2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

A Unified Framework for Mutual Improvement of SLAM and Semantic Segmentation

Kai Wang[1] Yimin Lin[1] Luowei Wang[1] Liming Han[1] Minjie Hua[1] Xiang Wang[1] Shiguo Lian[1] Bill Huang[1]

Abstract— **This paper presents a novel framework for simultaneously implementing localization and segmentation, which are two of the most important vision-based tasks for robotics. While the goals and techniques used for them were considered to be different previously, we show that by making use of the intermediate results of the two modules, their performance can be enhanced at the same time. Our framework is able to handle both the instantaneous motion and long-term changes of instances in localization with the help of the segmentation result, which also benefits from the refined 3D pose information. We conduct experiments on various datasets, and prove that our framework works effectively on improving the precision and robustness of the two tasks and outperforms existing localization and segmentation algorithms.**

I. INTRODUCTION

Localization and Segmentation are two of the most fundamental tasks for robotic movement and sensing. The former computes the robot's current position and orientation, and the latter helps to perceive the distribution and precise boundaries of the objects of interest within the robot's field of view. These two techniques are essential in many robotic applications including autonomous driving, Unmanned Aerial Vehicles (UAV), robot patrolling and logistics, etc.

For the localization task, visual Simultaneous Localization and Mapping (vSLAM) is one of the most promising methods due to its relatively low hardware and computational cost characteristics in recent years. It utilizes image sequences with some auxiliary sensor data such as depth map, Inertial Measurement Unit (IMU) data, etc., to create the map of the environment and return the current location information at the same time. A big challenge in vSLAM is that the environment in which the robot locates is usually changeable. On one hand, instantaneous movement of some objects during mapping will affect the precision of the map due to the inconsistency of the moving trend in the scene [1]. On the other hand, the map created will no longer be consistent with the environment once some objects have moved after mapping completes. As a result, subsequent localization based on this map will not be accurate.

For the segmentation task, 2D image-based semantic segmentation using deep neural network has proved to be effective in most cases and has been widely used in many systems [2]. It is able to output the exact boundaries of a series of segmented regions and their classes. Anyway, unprecise manual labeling and lack of similar training data usually lead to inaccurate segmentation results for these deep learning methods.

Previously, these two tasks were generally regarded as two independent tasks whose results were rarely utilized by each other. In this paper, we propose a novel framework for simultaneously improving the vSLAM as well as semantic segmentation precisions. The segmentation and vSLAM are performed in an interweaved method and the results are used to refine each other's. Specifically, the computed pose information of the previous and current frames are utilized to refine the segmentation of the latter one, in which all the potentially moveable objects are then identified and sent to the vSLAM module for further computation of the tracking and mapping of the corresponding frame. This scheme repeats through the whole process and both the vSLAM and segmentation precisions of this sequence are therefore enhanced. Furthermore, the map created becomes more robust to changes of the scene and the localization in the same environment afterwards will benefit from it and become more precise. This framework is tested on different datasets and proves to be more effective over existing works on both the vSLAM and segmentation tasks.

The contributions of this paper include:

- A unified framework of enhancing the vSLAM and segmentation tasks mutually.
- A novel approach for enhancing both the mapping and localization precisions in vSLAM by identifying and processing both the moving and potentially moveable objects respectively.
- An effective refinement scheme for image segmentation by making use of 3D pose information.

The rest of the paper is organized as follows: Section II reviews the related works on vSLAM and segmentation. Section III introduces the proposed framework and workflow in detail, and experimental results are shown and discussed in Section IV. Section V gives the conclusion.

II. RELATED WORK

A. vSLAM for Dynamic Scenes

vSLAM is used to estimate the camera location and 3D map of the scene through a set of feature correspondences extracted from a series of images [3]. Various works on vSLAM have been proposed in recent years, from the seminal work PTAM [4] to the popular ORB-SLAM2 [5]. Most of these approaches assume that the observed scenes are relatively static, and pose estimation might drift or even be lost as there are not features to be matched consistently in the case of scenes with dynamic objects.

