Proceedings of the 2020 IEEE/SICE
International Symposium on System Integration
Honolulu, Hawaii, USA, January 12-15, 2020

Multi-purpose SLAM framework for Dynamic Environment

Leyuan Sun[1,2], Fumio Kanehiro[2,3,1], Iori Kumagai[2], Yusuke Yoshiyasu[3]

Abstract— **Nowadays, SLAM in the dynamic environment has become a popular topic. This problem is called dynamic SLAM where many solutions have been proposed to segment out the dynamic objects that bring errors to camera tracking and subsequent 3D reconstruction. However, state-of-the-art dynamic SLAM methods face the problems of accuracy and speed, which is due to the fact that one segmentation algorithm cannot guarantee both points at the same time. In this paper, we propose a multi-purpose dynamic SLAM framework to provide a variety of selections for segmentation, each has its applicable scene. The experimental results show that the framework is compatible with different segmentation methods, which improves corresponding existing methods in some aspects.**

I. INTRODUCTION

Two major issues need to be solved when we use Simultaneous Localization and Mapping (SLAM) technology in a dynamic environment. One is the inaccuracy in camera pose tracking because this calculation is based on a strict condition: the position of the point between corresponding pixels used for the calculation is constant in the global space, i.e., the objects must be static and rigid. If the pixels in the field of view are constantly moving, e.g., the regions belonging to the human body in Fig. 1, and these points participate in the pose calculation, they will continue to bring errors to the system, eventually leading to loss of camera track. The other issue is the point cloud corresponding to dynamic objects remains in the final dense point cloud map, which may not be suitable for the subsequent use (the right of Fig. 12).

Although there are several SLAM methods for a dynamic environment, state-of-the-art dynamic SLAM methods face the problems of accuracy and speed, which is because a single segmentation algorithm cannot handle all the types of dynamic objects and guarantee accuracy and speed at the same time. There are basically two kinds of dynamic objects, *moving* and *movable*. *Moving* objects are objects which are actually moving in the scene, whereas *movable* objects are not necessarily moving in the scene. In order to detect movable objects, we need to give prior knowledge about moving objects based on our life experience. Hence a single segmentation method is difficult to handle different purposes of users.

In this paper, we propose a multi-purpose dynamic SLAM framework that is configurable depending on the user's purpose and it is also useful to compare different segmentation methods on a single platform. Considering the above difficulties, we sort out the 3 purposes of SLAM in the dynamic environment. The first one is to obtain an accurate visual localization result. For example, we need this kind of SLAM to evaluate the trajectory of the mobile robot in the warehouse. For this purpose, we should choose the segmentation method for moving objects. The second is to get the dense point cloud map of static objects which can be used for the robot navigation. For this purpose, we should select the segmentation method for movable objects. The last one is for users who would like to do online processing, so the segmentation must be done in real-time. In the proposed framework, the segmentation methods can be replaced easily. Besides, in addition to the sparse keypoint map, it can generate a dense point cloud map of static objects if depth information is available.

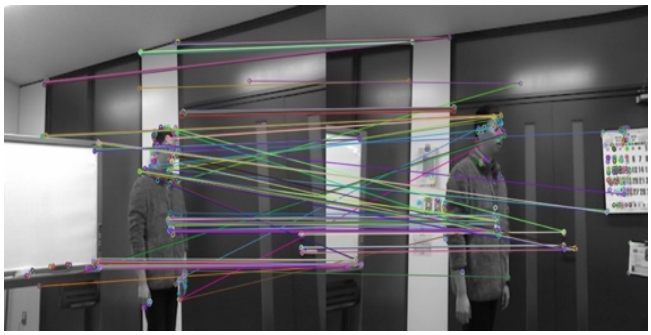

Figure 1. Example of incorrect keypoint matching result in a dynamic environment

This paper is structured as follows. Section II discusses the related work and corresponding category. Section III is about the proposed multi-purpose SLAM framework, Section IV demonstrates some segmentation methods. Section V shows experimental results to compare the presented segmentation methods with their corresponding existing methods. Conclusions and future works are discussed in Section VI.

