A Real-Time Dynamic Object Segmentation Framework for SLAM System in Dynamic Scenes

Jianfang Chang, Na Dong, and Donghui Li

Abstract—**To accurately detect dynamic objects in dynamic scenes (DSs), a detection framework equipped with visual-based measurement methods has been proposed in this article. First, to segment dynamic objects in real time, the real-time instance segmentation network, You Only Look At CoefficienTs (YOLACT), has been introduced. Second, the geometric constraints have been utilized to further filter the missing dynamic feature points outside the segmentation mask. The dense optical flow method with adaptive threshold has been introduced to detect the missing dynamic objects driven by humans. Third, a background inpainting strategy has been proposed to restore the features occluded by dynamic objects. In order to verify the effectiveness of the dynamic object detection, the proposed method has been embedded in the visual simultaneous localization and mapping (SLAM) system to improve its performance in dynamic environments. Experiments performed on the Technische Universität München (TUM) and KITTI datasets have proved that the proposed detection method has an excellent performance in DSs, which is of great significance to improve the robustness of the SLAM system.**

Index Terms—**Background inpainting, geometric constraint, instance segmentation network, optical flow, visual-based measurement.**

I. Introduction

Dynamic object detection in dynamic scenes (DSs) is a research hotspot in computer vision. Visual-based measurement methods are mostly based on computational intelligence, especially machine learning, pattern recognition, and pattern matching [1]. Many visual-based measurement methods are used to detect dynamic objects in DSs.

The robust constraints and random sample consensus (RANSAC) have been applied to remove outliers (dynamic points), while sparse feature points cannot reflect the boundaries of dynamic objects. Geometric constraint [2] can verify dynamic feature points; the pose of the camera needs to be known in advance. Object detection network can detect the bounding rectangle of the *a priori* dynamic objects [3]. Feature points between the dynamic object and the bounding rectangle will be discarded, which wastes the necessary feature points. Mask-region convolutional neural networks (RCNNs) is an excellent semantic segmentation network, with GPU acceleration, and its speed is about 5–10 FPS (100–200 ms). However, 30 FPS (33 ms) is desirable in real-time or near real-time simultaneous localization and mapping (SLAM) applications. Although the performance of the semantic segmentation network, You Only Look At CoefficienTs (YOLACT), is slightly inferior to Mask-RCNN [4], its speed reaches 30 FPS (33 ms). Thus, the YOLACT has been embedded in the SLAM system in this article.

Manuscript received June 10, 2021; revised July 20, 2021; accepted August 25, 2021. Date of publication September 1, 2021; date of current version September 14, 2021. This work was supported by the National Natural Science Foundation of China under Grant 61773282. The Associate Editor coordinating the review process for this article was Dr. Anirban Mukherjee. *(Corresponding author: Na Dong.)*

The authors are with the School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: dongna@tju.edu.cn).

Digital Object Identifier 10.1109/TIM.2021.3109718

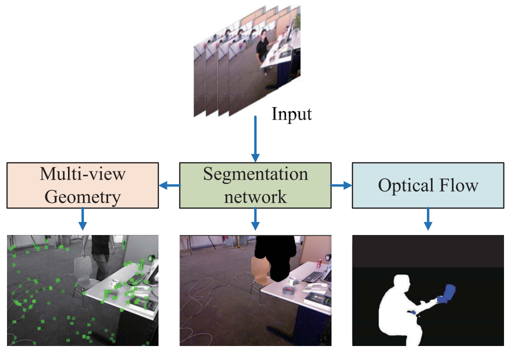

Fig. 1. Framework of the dynamic object detection.

In this article, a fusion dynamic object detection framework is proposed, as shown in Fig. 1, which can meet a variety of dynamic detection requirements. First, a real-time instance segmentation network, YOLACT, is applied to detect the boundary of the *a priori* dynamic object. Second, the geometric constraints are utilized to further filter the missing dynamic feature points outside the segmentation mask. Third, the dense optical flow [5] with adaptive threshold is proposed to detect dynamic objects driven by humans (books or cups in human hands).

Some advanced visual SLAM has been designed and achieved satisfactory performance in static scenarios [4], [5], including ORB-SLAM2 [6], large scale direct monocular (LSD)-SLAM [7], and RGB-D-SLAM [8]; the systems that run continuously in a densely populated interactive environment will be restricted [9].