[1] All the authors are with CloudMinds Technologies Inc., Beijing 100102, China. kai.wang,anson.lin,luowei.wang, liming.han,michael.hua,xiang.wang,scott.lian, bill@cloudminds.com

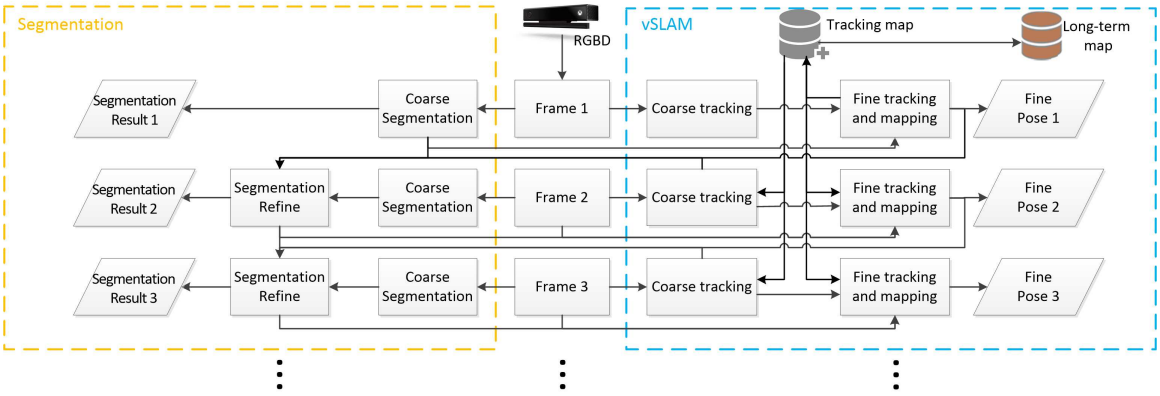


Fig. 1. The overall workflow of the proposed framework, which contains a segmentation module and a vSLAM module. For each input frame, a coarse pose and segmentation are first calculated. The two results are then used to estimate a fine pose and update a tracking map. A long-term map is also maintained for the further visit of the same area. At the same time, the segmentation results can also be refined by using that of the previous frame and the poses estimated in the two frames. The refinement of the vSLAM and segmentation results is implemented within a single iteration for each frame.

There have been works proposed to handle dynamic environments [6]. For example, [7] computed the likelihood of a moving object based on a motion metric computed from optical flow and then segment the moving objects. [8] further extended it to handle stereo image sequences. Recently, researchers have shifted their focus to using deep neural network to do the segmentation to remove outliers for accurate pose estimation. For example, Mask-SLAM [9] excludes feature points detected in the sky area or on cars using the segmentation mask trained by DeepLab v2 [10]. The work [1] proposed to combine multi-view geometry models and deep-learning-based algorithms for detecting dynamic objects and removed them from the frames. In [11], the depth map, sparse scene flow and semantic cues are combined to classify scene as either static background, movable and moving objects. While these methods have proved that excluding feature points in certain masked area makes the estimation of camera motion more stable, they rely heavily on the exact segmentation of the moveable objects and are prone to be inaccurate when its precision is limited. The idea in [12] which identified dynamic objects to enhance the vSLAM precision and further provided a refined dataset for training the object detection network is similar to our work, except that the extraction of objects with object detection in the first step is less accurate, and the second step remains an offline scheme.

B. Image and Video Segmentation

The pioneering work [13] on deep neural network based image segmentation explored the use of Convolutional Neural Network (CNN) to segment the images, through adapting classifiers for dense prediction by replacing the last fully-connected layer with deconvolution layers. Later on, [14] made use of the encoder-decoder architecture and reused the pooling indices from the encoder to decrease parameters. DeepLabv3 [15] augments the Atrous Spatial Pyramid Pooling (ASPP) module in [10] with image-level feature to capture longer range information as in [16], and DeepLab-v3+ [17] further extends it to include an effective decoder module to refine the segmentation results along object boundaries. Pyramid Scene Parsing Network (PSPNet) [18] implements spatial pooling at several grid scales and demonstrates satisfactory performance.