II. RELATED WORK

Exiting dynamic SLAM methods use segmentation methods to remove dynamic objects from input images. The segmentation methods could be divided into deep learning-based methods and geometry-based methods. In Dynamic-SLAM [1], the authors used the Single-Shot multi-box Detector (SSD) [2] to detect the movable objects. Hence, it could not deal with objects which are moving but not movable, such as a chair being pulled by a human. Another limitation is that the mask detected by SSD is in the shape of a bounding

[1.] Department of Intelligent Interaction Technologies, University of Tsukuba, Ibaraki, Japan.
[2.] Humanoid Research Group, National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan.
[3.] CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/RL, Tsukuba, Japan.

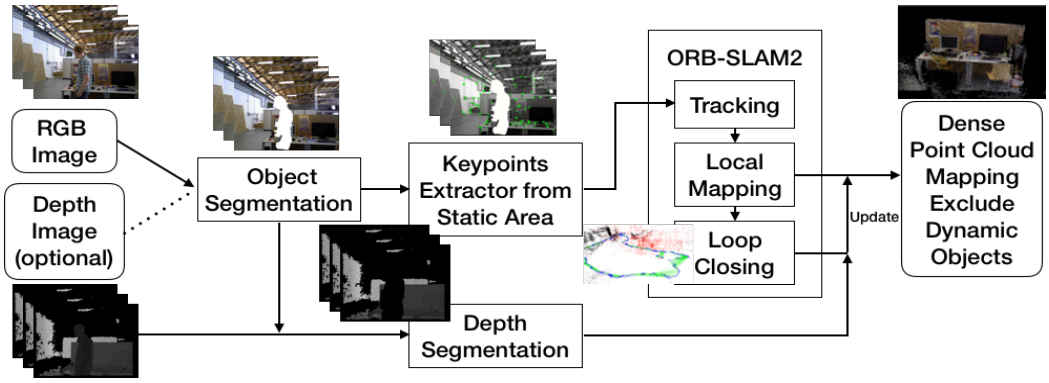

Figure 2. System overview. 'Object Segmentation' and 'Keypoints Extractor' blocks are presented in Section IV, III.B and C respectively.

box which will cause some valid keypoints near the dynamic wobjects to be ignored too. DynaSLAM [3] combines Mask R-CNN [4] and a multi-view geometry method to do object detection and segmentation. The limitation is that Mask R-CNN is not a real-time method.

The traditional geometry-based methods usually depend on the essential matrix or fundamental matrix which represents the correlation between two consecutive frames. As these matrices are usually calculated before the segmentation of dynamic objects, they might be inaccurate. [5] is based on the essential matrix and the threshold needs to be set manually, which may not be an optimal value for all situations. [6] is an RGB-D SLAM in the dynamic environment with a geometry-based segmentation method, which relies on the fundamental matrix and threshold as well. Besides, compared with ORB-SLAM2 [7] in our framework, the back-end system in [6] is only for visual odometry without the global optimization and closed-loop detection parts.

Earlier work on robust SLAM [8] in the dynamic environment does not consider how to segment out the dynamic objects. It proposed a keyframe update method to label the frames invalid when the invalid points are larger than 90% compared to other keyframes. But it cannot deal with a sudden change in most of the scene and the SLAM is limited to work in small space.

TABLE I. CATEGORIES OF RELATED WORKS

Categories	Movable	Moving	Both
Real-time	Dynamic-SLAM [1]	Reference [5]	
Non-realtime		Reference [6]	DynaSLAM [3]

Table I summarizes related works. To the authors' knowledge, a real-time method that deals with both kinds of dynamic objects does not exist. Even if it exists, it is still not suitable for accurate tracking, especially when dynamic objects occupy most of the view field. Considering all, it is difficult to use only one method for all purposes.

III. MULTIPURPOSE SLAM FRAMEWORK

A. Framework overview

The framework overview is shown in Fig. 2. Using RGB frames as input, the localization can be done if the segmentation frames as input, the localization can be done if the segmentation method does not need depth frames. To get the dense point cloud map, depth frames are necessary. The differences from original ORB-SLAM2 are (1) extraction of uniformly distributed keypoints only in the static area and (2) the dense static object point cloud map could be generated.