The above dynamic object detection method has been embedded in the SLAM system to achieve accurate tracking and mapping in DSs. The following innovation points have been proposed in this article.

1) The real-time semantic segmentation network, YOLACT, can achieve real time with GPU acceleration;

YOLACT has been embedded in the SLAM system to segment dynamic objects in real time.

2) Monocular and Stereo introduce epipolar geometric constraints to filter the missing dynamic feature points, and the RGB-D uses multiview geometric constraints to filter the missing dynamic feature points.
3) The dense optical flow method with adaptive threshold has been proposed to detect dynamic objects driven by humans (books or cups in human hands).
4) The static feature points are introduced to construct the 3-D point cloud map.

The rest of the article is structured as follows. Section II describes the related work. Section III shows the measurement/detection methods. The YOLACT-based SLAM system is discussed in Section IV. Section V details the experimental results. Section VI presents the conclusions and future works.

II. Related Work

Researchers defined dynamic objects as outliers and applied robust constraints (e.g., Bundle Adjustment) [6], [7], [10] or RANSAC [11] to remove outliers. Gutiérrez-Gómez *et al.* [12] compare different robust functions focusing on the quality of the resulting pose estimate, while Kerl *et al.* [13] demonstrate robustness to the presence of small moving objects in the scene. These solutions can effectively remove slight dynamic (objects) feature points from a highly dynamic environment, while they are difficult to filter highly dynamic feature points.

In DSs, if the extracted feature points belong to a moving object, the points will violate the geometric constraints [14]. Yuxiang Sun *et al.* [15] applied the optical flow to match two frames to obtain a homograph matrix; aligning the background of the two frames, the codebook method is used to segment dynamic objects.

With the development of deep learning, convolutional neural networks (CNNs) have been applied to detect dynamic objects [16]. The object detection and semantic segmentation can mark potential dynamic objects, and the tracking (if there is a tracking thread and relies on feature points) will not extract feature points from these dynamic objects [17].

On the one hand, object detection networks have been introduced to dynamic object detection. Xiao *et al.* [18] proposed a Single Shot MultiBox Detector (SSD) object detector-based semantic SLAM framework to make the system suitable for a dynamic environment. Zhong *et al.* [19] presented a novel robotic vision system, which integrates SLAM with an object detector to make object detection mutually beneficial.

On the other hand, semantic segmentation and instance segmentation networks are introduced to detect the boundaries of dynamic objects. Bescos *et al.* [17] proposed DynaSLAM, in which the Mask-RCNN [20] has been applied to detect dynamic objects. However, MaskRCNN cannot be real time, and DynaSLAM does not design geometric constraints for Monocular and Stereo. Zhang *et al.* [21] proposed a novel semantic SLAM framework to achieve robustness in DSs for RGB-D camera, which detected potentially dynamic objects by Mask R-CNN. Zhao *et al.* [22] proposed a workflow to segment the objects accurately, which will be marked as the potentially dynamic-object area based on the semantic information. Zhang *et al.* [23] proposed a CNN model to improve the accuracy of terrain segmentation, which has been applied to autonomous robot navigation in wild environments. Ai *et al.* [24] proposed the dynamic deep learning (DDL)-SLAM, a robust RGB-D SLAM system for dynamic scenarios that adds dynamic object segmentation (DOS) and background inpainting. Yu *et al.* [25] proposed DS-SLAM that combines semantic segmentation network with moving consistency check method to reduce the impact of dynamic objects, and thus, the localization accuracy is highly improved in dynamic environments. Han and Xi [26] combined the visual SLAM and pyramid scene parsing network (PSPNet) and propose a PSPNet-SLAM system, which uses the optical flow and semantic segmentation to detect and eliminate dynamic points.

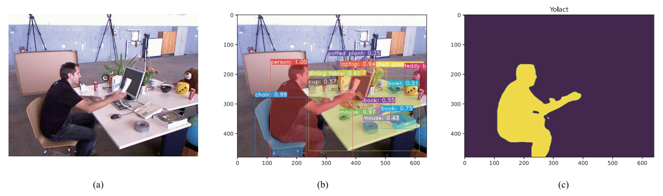

Fig. 2. Input and output of the segmentation network. (a) RGB images. (b) Output image of the YOLACT. (c) Masks of the a priori dynamic objects.