Furthermore, algorithms have been proposed to achieve instance-level segmentation. The prior work [19] task uses R-CNN [20] to classify region proposals, which are then refined by category-specific coarse mask predictions. MNC [21] proposed a cascaded structure, which consists of three networks used for differentiating instances, estimating masks, and categorizing objects respectively. FCIS [22] performs object segmentation and detection sub-tasks jointly and exploits the strong correlation between the two sub-tasks with shared score maps. Mask R-CNN [23] extends Faster R-CNN [24] by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

There are also some works proposed for video sequence-based segmentation. For example, [25] made use of the spatial-temporal information of consecutive frames by introducing 3D-Conv [26] and Conv-LSTM [27] modules, so as to enhance the precision of video segmentation. Since the 3D spatial information of adjacent frames was not utilized, they may still fail to predict precise boundary information.

III. Framework

A. Overall Workflow

The general workflow of the proposed framework is shown in Fig. 1. This framework takes the RGB image sequences as well as the depth map sequences as input. It includes two major modules: the vSLAM module and the segmentation module. For each input frame, the vSLAM module will output the pose information of the camera w.r.t. the world and update the map of the environment for long-term use, and the segmentation module will produce an image segmentation result with the semantic information of each pixel.

Specifically, the initial input frame will be first segmented, and potentially dynamic objects are identified. At the same time, a coarse pose is computed in the vSLAM module. The

results will then be sent to the vSLAM module to build the map. Next, when a new frame comes, a coarse vSLAM and segmentation will be performed first, and the coarse pose together with the pose and segmentation result of the last frame will be sent to the segmentation module to refine the coarse result. After the final segmentation result of this frame is computed, it will be sent to the vSLAM module to proceed fine tracking and mapping, after which the precise map and location information will be obtained.

Next, the detailed vSLAM and segmentation modules will be introduced.

B. Initial Segmentation

For each input RGB frame, we used the FCIS [22] algorithm which proved to be effective on various datasets to perform an initial segmentation. We trained the network on MS COCO [28] dataset which contains 80 classes for both indoor and outdoor objects. For an input RGB image, FCIS is able to compute the bounding box for each object. If the pixel value in the bounding box is larger than a threshold, it is regarded as part of the object; otherwise, it will be marked as the background. We repeat this operation for all the bounding boxes to get the mask for the whole image.

After the segmentation, we identified the moveable objects from all the instances in the result, according to a predefined shortlist in which only objects that are likely to move or be moved (such as person, cars, cup, chair, etc.) among all the 80 classes are selected. The result is in the form of a mask image with the region and instance ID of each segmented instance encoded, and will be sent to the vSLAM module to proceed the tracking and mapping computation.

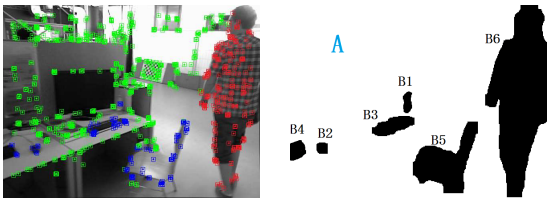
C. vSLAM based on Segmentation Result

We use the ORB-SLAM2 algorithm [5] which has shown satisfactory performance in many scenarios. To ensure the stability, we used the RGB-D version of ORB-SLAM2 which takes both RGB image and depth map as input.

Each time a new frame comes, we first implement a coarse tracking to get an initial guess of the pose of the current frame. Specifically, we first extract the ORB feature points and align them with the depth map to get the 3D coordinates (P_x, P_y, P_z) of each point P, and get the coarse rotation R_c and translation T_c by minimizing the reprojection error as what the original ORB-SLAM2 did.