B. Dilated mask image

We propose different segmentation techniques in Section IV. Here we explain post-processing of mask images with dilation. When we use the mask image to extract keypoints on the region of interest(ROI), many keypoints are extracted on the boundary of the mask and this causes many mismatches as shown in the top line of Fig. 3. In order to solve this problem, we use the dilate function to scale up the mask as shown in the lower row of Fig. 3. After this process, the matching result is significantly improved.

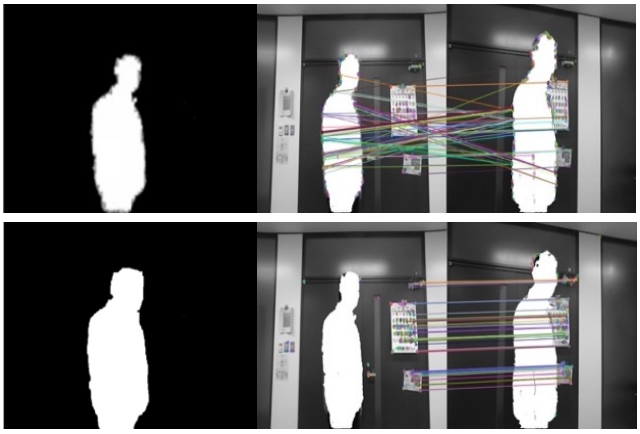

Figure 3. Keypoint matching result obtained using the original mask image (upper row) and the dilated mask image (lower row)

As Fig. 4 shows, when the shape of the mask does not cover the dynamic object accurately, the dilation process also makes up for the inaccuracy of the segmentation result.

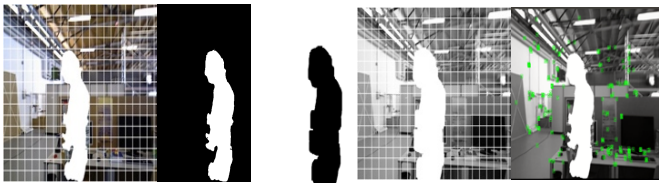

Figure 4. Dilation process makes up the inaccuracy of contour

C. *Uniformly distributed keypoints extraction*

The keypoint extractor used in ORB-SLAM2 cannot use the mask image obtained as the result of dynamic object segmentation. However, the keypoint extractor in OpenCV can use it. The difference between the ORB-SLAM2 extractor and OpenCV extractor is that keypoints extracted by the former method are distributed uniformly as shown in the left of Fig. 5. In contrast, keypoints extracted by the OpenCV extractor concentrate in areas where good features exist as shown in the right of Fig. 5. The advantage of a uniform distribution of keypoints is shown by Absolute Trajectory Error (ATE) [9] comparison shown in Table II, especially on the sequence fr3_siting_halfsphere(fr3_sh) in [10] because in this sequence, the illumination condition is bad and the movement of the camera is intense.

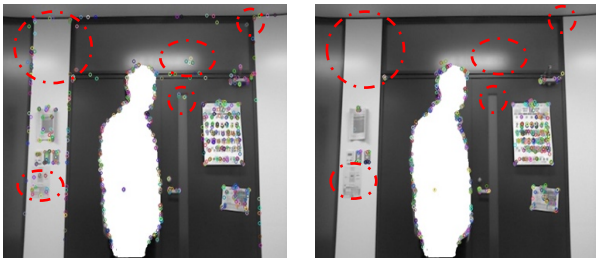

Figure 5. Keypoints extracted by ORB extractor (left) and OpenCV extractor (right)

Considering the above mentioned, we need a keypoint extractor which can use the mask image and extract uniformly distributed keypoints. Our solution is to improve the OpenCV extractor so that it can extract uniformly distributed keypoints. The improved extractor splits both the RGB image and mask image into 30*30 pixels patches first and then extracts the keypoints in each patch with the mask image. The final keypoint extraction result is shown in Fig. 6, where no keypoint is extracted inside the mask and keypoints are distributed uniformly. We also tested the ATE of improved OpenCV extractor. As shown in Table II, its accuracy is almost similar to that of ORB-SLAM2 extractor.

