fine-tuned.

D. Moving Consistency Testing

To test the moving consistency of the feature points and map points, we add "movable" characteristic which consists of static point, dynamic point, static but movable point and static point changed from dynamic or static but movable points four classes, to map points and feature points of frames and keyframes. The characteristic of feature points extracted from static objects, dynamic objects and static but movable objects is initialized as static points, dynamic points and static but movable points. The definitions of the frame, keyframe and map point here are the same as ORB-SLAM2. Please refer to ORB-SLAM2 [1] for more details.

As a result of that TUM RGB-D dataset has 30 frames per second, it is hard for us to distinguish moving points from random errors and pose estimation error in two consecutive frames. To resolve this problem, we record the feature point

ID of the last frame, which corresponds to the map point of the current frame, so that we can find the current frame map point's corresponding feature point in any previous frame.

In this paper, we track possible moving feature points for five consecutive frames. And if the feature point in the current frame and its corresponding feature point in the fourth frame before the current frame meet the epipolar constraint, it and its corresponding feature points in the last four frames will be considered as static feature points. And you may change the number of tracking frames according to your condition like camera frequency.

Let n and n-4 denote the id of the current frame and the id of the fourth frame before the current frame. Let p_n, p_{n-4}, denote the matched points in the current frame and frame n-4 respectively, and x_n, x_{n-4}, are their homogeneous coordinate form:

$$p_n = [u_n, v_n], p_{n-4} = [u_{n-4}, v_{n-4}] \tag{1}$$

$$x_n = [u_n, v_n, 1], x_{n-4} = [u_{n-4}, v_{n-4}, 1] \tag{2}$$

And the transform matrix from frame n-4 to the current frame can be computed by the following equation:

$$T_{n(n-4)} = T_{nw} * T_{w(n-4)} = \begin{bmatrix} R_{n(n-4)} & t_{n(n-4)} \\ 0^T & 1 \end{bmatrix} \tag{3}$$

where T_{nw} is the initialized transform matrix from the world coordinate to current frame coordinate, and $T_{w(n-4)}$ is transform matrix from frame n-4 coordinate to world coordinate. Then we compute fundamental matrix F and epipolar line L as follows:

$$F = K^{-T} t_{n(n-4)} \wedge R_{n(n-4)} K^{-1} \tag{4}$$

$$L = \begin{bmatrix} A \\ B \\ C \end{bmatrix} = F x_{n-4} \tag{5}$$

where K is the intrinsic parameter matrix of the camera.

And in the end, the distance from the matched point of the current frame to its corresponding epipolar line is determined as follow:

$$D = \frac{x_n F x_{n-4}}{\sqrt{A^2 + B^2}} \tag{6}$$

If D is less than a set specific value, we will change the feature point's characteristic and its corresponding map point's characteristic to static point from dynamic or static but movable objects. Otherwise, the feature point's characteristic will be converted to moving point and the corresponding map point will be deleted just in the current frame so that it doesn't need to be tracked in the next frame. If the current frame is judged as a keyframe, a new map point will be created for the feature point.

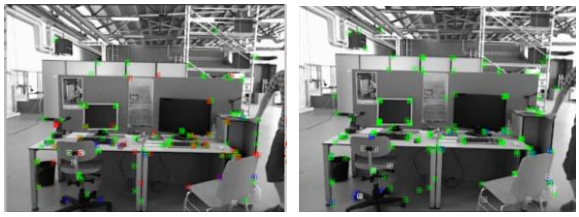

Figure 3. Red points represent the feature points that do not meet the moving consistency testing between two adjacent frames. The threshold is set 1 and 8 in left picture and right picture respectively.

We use a similar testing strategy for static point. If a feature point extracted from static objects could be tracked in five consecutive frames but violate the epipolar constraint, the feature point's characteristic will be converted to moving point, and the corresponding map point will be deleted just in the current frame as above, in case the semantic segmentation has some errors or mismatching errors. Also, if the current frame is judged as a keyframe, a new map point will be created for the feature point.