The extracted feature points are then classified into a background set A and other different sets $\{B_i | i = 1...n\}$ according to their positions in different segmented areas. If a point P lies in the background area, it belongs to set A; otherwise it falls into set B_i which corresponds to the area of segmented instance i. The motion states of the classified point sets will then be judged according to the coarse rotation R_c and translation T_c. Specifically, we project the points in the tracking map onto the current frame, and for each point P_i in the frame, a best matching point P_{match} is found. If the Euclidean distance between P_i and P_{match} is less than a predefined threshold, then P_i is regarded as static. For the set B_i that P_i belongs to, if the percentage of moving points is less than a threshold, then the instance that set B_i corresponds to is regarded as a static object in the current frame, otherwise, it is deemed moving.

An example of the segmented regions and classified features points is shown in Fig. 2.



(a) (b)

Fig. 2. Illustration of feature points and segmented area classifications. (a) The detected feature points are classified into background (in green), moving (in red) and moveable (in blue) points; (b) The segmentation result with regions classified into background (A) and moving or moveable (B1-B6) .

Next, 2D-3D matching between the points in the background set A and the sets $\{B_s\}$ that are considered as static and also in the tracking map is implemented by minimizing the reprojection error, and fine rotation R_f and translation T_f can thus be obtained. After the fine pose has been obtained, it will be sent to the segmentation module for the refinement of the initial segmentation result.

There are two types of maps created and maintained in the vSLAM module: tracking map and long-term map.

The tracking map is used to compute the trajectory of the camera during the tracking process. The new point P_m in the tracking map is computed by projecting each point P_c the background point set A and moving point set B_s of the new key frame onto the tracking map through $P_m = R_f P_c + T_f$. If there are already matching points, then no more update of the map is required; otherwise, the newly projected 3D points will be added into the tracking map. The use of only points of the static objects will help the preservation of the information used for computing the camera pose in the current scene, and thus improves the tracking stability and trajectory precision.

The long-term map is designed for long-term use. It only needs to be created at the first time when a robot navigates in a new area, and can be reused later on to avoid duplicated mapping computation when the same region is visited . Therefore, only the points whose positions will probably remain fixed over time should be included in this map to provide stable environment information. To do that, each time the tracking map is updated, we remove the points that belong to set B_s in the tracking map and have the potential to move in future, and add the rest points (i.e. the points in set A) into the long-term map.

D. Refinement of Segmentation Result

After we get the coarse pose R_c, T_c of the current frame, and the fine pose R_f, T_f as well as the segmentation result of the previous frame, we can use them to update the segmentation result in the current frame.

First, we project each 2D point (p_u, p_v) of the segmented regions in the last frame which has been refined and assumed

to be accurate to (p'_u, p'_v) in the current frame according to the following equations:

$$P_z = D(p_u, p_v)/DF, \quad (1)$$

$$P_x = (p_u - c_x) * P_z/f_x, \quad (2)$$

$$P_y = (p_v - c_y) * P_z/f_y, \quad (3)$$

$$\begin{bmatrix} p'_u \\ p'_v \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [R|T] \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} /s. \quad (4)$$

In the above equations, f_x, f_y and (c_x, c_y) are the focal lengths and principal point of the camera respectively. $D(p_u, p_v)$ is the depth value of (p_u, p_v) and DF is the depth factor of the depth map. $R = R_c^{-1} R_f$ and $T = T_f - T_x$ represent the relative rotation and translation w.r.t. the last frame. s is the scale factor of the image.