Figure 6. Keypoints extracted by improved OpenCV extractor

TABLE II. ATE COMPARISON BETWEEN ORB-SLAM2 EXTRACTOR, OPENCV EXTRACTOR, AND IMPROVED OPENCV EXTRACTOR (UNIT:[M])

SEQUENCE	ORB-SLAM2	OPENCV	IMPROVED OPENCV
FR1_XYZ	0.0135	0.0120	**0.0115**
FR1_DESK	**0.0193**	0.0219	0.0183
FR2_3H	0.1128	**0.0928**	0.1156
FR2_DESK	0.0072	0.0145	**0.0068**
FR3_LOH	**0.0098**	0.0159	0.0105
FR3_SH	**0.0310**	0.1958	0.0386

Sequences belong to TUM RGB-D dataset [10]

The modified extractor keeps the real-time characteristic of ORB-SLAM2. we extract the maximum of 1000 keypoints in one frame with 5 times, and use the same dataset, with all the same parameter to test the tracking time 5 times, the average processing time is shown in Table III.

TABLE III. COMPUTATIONAL COSTS FOR EXTRACTION AND TRACKING

	Keypoint extraction		Tracking	
	ORB-SLAM2 extractor	**Improved OpenCV extractor**	**ORB-SLAM2 extractor**	**Improved OpenCV extractor**
Processing time [ms/frame]	1.24	1.32	38	43

CPU: i7-7700hq 2.8GHz

D. *Dense point cloud generation*

$$\begin{cases} z = d / s \\ x = (u - c_x) \cdot z / f_x \\ y = (v - c_y) \cdot z / f_y \end{cases} \tag{1}$$

The formula for calculating a 3D point cloud from a 2D color image and a depth image is shown in Eq. (1). u and v represent the coordinates in frame coordinate. x, y, and z represent the coordinates in global coordinate. fx, fy, c_x, c_y and s are the camera parameters. In order to avoid redundancy of 3D point cloud and unnecessary calculation, we only project the extracted keyframes, which are selected in the same way as in ORB-SLAM2. For obtaining the point cloud map of static objects, we use the mask image to segment the depth image, then project the segmented depth image into point cloud map.

IV. SEGMENTATION METHODS

A. *Geometry-based method for a moving object*

The overview of the proposed geometry-based segmentation method (hereinafter, called GS) is shown in Fig. 7. The inputs are RGB and depth frames, the output is a mask image in which the value of pixels corresponding to dynamic objects is set to 255.

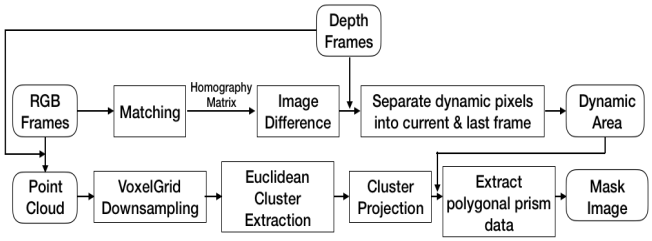

Figure 7. Geometry-based method overview. Rectangles and rounded rectangles represent processing and data respectively.

For detecting the moving objects while the camera is moving, we need to find the projection mapping relationship between the image planes of the previous frame and the current frame, which is called the homography matrix (given by Eq. (2)). In Eq. (2), u_C and v_C represent the coordinates of one pixel in the current frame and u_L and v_L do in the last frame.

$$\begin{pmatrix} u_C \\ v_C \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} \tag{2}$$

After the input RGB frames are processed by grayscale and Gaussian filtering, the homography matrix is calculated by function *findHomography* with RANSAC in OpenCV library.