III. Dynamic Object Detection Framework

The YOLACT has been applied to detect dynamic objects. In order to compensate for the missing areas, on the one hand, geometric constraints can be applied to filter outlier feature points, and on the other hand, dense optical flow can reflect the movement in the scene and can be used to infer dynamic regions. The static information in the previous frames has been applied to repair the dynamic mask.

A. Segmentation of Potentially Dynamic Object Using YOLACT

The YOLACT [27], [28] trained on Microsoft (MS) common objects in context (COCO) [29] could segment up to 80 classes. Since its segmentation speed is about 30 FPS with GPU, the DOS can be (near) real time.

The following labels are set as the *a priori* dynamic objects—person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, and giraffe.

As shown in Fig. 2(a), the RGB image is fed to the network. Fig. 2(b) shows the output image of the YOLACT, which contains the bounding rectangles, masks, labels, and confidences of *a priori* dynamic objects. The masks of the *a priori* dynamic objects are synthesized into the final mask, as shown in Fig. 2(c).

B. Geometric Constraint

As shown in Fig. 3, F_n and F_{n+1} are adjacent frames. x and x' mean the observation of the same 3-D points (P and P') on dynamic objects. C and C' represent optical center. e, e' and l, l' individually mean epipolar points and epipolar lines. $\hat{x}$ represent the desired observation of static feature point P. The depth has been captured by RGB-D camera.

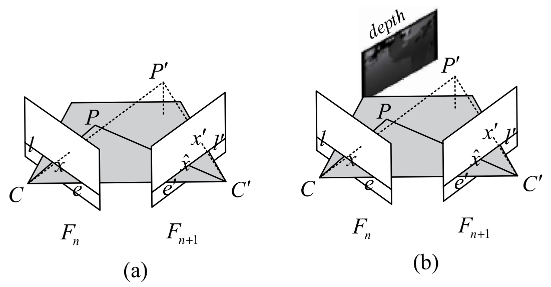

Fig. 3. Epipolar constraint and multiview geometric constraint. (a) Epipolar constraint for monocular and stereo. (b) Multiview geometric constraint for RGB-D.

Two kinds of geometric constraints are designed for different types of cameras.

In the Monocular and Stereo Cases: The epipolar geometric constraint has been applied to filter dynamic feature points, which requires that the matched feature point need to be located on the epipolar line of the current frame [14]. The epipolar constraint can be described as follows:

$$x'^T Ex = 0 \quad \text{and } l' = Ex \tag{1}$$

where x and x' indicate normalized coordinates of P and P', and E means the essential matrix. If the vertical distance between the x' and the epipolar line l' exceeds the threshold, x' will be defined as the dynamic point.

In the RGB-D Case: The multiview geometry can be expressed as follows:

$$s_2\hat{x} = s_1 Rx + t \tag{2}$$

where s_1 and s_2 are the depth of point P in the two frames, respectively. x and $\hat{x}$ indicate normalized coordinates of P in two frames. When $|s_2 - \text{depth}|$ or $|x' - \hat{x}|$ exceeds the threshold, x' is considered as a dynamic point.

In practical applications, when the Euclidean distance of a pixel exceeds 4 or the depth difference exceeds 0.1 m, the feature point will be removed. 4 and 0.1 are empirical parameters. When these thresholds are large, dynamic feature points are difficult to remove, and when these thresholds are relatively small, static feature points are easily filtered out.

C. Dense Optical Flow

Two consecutive adjacent frames have been shown in Fig. 4(a) and (b).

The Farneback [5] is applied to detect the dense optical flow of adjacent frames. Assuming that the optical flows in the x- and y-directions of the pixel p_i are u_i and v_i, respectively, then $\text{pixel}_i = ({u_i}^2 + {v_i}^2)^{1/2}$ is recorded as the intensity of the optical flow of p_i. Normalize the optical flow intensity of all pixels to [0, 255], and the gray image of optical flow intensity can be obtained. To enhance the contrast, as shown in Fig. 4(c), the gray image is filled with colors, and the cool colors represent areas with obvious movement.

The microphone is driven by the arm and moves slowly. The boundary of the person is accurately segmented by YOLACT, as shown in Fig. 4(d), and the moving microphone in the hands is not detected.