Next, we try to refine the initially segmented image with each projected region Re_p. The workflow is listed in Algorithm 1. We first try to find a matching region for Re_p in the current frame, by measuring the similarity S_{cp} between each region Re_c in the roughly segmentation result and Re_p using:

$$S_{cp} = w1 * Dist(Re_c, Re_p) + w2 * \sqrt{\frac{Area((Re_c - Re_p) \cup (Re_p - Re_c))}{Area(Re_c) + Area(Re_p)}}, \quad (5)$$

where $Dist(Re_c, Re_p)$ refers to the Euclidean distance between the barycenters of Re_c and Re_p. The function $Area(\cdot)$ computes the total number of pixels inside a region. The first item measures the positional distances of the two regions and the second one is used to compute their shape difference. $w1$ and $w2$ are the weights for the two items respectively. The region Re_c with the smallest S_{cp} value which is smaller than a predefined threshold will be selected as the matching region for Re_p. We then compare the ratios of their intersection area to the two regions, and the region with larger ratio is considered as a reliable one and preserved as the finally segmented region.

If no matching region is found for Re_p, there is a high possibility that the segmentation algorithm failed to recognize an instance that was supposed to be segmented when the number of segmented instances in the current frame is less than that of the previous one. In that case, we will update the segmentation result by adding Re_p to it. If the numbers of segmented instances are same, then we simply skip the current Re_p and repeat the same process for the next Re_p.

It should be mentioned that this strategy is based on the assumption that there is no drastic changes between two adjacent frames. In some extreme circumstances, for example, if the frequency of the camera is not high enough to ensure the fast-moving objects be well captured, the algorithm may fail on judging the region correspondence and lead to fake results. This may be alleviated by introducing frame interpolation into the computation, although this case is rarely seen in real applications.

Algorithm 1 Workflow for segmented regions' update.

```
for a projected region Re_p do
    find the matching region Re_c for Re_p with (5)
    if found then
        compute In_cp = Re_c ∩ Re_c
        compute Ra_c = In_cp / Re_c
        compute Ra_p = In_cp / Re_p
        if Ra_c < Ra_p then
            replace Ra_c with Ra_p
        else
            //do nothing.
        end if
    else
        if #regions^{t-1} > #regions^t then
            add Ra_p to segmentation result
        else
            //do nothing.
        end if
    end if
end for
```

IV. RESULTS AND DISCUSSIONS

We test our framework on different datasets with ground truths available, and compare with other state-of-the-art works on vSLAM and image segmentation. We run each sequence ten times as in [1] to compensate for the non-deterministic nature of dynamic scenes. All tests were implemented on a workstation with Intel i7 6700K CPU, with 32 *GB* RAM and Nvidia GTX1070 GPU.

A. Test Results on TUM Dataset

We first test the performance of the vSLAM module of our framework on TUM dataset [29] in which 39 RGB-D sequences are collected. Each sequence contains both 640×480 8-bit RGB images and 640×480 16-bit depth images, with the ground truth of the camera trajectory provided. Specifically, we select 6 sequences which contain 'walking' and 'sitting' from the 'fr3' subset. The images were taken in the 'desk' scene, in which two persons are either walking or sitting, and thus are suitable for testing the efficiency of our algorithm under scenes with dynamic objects.

We compared our algorithm with the original ORB-SLAM2 [5] and DynaSLAM [1] in terms of Absolute Trajectory Error (ATE) [29] which represents the tracking precision by taking the ground truth as reference, and the results are shown in Table I.

It can be seen from Table I that the improvement of the performance of our algorithm on the 'walking' datasets is obvious. In these datasets, ORB-SLAM2 created a lot matches of dynamic feature points due to the movement of the two persons. This enlarges the pose error during optimization. Similar to DynaSLAM, we segmented and discarded the

TABLE I

COMPARISONS OF ATE[M] OF OUR VSLAM MODULE AGAINST THE ORIGINAL ORB-SLAM2 [5] AND DYNASLAM [1].