Algorithm to detect and separate the dynamic object

Input: LastImage, CurrentImage, LastDepth, CurrentDepth, threshold1, threshold2

```
for each pixel (uL,vL) in LastImage do
  (uC,vC) = Homography matrix * (uL,vL)
  diff ← abs(IL(uL,vL) – IC(uC,vC))
  if diff > threshold1
      DiffImg(uL,vL) ← 255
  else
      DiffImg(uL,vL) ← 0
  end if
end for (obtain the first image in Fig. 8)
for each white pixel (u,v) in DiffImg do
  dL ← LastDepth(u,v), dC ← CurrentDepth(u,v)
    if dL – dC > threshold2 then
      Dyna_LastImg(u,v) ← 0, Dyna_CurrentImg(u,v) ← 255
    else if dC – dL > threshold2
      Dyna_LastImg(u,v) ← 255, Dyna_CurrentImg(u,v) ← 0
    else if
      Dyna_LastImg(u,v) ← 0, Dyna_CurrentImg(u,v) ← 0
    end if
end for (obtain the second the third image in Fig. 8)
```

Then all the pixels are traversed to calculate the absolute difference of intensities and it is compared with *threshold1* (I_L and I_C are intensities of the last image and current image respectively) and the image (a) of Fig. 8 is obtained. The white and black pixels correspond to the moving and static objects respectively. This image contains the moving objects of the two frames. To separate them into the image (b) and (c) of Fig. 8, the algorithm uses the change in depth values of the corresponding coordinates and it is compared with the *threshold2* (d_L and d_C represent the depths in the last image and current image respectively). After the separation, all white pixels' coordinates are traversed to get the ranges of u, v and depth shown as the rectangle in the image (c) of Fig. 8.

Figure 8. Dynamic binary pixels separation(from left to right: (a)DiffImg, (b)Dyna_LastImg, (c)Dyna_CurrentImg)

We use the Voxel Grid Filter to downsample the point cloud for ensuring the efficiency of the algorithm. After clustering(the image (a) of Fig. 9), the bounding box of each cluster is obtained by the projection formula in Eq. (3) of the camera pinhole model and compared with the range of the area of dynamic pixels. Finally, the cluster corresponding to a dynamic object is decided.

$$\begin{cases} u = f_x \dfrac{x}{z} + c_x \\ v = f_y \dfrac{y}{z} + c_y \end{cases} \tag{3}$$

Once we have determined the dynamic object cluster, we need to remove it in the original point cloud instead of the downsampled one. The solution is to perform the projection of the downsampled dynamic object cluster as a cross-section, using the Extract Polygonal Prism Data algorithm to remove the dynamic cluster in the original dense point cloud map, the prism's cross-section is the previously projected shape. Then use the point cloud of static clusters to project back to the 2D image plane, the mask image is obtained. As shown the image (b) of Fig. 9, for the static area we set value of pixels to 255.

Figure 9. From left to right: (a)all clusters, (b)static mask image

The limitations of this method are those it assumes that there is only one dynamic object and no occluded situation. Besides, since this method depends on the homography matrix and is sensitive to thresholds, if the dynamic object is dominant in the picture, the obtained homography matrix might be inaccurate. Table IV shows the processing time of each part in this algorithm under CPU i7-7700hq.

TABLE IV. PROCESSING TIME OF EACH PROCESS WITH FR3_WALKING_XYZ FROM TUM[12]

Process	Matching for homography matrix	Image difference	Separation of dynamic pixels	Down sampling	Remove cluster	Total
Processing Time(s)	0.0503	0.0392	0.0135	0.0256	0.318	0.4466

B. Deep learning-based methods for movable objects

We tested two deep learning-based segmentation methods, which are robust to occluded persons situation, called Mask R-CNN and a lightweight deep learning-based method (hereinafter, called LWDL) [11]. Mask R-CNN is one of the instance segmentation methods, which was trained on the COCO dataset [12]. This segmentation method can detect up to 80 different kinds of common objects in everyday life. However, this kind of deep learning-based segmentation method requires the user to provide labels of dynamic objects. As we can see in Table V, LWDL can run in real-time with GPU, whereas Mask R-CNN cannot. However, as shown in Fig. 10, the robustness of LWDL against blurred frame is inferior to that of Mask R-CNN.