E. Pose Fine-tuning

At the beginning of this phase, we have the initialized camera pose, static feature points, dynamic feature points, feature points extracted from static but movable objects and static points changed from movable feature points. Subsequently, we have to do a two-stage pose fine-tuning.

On the first stage, we minimize the reprojection error to fine-tune the camera pose using static feature points, feature points from static but movable objects and static points changed from movable feature points.

On the second stage, we extract more feature points in the current frame by projecting local map points to the current frame and set their "movable" characteristic the same as the corresponding local map point's. We use the map points whose characteristic are static point because other kinds of map points could move from its original location. And then we use the new feature points and feature points used in the first stage together to fine-tune the camera pose.

F. Local Mapping and Local Bundle Adjustment

We perform local bundle adjustment only with static map points whose characteristics are not dynamic point. Because moving map points could not only increase the computational burden of local bundle adjustment but also reduce the accuracy of local bundle adjustment.

Different from DynaSLAM and DS-SLAM, we still build map points from dynamic objects into a local map for tracking them and testing their moving consistency. They could also be used in other researches, such as real-time dynamic map construction and object tracking. And different from ORB-SLAM2, we only fuse static map points in local mapping thread.

IV. EXPERIMENTAL RESULTS

In this section, experimental results will be presented to demonstrate the robustness and accuracy of SaD-SLAM. The TUM RGB-D dataset [14] is composed of 39 sequences recorded with a Microsoft Kinect sensor in different indoor scenes at full frame rate (30Hz). Both the RGB and the depth images are available, together with the ground-truth trajectory. In the sequences named sitting, two people are sitting in front of a desk while speaking and gesturing, i.e., there is a low degree of motion. In the sequences named walking, two

TABLE I. COMPARISON OF THE MEDIAN RMSE OF ATE (M) OF SAD-SLAM AGAINST ORB-SLAM2 AND DYNASLAM FOR TUM RGB-D DATASET

Sequences	ORB-SLAM2	DynaSLAM			SaD-SLAM		
	median	Median	min	max	median	min	max
fr3_walking_xyz	*	**0.015**	0.014	0.016	0.0167	0.0154	0.0180
fr3_walking_static	0.4038	**0.006**	0.006	0.008	0.0166	0.0161	0.0168
fr3_walking_rpy	0.7511	0.035	0.032	0.038	**0.0318**	0.0275	0.0384
fr3_walking_half	*	**0.025**	0.024	0.031	0.0257	0.0234	0.0335
fr3_sitting_xyz	**0.0091**	0.015	0.013	0.015	0.0124	0.0118	0.0130
fr3_sitting_static	0.0081	X	X	X	**0.0060**	0.0058	0.0062
fr3_sitting_rpy	**0.0205**^	X	X	X	0.0288	0.0270	0.0297
fr3_sitting_half	0.0392^	0.017	0.016	0.020	**0.0151**	0.0136	0.0173

A "X" indicates that the original paper does not provide the result. A "*" indicates the system would be break down while running the corresponding sequence. A "^" after the number indicates that the system would fail to track some frames of the sequence but success to relocate. All ORB-SLAM2 experiments failed to track some frames for fr3_sitting_rpy sequence in our five experiments. Two ORB-SLAM2 experiments failed to track some frames for fr3_sitting_half sequence in our five experiments.

people walk both in the background and the foreground most of the time, and they sometimes sit down in front of the desk. These sequences are challenging for original ORB-SLAM2. For example, the moving people in the scenes could corrupt the robustness and accuracy of ORB-SLAM2 or even break down the system. We test SaD-SLAM with sitting sequences and walking sequences to evaluate its ability to detect static feature points from movable objects and find out moving feature points. All the experiments are performed on a computer using Ubuntu 18.04 operating system with Intel i7-4790 CPU and 16GB memory.