Sequence	ORB-SLAM2	DynaSLAM	Our vSLAM module		
			median	min	max
Walking_halfsphere	0.351	0.025	**0.019**	0.010	0.028
Walking_static	0.090	0.006	**0.005**	0.0005	0.008
Walking_rpy	0.662	0.035	**0.032**	0.002	0.036
Walking_xyz	0.459	0.015	**0.014**	0.001	0.029
Sitting_halfsphere	0.020	**0.017**	0.021	0.002	0.031
Sitting_xyz	**0.009**	0.015	**0.009**	0.001	0.022

moving objects which contribute to the dynamic points and therefore reach higher precisions. The reason why our algorithm outperforms DynaSLAM is because we refined the segmentation results using 3D pose information and obtained more accurate segmentation regions and boundaries. The enhancement of segmentation precision makes the removal of dynamic points more accurate and thus reduces the pose error. For the 'sitting' datasets, the improvement of our algorithm is not quite obvious, as there are limited dynamic objects in that scene, which do not affect the feature points matching too much.

We also visualized the trajectory that our algorithm outputs with those of ORB-SLAM2 and ground truth in Fig. 3 with green, red and blue respectively. It can be seen that our result exhibits much higher similarity to ground truth than ORB-SLAM2 does.

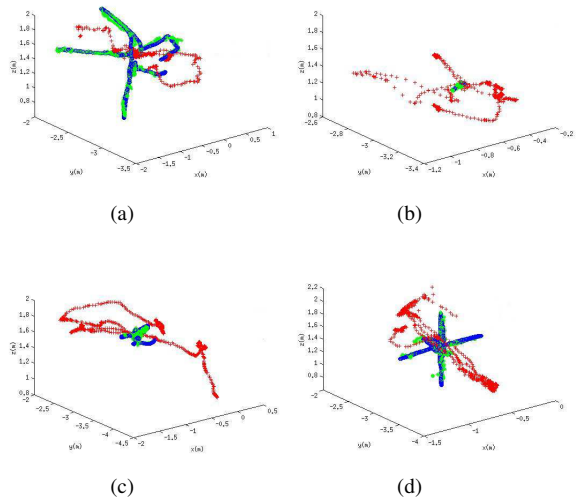


Fig. 3. Comparison of output trajectories of our vSLAM module(in green), ORB-SLAM2 [5](in red) and ground truth(in blue) of the (a)'walking_halfsphere', (b)'walking_static', (c)'walking_rpy' and (d)'walking_xyz' of the TUM dataset [29] respectively.

The average time for the coarse tracking is 6 *ms*, and the fine tracking and mapping takes 22 *ms*.

B. Test Results on ScanNet Dataset

As the ground truth for segmentation is not available in TUM dataset, we used the ScanNet dataset [30] to evaluate the performance of our segmentation module. ScanNet contains 1500 RGBD sequences taken in indoor environment, and has totally 2.5 million images available. The resolutions of RGB images and depth maps are 1296×968 and 640×480 respectively. With the provided extrinsic parameters, each depth map can be mapped to the RGB image. Ground truths of the segmentation is available for every RGB image. As the image sets in ScanNet has 550 object classes, we manually map each class to the MS COCO 80 classes according to its name or general type.

For all the images, we compute the mean Average Precision (mAP) and mean Intersection over Union (mIoU) for the results generated using our segmentation module and the original FCIS [22] algorithm. The results are shown in Table II.

TABLE II

COMPARISON OF FCIS [22] AND OUR SEGMENTATION MODULE ON SCANNET DATASET.

	FCIS	Our segmentation module
mAP	0.6314	**0.6504**
mIoU	0.5620	**0.5751**

It can be seen that the segmentation precision of our module has been improved comparing to that of FCIS [22]. This proves that the use of 3D pose information for the refinement of segmented areas works well as expected.