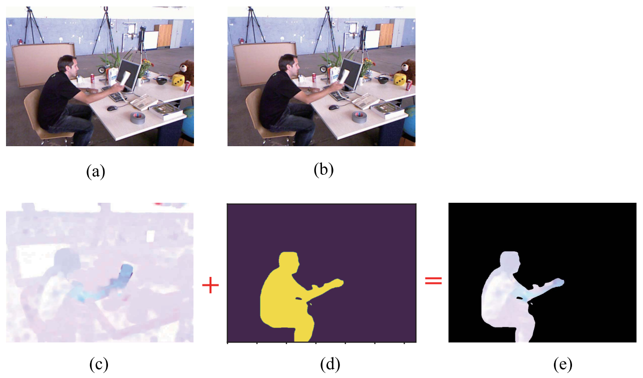

Fig. 4. Two consecutive adjacent frames and optical flow with mask. (a) First of two consecutive adjacent frames. (b) Second of two consecutive adjacent frames. (c) Optical flow. (d) Mask. (e) Dynamic objects.

In order to separate human-driven objects in the optical flow, the Adaptive threshold optical flow has been proposed in this section.

1) Since the motion of the static object is triggered by the *a priori* dynamic object, the mask of the dynamic object overlaps with the object in the optical flow, as shown in Fig. 4(c) + (d) = (e).
2) Assuming that there are n pixels in Fig. 5(a), the average value $\bar{p}$ of the optical flow inside the dynamic mask is given as follows:

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{n} \text{pixel}_i = \frac{1}{n}\sum_{i=1}^{n}\sqrt{u_i^2 + v_i^2}. \tag{3}$$

   $2\bar{p}$ has been defined as the adaptive threshold, with the mask of the semantic segmentation network, which adaptively calculates the threshold for different frames.
3) The adaptive threshold $2\bar{p}$ has been introduced to Fig. 5(a); the arm that moves faster than the body will be displayed, as shown in Fig. 5(b).
4) The adaptive threshold $2\bar{p}$ has been introduced to optical flow [see Fig. 5(c)]; the human-driven regions in the optical flow will be displayed, as shown in Fig. 5(d).
5) The dynamic regions have been jointly constructed with the mask detected by YOLACT (white) and the regions generated by optical flow (blue), as shown in Fig. 5(b) + (d) = (e).

D. Dynamic Mask Inpainting

Dynamic mask inpainting is dedicated to filling the above dynamic mask with the static information in the previous frames.

Historical keyframe F_k contains RGB, depth depth_k, and transformation matrix T_{cw_k}. The transformation matrix of the current frame F_c is T_{cw_c}, and its depth is depth_c. The historical keyframe F_k is mapped to the current frame F_{c_map}, as follows:

$$F_{c\_\text{map}} = K\left(T_{\text{cw}\_c}\left(T_{\text{cw}\_k}^{-1}\left(K^{-1}F_k\right)_{\text{depth}\_k}\right)\right)_{/\text{depth}\_c}. \tag{4}$$

Therefore, the RGB of F_{c_map} can be used to complement the mask in the current RGB image, as shown in Fig. 6. In experiments with the Technische Universität München

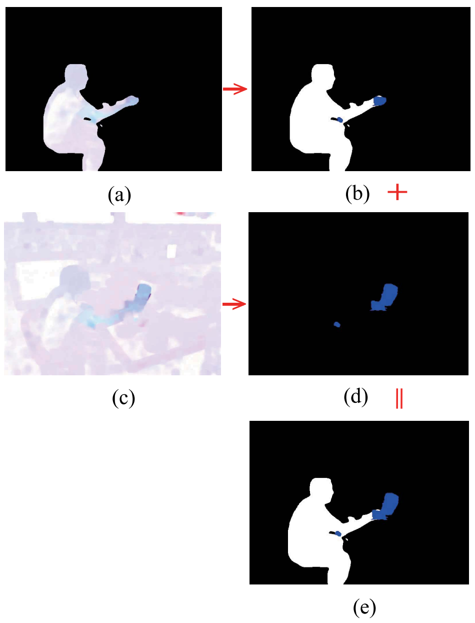

Fig. 5. Adaptive threshold optical flow for dynamic region detection. (a) Dynamic objects. (b) Obviously move. (c) Optical flow. (d) Human-driven. (e) Dynamic regions.