First of all, we evaluate our strategy of tracking feature points for five consecutive frames and then testing their moving consistency. As shown in Fig. 3, if we set the threshold too little, the algorithm may incorrectly identify static feature points extracted from static objects as moving feature points because of the pose estimation error. And those static feature points may be crucial to correct the error. The situation is the same for potentially moving feature points. Therefore, if we set the threshold too small, the algorithm couldn't detect static feature correctly and robustly. Otherwise, if we set the threshold relatively large, the system is prone to wrongly identify the moving feature points as static because the motion of feature points could be small between two adjacent frames. And if we track feature points for some consecutive frames, the movement will become large enough for the system to distinguish the moving features points from pose estimation error. According to our experiments, if we

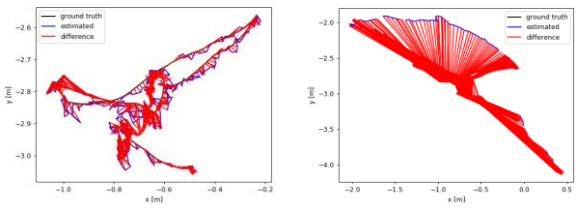

Figure 4. Left picture shows Ground truth, trajectories estimated by SaD-SLAM and absolutely trajectory error (ATE) in the TUM sequence fr3_walking_rpy. Right picture shows Ground truth, trajectories estimated and absolutely trajectory error by ORB-SLAM2 in the TUM sequence fr3_walking_rpy.

TABLE II. COMPARISON OF THE RMSE OF ATE (M) OF SAD-SLAM AGAINST SAD-SLAM-- FOR TUM RGB-D DATASET

Sequences	SaD-SLAM--	SaD-SLAM
fr3_walking_xyz	0.0169	**0.0167**
fr3_walking_static	0.0173	**0.0166**
fr3_walking_rpy	0.0398	**0.0318**
fr3_walking_half	0.0267	**0.0257**
fr3_sitting_xyz	0.0125	**0.0124**
fr3_sitting_static	0.0065	**0.0060**
fr3_sitting_rpy	0.0305	**0.0288**
fr3_sitting_half	0.0189	**0.0151**

TABLE III. COMPARISON OF THE RMSE OF ATE (M) OF SAD-SLAM AGAINST DS-SLAM FOR TUM RGB-D DATASET

Sequences	DS-SLAM	SaD-SLAM
fr3_walking_xyz	0.0247	**0.0167**
fr3_walking_static	**0.0081**	0.0166
fr3_walking_rpy	0.4442	**0.0318**
fr3_walking_half	0.0303	**0.0257**
fr3_sitting_xyz	X	**0.0124**
fr3_sitting_static	0.0064	**0.0060**
fr3_sitting_rpy	X	**0.0288**
fr3_sitting_half	X	**0.0151**

A "X" indicates that the original paper does not provide the result.

track two consecutive frames, the pose estimation error would drop dramatically, and tracking five consecutive frames is a proper method.

Secondly, we show the effectiveness of using static feature points from movable objects to optimize pose estimation. We only remove the moving consistency testing part from our algorithm, and other sections are unchanged. That is, we use feature points extracted from static objects to fine-tune pose estimation. We name our contrast algorithm as SaD-SLAM--. We compare the performance of SaD-SLAM-- and SaD-SLAM using the absolute trajectory RMSE as the error metric, as proposed by Sturm et al. [14]. Same as Mur and Tardos [1], to account for the non-deterministic nature of the multi-threading system, we run each sequence five times and show median results for the accuracy of the estimated trajectory. And the contrast results for eight sequences within the TUM dataset are shown in Table II. The absolute trajectory accuracy of SaD-SLAM is higher than SaD-SLAM-- in all of the eight sequences. And if some frames of a sequence can't provide enough reliable feature points from static objects like the fr3_walking_rpy sequence, the improvement of absolute trajectory accuracy would be more significant with the help of static feature points from movable objects.