Figure 10. Robustness comparison between Mask R-CNN (left) and LWDL (right)

C. Combination of movable and moving objects

If the user's purpose is to segment out both movable and moving objects, the user can combine two segmentation methods. One example is a scene where a human is pulling

TABLE V. COMPUTATIONAL COST COMPARISON USING FR3_WALKING_XYZ FROM TUM[12]

Method	Mask R-CNN	LWDL
Processing Time [ms/frame]	4137(CPU) 673 (GPU)	167 (CPU) 17 (GPU)

CPU: i7-7700hq 2.8GHz/ GPU: Nvidia GTX 1060 6GB

a chair from the desk. In this case, the user can use Mask R-CNN to segment the human first, then GS to segment out the moving chair. More details are shown in Fig. 11. This combination is suitable for obtaining a 3D model of a static environment offline.

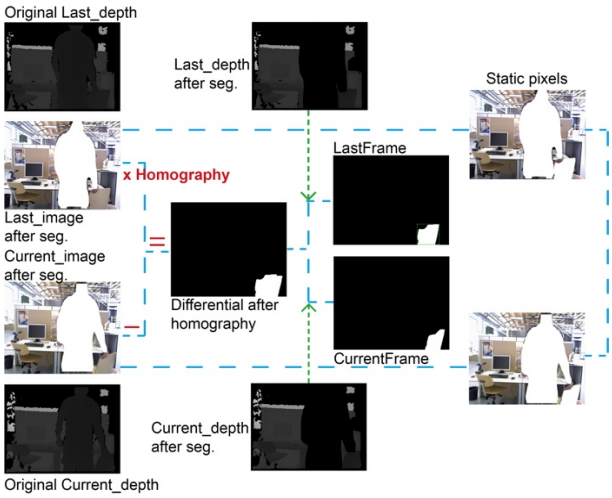

Figure 11. Mask R-CNN + geometry-based method

V. EXPERIMENTAL RESULTS

In this section, we evaluate our framework combined with segmentation methods in Table VI with corresponding related works in Table I. The used dataset is TUM RGB-D dynamic sequence. In the low dynamic situation, two persons sit at a desk, talk, and gesticulate a little bit. In the high dynamic situation, they stand up and walk through an office scene.

TABLE VI. DIFFERENT SEGMENTATIONS

Categories	Movable	Moving	Both
Real-time	LWDL		
Non-realtime	Mask R-CNN	Geometry-based Segmentation (GS)	Mask R-CNN + GS

Table VII is the comparison between Dynamic-SLAM and ORB-SLAM2 with LWDL through ATE and RPE (Relative Pose Error) [13], which is usually used for the evaluation of visual odometry. According to this table, the accuracy of ORB-SLAM2 with LWDL is higher than the Dynamic-SLAM in most sequences. Besides, Dynamic-SLAM is a monocular SLAM and it cannot provide the dense point cloud map. However, if the depth information is available, we can provide the dense point cloud map as in the left of Fig. 12.

To confirm the real-time ability, we set the fps of reading segmentation results as 15. Then, the mean tracking time is found to be 0.064 s, which is shorter than the input rate. Based on the definition of 'real-time ability' in ORB-SLAM2, we could ensure real-time characteristic of this configuration.

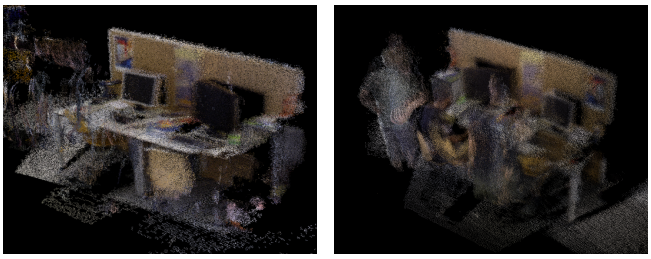

Figure 12. Dense point cloud map by LWDL(left) and GS(right)

For the comparison in the category of non-realtime with moving object segmentation in Table VIII, we used GS. GS could not segment out multi-dynamic objects but, in some sequences in which one moving object occupies large proportion such as *fr3_sitting_halfsphere* in [10] and the low dynamic sequence such as *fr3_ sitting_static* in [10], the ATE and RPE are lower than reference [6]. The back-end of [6] does not contain the global optimization and closure loop detection, but the ORB-SLAM2 in our framework has these features which are beneficial for the accuracy of the SLAM system. Whereas in some multi-high dynamic objects environment, the reference [6] has better performance than GS. Besides, [6] is an RGB-D SLAM but it does not provide a dense point cloud map. The dense point cloud map by GS is shown in the right of Fig. 12. Walking persons are remaining in the point cloud.