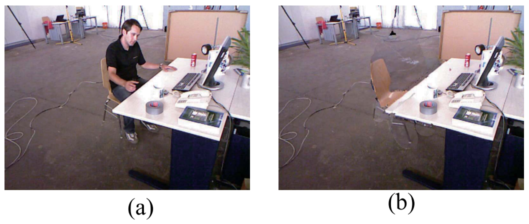

Fig. 6. Results of dynamic mask inpainting. (a) RGB original image. (b) Inpainted image.

(TUM) dataset, we found that the current frame can be almost completely filled with 10–20 historical keyframes; however, backgrounds not presented in historical keyframes cannot be repaired.

IV. SLAM System for Dynamic Environments

The above dynamic object detection method has been embedded in the SLAM system to achieve accurate tracking and mapping.

The overview of the YOLACT-based SLAM has been illustrated in Fig. 7. The RGB channels are fed into the YOLACT, and the pixel-wise mask (Mask_Y) corresponding to the dynamic objects can be obtained. In order to get a preliminary rough pose, the feature points outside the mask are utilized to slightly tracking. Then, the geometric constraints have been introduced to further filter the missing dynamic feature points.

In the Monocular and Stereo Cases: With blue solid line, static feature points filtered by epipolar geometric constraint are used for tracking and mapping.

In the RGB-D Case: Static feature points filtered by multiview geometric constraints are used for tracking and mapping. The dense optical flow has been applied to compensate YOLACT. Since RGB-D has depth information, we aim to reconstruct the occluded background and build a point cloud map and octree map with the static information.

It is worth noting that background inpainting is only used for visualization in our article, and the filled (inpainting) image is no longer used for camera pose tracking.

A. Slightly Tracking

The slightly tracking implemented at this stage estimates the pose of the camera between adjacent frames.

In the Monocular and Stereo Cases: The epipolar geometry has been applied to solve the camera pose. The description of epipolar geometry is given as follows:

$$x_2^T E x_1 = 0 \tag{5}$$

where E is the essential matrix. $E = \hat{t}R$, where R is the rotation matrix and t is the translation. x_1 and x_2 are normalized coordinates of the previous frame and the current frame, respectively.

The eight-point algorithm has been applied to solve the Essential Matrix E. The singular value decomposition (SVD) has been performed on Essential Matrix E to solve the rotation matrix R and translation t. When the solution fails or $t = 0$, the Slightly Tracking and multiview geometry of the current frame are skipped.

In the RGB-D Case: The Perspective–n-Point (PnP) has been applied to solve the rotation and translation.

B. Tracking and Sparse Mapping

The tracking and mapping are based on ORB-SLAM2. The feature points falling in the segmentation mask and filtered by geometric constraints have been removed, and the static feature points are fed back to the tracking and mapping process.

The operation interface of the YOLACT-based SLAM system is shown in Fig. 8.

V. Experiment Results

The TUM [31] and KITTI [32] datasets have been introduced to evaluate the YOLACT-based SLAM in dynamic environments. The TUM has been recorded with a MS Kinect sensor in different indoor scenes at a full frame rate (30 Hz). Both the RGB and the depth images are available, together with the ground-truth trajectory. The resolution of RGB and depth is 640 × 480, and the ground-truth trajectory is recorded by the high-speed capture device. The KITTI contains stereo sequences recorded from a car in urban and highway environments. The frame rate is 30 Hz, and the resolution of the left and right images is 1241 × 376.

Mur-Artal and Tardós [6] proposed to run each sequence five times and show median results. Furthermore, ten experiments have been implemented in this article, as dynamic objects tend to cause instability. The YOLACT-based SLAM is executed on Ubuntu 18.04 system, which is equipped with i7-9700K CPU, NVIDIA RTX2080 GPU, and 48-GB RAM.