Subsequently, we compare our algorithm with the original ORB-SLAM2 and DynaSLAM. The results are shown in Table I. Compared to original ORB-SLAM2, the accuracy of the estimated trajectory of SaD-SLAM improves dramatically in the walking sequences. Because ORB-SLAM fails to track some frames in the sequences fr3_sitting_rpy and fr3_sitting_half, our algorithm is more robust than ORB-SLAM2, even when people are sitting in the scene. Fig. 4 shows an example of the estimated trajectories error of SaD-SLAM and ORB-SLAM2, compared to the ground-truth. Moreover, SaD-SLAM performs better than Dynaslam in the sitting sequences, because we extract more reliable feature points than DynaSLAM, which discards all the feature points on dynamic objects. Because static objects couldn't provide enough reliable feature points in some frames of the sequence fr3_walking_rpy, our algorithm performs better than Dynaslam in this walking sequence.

Lastly, we compare our performance with DS-SLAM that is capable of detecting whether the feature points extracted from movable objects are moving or keeping still as our method. The results are shown in Table III. Our method works more accurately in most of the sequences.

V. CONCLUSION

In this paper, we have presented a visual SLAM system that is built on ORB-SLAM2 and performs robustly and accurately in dynamic environments through discarding the moving feature points with the help of semantic information obtained by Mask_RCNN and depth information provided by RGB-D camera. Our method tries to exploits more reliable feature points for camera pose estimation by finding out the static feature points extracted from movable objects, which would benefit a lot when static objects couldn't provide enough feature points in the scene. We show the effectiveness of our algorithm by comparing SaD-SLAM with original ORB-SLAM2, sad-SLAM-- that is a contrast of SaD-SLAM, DynaSLAM, and DS-SLAM on the challenging dynamic sequences of the TUM RGB-D dataset. The results indicate our methods could work more accurately and robustly in dynamic environments.

With the development of machine learning, Ono et al. [15] and DeTone et al. [16] use CNN to extracted feature points and compute their descriptors, which could obtain more reliable feature points and track them more correctly and efficiently against the traditional methods. Our future work might exploit CNN to extract feature points and track them in the front-end of our SLAM system.

REFERENCES

[1] Mur-Artal, Raul, and Juan D. Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." *IEEE Transactions on Robotics* 33.5 (2017): 1255-1262.

[2] Bescos, Berta, et al. "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes." *IEEE Robotics and Automation Letters* 3.4 (2018): 4076-4083.

[3] Yu, Chao, et al. "DS-SLAM: A semantic visual SLAM towards dynamic environments." *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018.

[4] Engel, Jakob, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM." *European conference on computer vision*. Springer, Cham, 2014.

[5] Forster, Christian, Matia Pizzoli, and Davide Scaramuzza. "SVO: Fast semi-direct monocular visual odometry." *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014.

[6] Engel, Jakob, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry." *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017): 611-625.

[7] Davison, Andrew J., et al. "MonoSLAM: Real-time single camera SLAM." *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007): 1052-1067.

[8] Klein, Georg, and David Murray. "Parallel tracking and mapping for small AR workspaces." *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007.

[9] He, Kaiming, et al. "Mask r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2017.

[10] Tan, Wei, et al. "Robust monocular SLAM in dynamic environments." *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2013.

[11] Kundu, Abhijit, K. Madhava Krishna, and Jayanthi Sivaswamy. "Moving object detection by multi-view geometric techniques from a single camera mounted robot." *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009.

[12] Lin, Kuen-Han, and Chieh-Chih Wang. "Stereo-based simultaneous localization, mapping and moving object tracking." *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010.

[13] Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017): 2481-2495.

[14] Sturm, Jürgen, et al. "A benchmark for the evaluation of RGB-D SLAM systems." *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012.

[15] Ono, Yuki, et al. "LF-Net: learning local features from images." *Advances in neural information processing systems*. 2018.

[16] DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018.

[17] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009.