The final comparison is between Mask R-CNN with GS and DynaSLAM. Table IX shows that the accuracy is similar because both of them use Mask R-CNN to segment the human, which is the main movable element in frames. This kind of combination is applicable for obtaining the dense point cloud map of static objects as Fig. 13 shows, because Mask R-CNN is more accurate than LWDL for movable objects, as we compared in Fig. 10. Moreover, unlike DynaSLAM, our framework has the update function when ORB-SLAM2 detects the closed-loop.

In the case of original ORB-SLAM2, due to the inaccuracy of camera pose, the point cloud belonging to the backboard is projected into different orientations as shown in the first figure of Fig. 13. The improvement of the dense point cloud map by GS is shown in Fig. 13. The backrest of the moving chair is excluded by GS.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we found that a single segmentation method could not meet the requirements for all purposes of SLAM when using it in a dynamic environment. We proposed a framework which is compatible with different segmentation methods for different purposes and situations. Compared with existing dynamic SLAM methods, our system achieves similar or better performance in some aspects with a single framework.

For future work, with the continuous development of computer vision technology, our framework could be used to compare different segmentation methods for evaluating its performance in the dynamic SLAM field. Besides, we will let the robot understand the environment at a high-level instead of the simple point cloud model.

TABLE VII. ACCURACY COMPARISON BETWEEN DYNAMIC-SLAM[1] AND ORB-SLAM2 WITH LWDL[11] (some data from original paper [1])

TUM RGB-D Datasets Sequence: Dynamic Objects		ATE (cm) RMSE			Translation RMSE (cm/frame) RMSE			Rotation RMSE (deg/frame) RMSE		
		Dynamic-SLAM	LWDL	Improvement	Dynamic-SLAM	LWDL	Improvement	Dynamic-SLAM	LWDL	Improvement
Low Dynamic Environment	fr2/desk_with_person	1.873	0.316(Traj% 85.62)	**83.12%**	1.958	0.422	**78.45%**	0.833	0.283	**66.03%**
	fr3/sitting_xyz	0.601	0.298(Traj% 95.56)	**50.42%**	0.998	0.517	**48.20%**	0.613	0.325	**46.98%**
	fr3/sitting_halfsphere	1.461	0.992(Traj% 71.08)	**32.10%**	1.451	0.828	**42.94%**	0.551	0.439	**20.32%**
	fr3/sitting_rpy	3.448	4.023(Traj% 61.08)	-16.68%	4.303	4.832	-12.29%	0.991	1.231	-24.22%
High Dynamic Environment	fr3/walking_xyz	1.324	2.021(Traj% 98.49)	-52.64%	1.796	0.970	**45.99%**	0.598	0.526	**12.04%**
	fr3/walking_halfsphere	2.139	0.434(Traj% 86.79)	**79.71%**	2.192	0.813	**62.91%**	0.666	0.408	**38.74%**
	fr3/walking_rpy	6.025	0.669(Traj% 81.10)	**88.90%**	5.605	0.544	**90.29%**	1.149	0.429	**62.66%**