A. TUM Dataset

The root mean squared error (RMSE) of absolute trajectory error (ATE) has been widely applied to evaluate the system

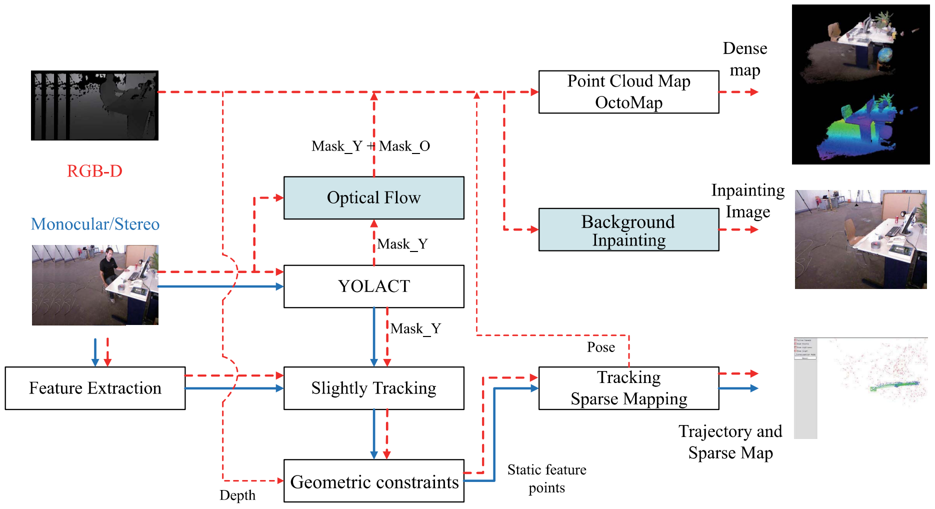

Fig. 7. Flowchart of the YOLACT-based SLAM system.

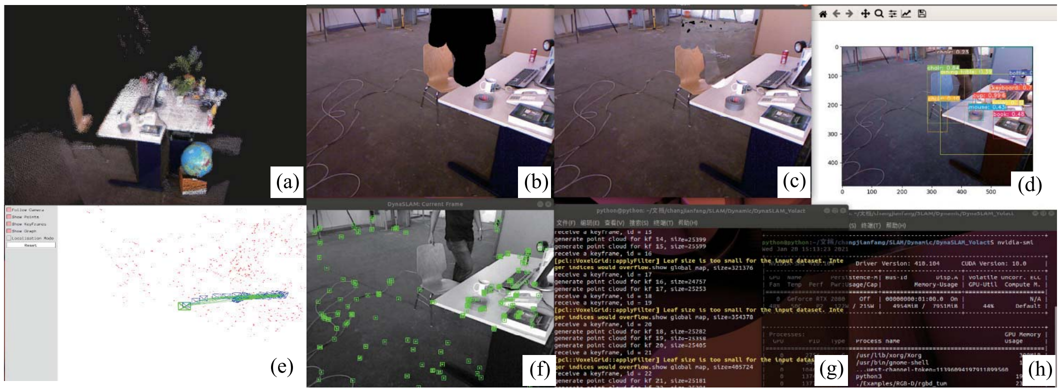

Fig. 8. Operation interface of YOLACT-based SLAM. (a) Point cloud viewer. (b) Mask of dynamic object. (c) Inpainting. (d) Object detection and segmentation. (e) Tracking. (f) Current frame. (g) Operating status of SLAM system. (h) GPU information.

TABLE I

ATE RMSE [m] OF YOLACT-BASED SLAM WITH DIFFERENT COMPONENTS

	ORB-SLAM2	SLAM only with YOLACT	SLAM only with multi-view geometry	YOLACT based SLAM
w_halfsphere	0.351	0.026	0.132	**0.026**
w_xyz	0.459	**0.014**	0.326	0.017
w_rpy	0.622	0.043	0.461	**0.038**
w_static	0.09	0.011	**0.008**	0.009
s_halfsphere	0.02	0.016	0.021	**0.016**
s_xyz	**0.009**	0.013	0.016	0.012
Average Error	0.2585	0.0205	0.160666667	**0.019666667**
Improvement	-	92.07%	37.85%	**92.40%**

performance [31], which is defined as a unified indicator to evaluate the performance of compared SLAM systems.

The YOLACT-based SLAM with different components has been applied to run with different sequences. The RMSE(s) of ATE have been recorded in Table I.

According to Table I, the YOLACT filters out most of the dynamic feature points, so the performance of the SLAM system (only) with YOLACT is better than ORB-SLAM2 in a dynamic sequence. The performance of the SLAM system (only) with multiview geometry is also slightly better than ORB-SLAM2, which proves the effectiveness of multiview geometry. Since the performance of the YOLACT-based SLAM system is better than that of the system (only) with YOLACT, the multiview geometry is beneficial to improving system performance.