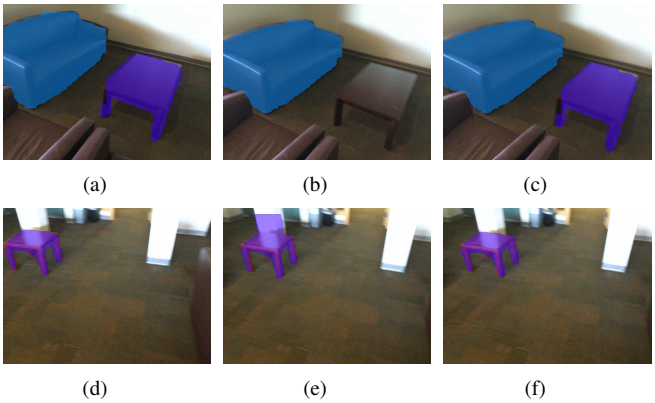


Fig. 4. Examples of the refinement of segmentation. (a)-(c): The results of segmentation of last frame, initial segmentation of current frame with a missing part, and refined segmentation of the current frame respectively. (e)-(f): The results of segmentation of last frame, initial segmentation of current frame with an oversized part, and refined segmentation of the current frame respectively.

We selected two example groups of segmented images to

illustrate the refinement of segmentation of our algorithm in Fig. 4. In Fig. 4(b), a table failed to be segmented probably due to motion blur is added to the refined result (see Fig. 4(c)) by projecting and adding the segmented part of the last frame (see Fig. 4(a)) onto the current one. Fig. 4(f) shows that the oversized segmented area (see the chair in Fig. 4(e)) in the initial segmentation result was shrunk to its correct range by projecting and combining the result in the last frame (see Fig. 4(d)).

The initial segmentation for each frame takes 113 *ms* on average, and the refinement takes about 50 *ms*. The latter process can be further accelerated by utilizing parallel computing or GPU techniques.

C. Test Results on AirSim Generated Dataset

In the above two tests, the results of our algorithm on performing the two tasks have not been tested simultaneously. Meanwhile, there is also a lack of a test on the relocalization performance of the vSLAM module. Therefore, we created a series of sequences using the Microsoft AirSim simulator [31]. It allows the users to control the movement of a car or UAV in a virtual outdoor environment, and collects the RGBD images as well as other sensor data during the process. The exact pose of the camera and also the exact segmentation results can be generated automatically.

To generate the image sequence data, we select totally 40 different routes in a virtual city area, and run two passes with different camera poses and moveable objects(vehicles, pedestrians, etc.) which may either be moving or static along each route, by controlling a virtual car. The resolutions for the RGB and depth images obtained from the virtual camera bound to the car are set to 640×480, with frame rate of 15 *fps*. The lengths of routes range from 160 *m* to 400 *m*. There are totally 16 classes in the segmentation results, and they are also mapped to the MS COCO 80 classes.

TABLE III
COMPARISON OF ORB-SLAM2 [5] AND OUR VSLAM MODULE ON AIRSIM GENERATED SEQUENCES.

	ORB-SLAM2			Our vSLAM module		
	median	min	max	median	min	max
ATE[m]	0.82	0.43	1.03	**0.39**	0.28	0.61

To test the precision of relocalization in vSLAM, we use the long-term map created in the first pass to compute the fine tracking in the second pass, and compare the ATE[m] of our vSLAM module and that of ORB-SLAM2. It can be seen from the results shown in Table III that our vSLAM module is much better than those of ORB-SLAM2. Note that the ATE[m] values of the tracking results of the AirSim generated dataset is much higher than those of the TUM dataset. This is because the areas of the outdoor scenes in AirSim are much larger than those in TUM which are only limited regions indoors.

We show the matching points between two consecutive frames in the generated dataset in Fig. 5. It can be seen that the car contains some feature points which will be mapped to incorrect positions if the car disappears (see Fig. 5(a)). By segmenting the car and excluding the feature points (see Fig. 5(b)) on it during tracking and mapping, the tracking precision when revisiting the same region will be enhanced.

(a)

(b)

Fig. 5. Feature point matching for two adjacent frames with (a) and without (b) segmenting dynamic objects.