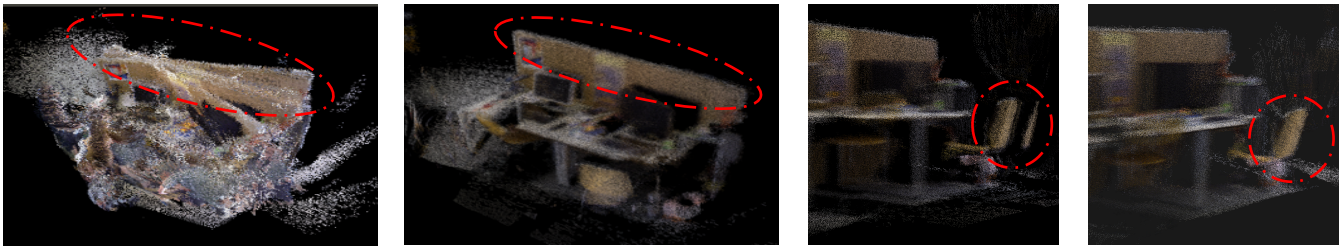

Figure 13. (1-4 from left to right)Dense point cloud map generated by (a)ORB-SLAM2, (b)Mask R-CNN+GS, (c) and (d) are the details comparison of a chair back between only Mask RCNN and Mask R-CNN+GS.

TABLE VIII. ACCURACY COMPARISON BETWEEN [6] AND ORB-SLAM2 WITH GS

Sequence	ATE (m) RMSE		RPE Translation(m/s) /Rotation(deg/s) RMSE	
	Reference[6]	GS	Reference[6]	GS
fr3_sittting_static	0.0066	**0.0052**	0.0077/0.2595	**0.0058/0.2342**
fr3_sitting_halfsphere	0.0196	**0.0185**	0.0245/**0.5643**	**0.0211**/0.6023
fr3_walking_static	**0.3080**	0.5532	**0.1881/3.2101**	0.3011/4.5342

TABLE IX. ACCURACY COMPARISON BETWEEN DYNASLAM AND ORB-SLAM2 WITH MASK R-CNN AND GS

Sequence (under RGB-D)	DynaSLAM[3]	ORB-SLAM2 with Mask R-CNN + GS	Improvement
	RMSE (m)	RMSE (m)	%
w_halfsphere	0.025	**0.024**	**4**
w_xyz	0.015	0.017	-13.33
w_rpy	0.035	0.036	-2.857
w_static	0.006	0.006	0
s_halfsphere	0.017	**0.015**	**11.76**
s_xyz	0.015	0.015	0

REFERENCES

[1] Xiao, Linhui, et al, "Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robotics and Autonomous Systems*, vol. 117, 2019, pp. 1-16.

[2] Liu W, et al, "SSD: Single Shot MultiBox Detector," *European conference on computer vision*, 2016, pp. 21-37.

[3] Bescos, Berta, et al. "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters* vol. 3, no.4, 2018, pp. 4076-4083.

[4] He, Kaiming, et al, "Mask R-CNN," *The IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2961-2969

[5] Cheng, Jiyu, Yuxiang Sun, and Max Q-H. Meng, "Improving monocular visual SLAM in dynamic environments: an optical-flow-based approach," *Advanced Robotics* vol. 33, no.12, pp. 576-589

[6] Wang, Runzhi, et al, "A New RGB-D SLAM Method with Moving Object Detection for Dynamic Indoor Scenes," *Remote Sensing*, vol. 11, no.10, 2019, pp. 1143.

[7] Mur-Artal, Raul, and Juan D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," I*EEE Transactions on Robotics*, vol. 33, no.5, 2017, pp. 1255-1262.

[8] Tan, Wei, et al, "Robust monocular SLAM in dynamic environments," *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013, pp. 209-218.

[9] Olson, Edwin, and Michael Kaess, "Evaluating the performance of map optimization algorithms," *RSS Workshop on Good Experimental Methodology in Robotics*, vol. 15, 2009.

[10] Sturm, Jürgen, et al, "A benchmark for the evaluation of RGB-D SLAM systems," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573-580.

[11] https://github.com/AntiAegis/Human-Segmentation-PyTorch

[12] Lin T Y, Maire M, Belongie S, et al, "Microsoft coco: Common objects in context," *ECCV*, 2014, pp. 740-755.

[13] Konolige, Kurt, Motilal Agrawal, and Joan Sola, "Large-scale visual odometry for rough terrain," *Robotics research*, 2010, pp. 201-212.