When the camera is "static," outliers only come from dynamic objects, and the multiview geometry is more suitable for filtering these dynamic points. Therefore, the SLAM system (only) with multiview geometry has achieved better performance on w_static. When the scene is the low degree of motion (s_halfsphere, s_xyz), ORB-SLAM2 is not disturbed by dynamic points, and the performance of the YOLACT-based SLAM achieved similar performance.

In order to clearly show the improvement of different steps, the average error of the SLAM system in different datasets has been recorded. Compared with ORB-SLAM2, SLAM only with YOLACT increased by 92.07%, SLAM only with multiview geometry increased by 37.85% and YOLACT-based SLAM increased by 92.40%. YOLACT-based SLAM achieved the best performance.

Compared with the state-of-the-art SLAM systems, the ATE RMSE(s) [m] of YOLACT-based SLAM systems have been shown in Table II. Most of the state-of-the-art SLAM systems perform better than ORB-SLAM2 in dynamic scenarios, and the performance of several top SLAM systems is similar.

TABLE II
ATE RMSE [m] OF YOLACT-BASED SLAM AGAINST THE STATE-OF-THE-ART SLAM IN DSS

Dataset/SLAM System	w_halfsphere	w_xyz	w_rpy	w_static	s_halfsphere	s_xyz
VO-SF [33]	0.482	0.874	-	0.327	0.18	0.111
ElasticFusion (EF) [34]	0.486	0.906	-	0.293	0.428	0.022
Co-Fusion (CF) [35]	0.756	0.696	-	0.551	0.036	0.027
StaticFusion (SF) [36]	0.063	0.127	-	0.014	0.04	0.04
Depth Edge SLAM [37]	0.049	0.06	0.179	0.026	0.043	0.04
Motion Segmentation DSLAM [38]	0.055	0.04	0.076	0.024	-	-
Motion Removal DVO-SLAM [39]	0.125	0.093	0.133	0.066	0.047	0.048
ORB-SLAM2 [6]	0.351	0.459	0.662	0.09	0.02	0.009
DynaSLAM [17]	**0.025**	**0.015**	0.035	0.006	0.017	0.015
Mask-RCNN based SLAM [21]	0.0364	0.0328	0.0792	0.0162	0.017	0.015
DOS-SLAM [40]	0.029	0.0394	0.5736	0.0088	**0.0144**	**0.0088**
DS-SLAM [25]	0.0313	0.0265	0.5977	0.0081	-	-
PSPNet-SLAM [26]	0.0256	0.0156	0.0334	0.0073	-	-
DDL-SLAM [24]	0.029	0.018	0.041	0.006	0.019	-
SDF-SLAM [41]	0.0367	0.019	**0.0137**	**0.0053**	-	-
SOF-SLAM [42]	0.029	0.018	0.027	0.007	-	-
MGC-SLAM [43]	0.0253	0.0164	0.1326	0.0071	0.0151	0.012
YOLACT based SLAM	0.026	0.017	0.038	0.009	0.016	0.012

TABLE III
IMPROVEMENTS OF SLAM SYSTEMS COMPARED WITH ORB-SLAM2

	w_halfsphere	w_xyz	w_rpy	w_static	s_halfsphere	s_xyz	Average Improvement
DynaSLAM [17]	**92.88%**	**96.73%**	94.71%	93.33%	15.00%	-66.67%	54.33%
Mask-RCNN based SLAM [21]	89.63%	92.85%	88.04%	82.00%	15.00%	-66.67%	50.14%
DOS-SLAM [40]	91.74%	91.42%	13.35%	90.22%	**28.00%**	**2.22%**	52.83%
DS-SLAM [25]	91.08%	94.23%	9.71%	91.03%	-	-	-
PSPNet-SLAM [26]	92.71%	96.60%	94.96%	91.91%	-	-	-
DDL-SLAM [24]	91.74%	96.08%	93.81%	93.33%	5.00%	-	-
SDF-SLAM [41]	89.54%	95.86%	**97.93%**	**94.11%**	-	-	-
SOF-SLAM [42]	91.74%	96.08%	95.92%	92.22%	-	-	-
MGC-SLAM [43]	92.79%	96.43%	79.97%	92.11%	24.50%	-58.89%	54.49%
YOLACT based SLAM	92.59%	96.30%	94.26%	90.00%	20.00%	-33.33%	**59.97%**