We also evaluate the performance on segmentation of our algorithm on all the 40 sequences of the second pass, and list the results in Table IV. It can be seen that by making use of the pose information to refine the initial segmentation result, our algorithm enhances the accuracy of segmentation.

TABLE IV
COMPARISON OF FCIS [22] AND OUR SEGMENTATION MODULE ON AIRSIM GENERATED DATASET.

	FCIS	Our segmentation module
mAP	0.6702	**0.6893**
mIoU	0.6491	**0.6611**

From the results tested on three different datasets, it can be seen that our framework effectively improves the precision of vSLAM and segmentation in both indoor and outdoor environment. The performance promotion of the two modules are more obvious for scenes with objects in motion in the current scan or relocated in further scans.

V. CONCLUSIONS

We present a unified framework for combining the vision-based localization and segmentation tasks for robotics. An accurate pose can be refined from the coarse one by identifying and handling the moving and possibly moveable objects respectively with the help of the initial segmentation result, and it further helps to remedy the errors and boundary inaccuracy of the segmented regions to get a more precise segmentation result. Experimental results on various datasets show that our approach is able to make enhancements to both the localization and segmentation for different environments, especially those with dynamic objects and obvious changes. The proposed framework has the potential to be applied to many robotic applications which use vision sensors for synthesized tasks, including autonomous driving, UAV, logistic robots, etc.

REFERENCES

[1] B. Bescos, J. M. Fcil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[2] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, "A review of semantic segmentation using deep neural networks," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 87–93, 2018.

[3] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 16, 2017.

[4] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007, pp. 225–234.

[5] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[6] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual slam and structure from motion in dynamic environments: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, p. 37, 2018.

[7] J. Klappstein, T. Vaudrey, C. Rabe, A. Wedel, and R. Klette, "Moving object segmentation using optical flow and depth information," in *Pacific-Rim Symposium on Image and Video Technology*, 2009, pp. 611–623.

[8] N. D. Reddy, P. Singhal, V. Chari, and K. M. Krishna, "Dynamic body vslam with semantic constraints," in *Proceedings of the International Conference on Intelligent Robot Systems (IROS)*, 2015, pp. 1897–1904.

[9] M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa, "Mask-slam: Robust feature-based monocular slam by masking using semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 258–266.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis & Machine Intelligence (TPAMI)*, vol. 40, no. 4, pp. 834–848, 2018.

[11] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust dense mapping for large-scale dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[12] F. Zhong, S. Wang, Z. Ziqi, C. Chen, and Y. Wang, "Detect-slam: Making object detection and slam mutually beneficial," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1001–1010.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 3431–3440.

[14] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[15] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[16] T. Wu and W. U. Bajwa, "A low tensor-rank representation approach for clustering of imaging data," *IEEE Signal Processing Letters*, vol. 25, no. 8, pp. 1196–1200, 2018.

[17] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *arXiv preprint arXiv:1802.02611*, 2018.

[18] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[19] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 297–312.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2014, pp. 580–587.

[21] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3150–3158.

[22] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," *arXiv preprint arXiv:1611.07709*, 2016.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *IEEE Transactions on Pattern Analysis & Machine Intelligence (TPAMI)*, 2018.

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence (TPAMI)*, no. 6, pp. 1137–1149, 2017.

[25] Z. Qiu, T. Yao, and T. Mei, "Learning deep spatio-temporal dependence for semantic video segmentation," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 939–949, 2018.

[26] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2015, pp. 4489–4497.

[27] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[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 *European conference on computer vision (ECCV)*, 2014, pp. 740–755.

[29] J. Sturm, E. N., E. F., B. W., and C. D., "A benchmark for the evaluation of rgb-d slam eystems," in *Proceedings of the International Conference on Intelligent Robot Systems (IROS)*, 2012, pp. 573–580.

[30] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 2432–2443.

[31] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.