In order to show the improvements of the extended SLAM systems in a dynamic environment, the improvements of SLAM systems, compared with ORB-SLAM2, have been recorded in Table III. In highly dynamic scenarios (w_halfsphere, w_xyz, w_rpy, and w_static), the improvements of the extended SLAM systems are about 90%. Among them, DOS-SLAM and DS-SLAM have only about 7%–13% improvement in"w_rpy."

The performance of the YOLACT-based SLAM system is improved by more than 90% on all dynamic sequences, and its performance is very close to the top SLAM systems. The experimental results can prove that the YOLACT-based SLAM system is excellent and reliable. In the slightly DSs (s_halfsphere, s_xyz), the deep learning network removes the *a priori* dynamic points (real static), and the reduction of effective feature points will affect the accuracy of the solution. Thus, the improvement of the extended SLAM systems is not obvious or even degradation.

The average performance improvements of the SLAM systems on all sequences have been recorded in Table III, and the average performance improvements of the YOLACT-based SLAM are the most obvious. The uncertainty of dynamic objects and the randomness of the system will also affect the performance indicators; therefore, the YOLACT-based SLAM has reached the range of the top SLAM system.

B. KITTI Dataset

The RMSE of ATE, the translation part, and the rotation angle of relative pose error (RPE) [32] have been applied to evaluate the system performance.

1) In the Monocular Cases: In order to verify the performance of the YOLACT-based SLAM in monocular mode, ORB-SLAM, ORB-SLAM2 [6], DynaSLAM [17], and Dynamic-SLAM [18] have been reconstructed to establish comparison experiments. The performance indexes are shown in Table IV, and the RMSE, Mean, Median, and Standard deviation of the ATE are given.

According to Table IV, the YOLACT-based SLAM system achieves the best performance on six sequences. The YOLACT filters out static feature points on static vehicles, which drags down the system performance. Therefore, except for sequences 03 and 08, the performance of the YOLACT-based SLAM system is better than ORB-SLAM2.

2) In the Stereo Cases: In order to verify the performance of the YOLACT-based SLAM in stereo mode, the ATE, the translation part, and the rotation angle of RPE have been recorded in Tables V–VII. Due to the large scale of the KITTI dataset, the unit of the RPE rotation part adopts degrees (deg) per 100 m, and the unit of the RPE translation part adopts meter (m) per 100 m, which has been applied.

As shown in Tables V–VII, the overall performance of the YOLACT-based SLAM is better than the ORB-

TABLE IV

ATE RMSE (UNIT: m)

Sequence	ORB-SLAM	DynaSLAM	Dynamic-SLAM	ORB-SLAM2				YOLACT based SLAM			
	RMSE	RMSE	RMSE	RMSE	Mean	Median	Std	RMSE	Mean	Median	Std
00	6.68	7.55	4.436	9.579	8.072	6.104	5.157	**3.4366**	3.0914	2.9086	1.5013
02	21.75	26.29	20.367	24.842	19.6	16.887	15.263	**20.1975**	17.5520	14.5830	11.1562
03	1.59	1.81	**0.828**	1.078	0.923	0.84	0.556	1.8232	1.5250	1.3384	0.9992
04	1.79	0.97	1.109	1.233	1.098	1.072	0.559	**0.5634**	0.5045	0.4800	0.2510
05	8.23	4.6	5.724	5.73	5.308	5.456	2.158	**4.3669**	4.1266	4.3989	1.4285
06	14.68	14.74	12.455	15.241	13.172	13.884	7.687	**12.4107**	11.0268	12.2324	5.8461
07	3.36	2.36	1.823	1.962	1.675	1.425	1.022	**1.7430**	1.5505	1.3700	0.8432
08	46.58	40.28	**27.487**	30.29	21.72	14.63	21.11	42.9650	31.7627	17.2859	28.9331
09	7.62	**3.32**	9.285	10.374	9.473	10.395	4.229	9.8425	8.9087	10.3374	4.1844
10	8.68	**6.78**	8.909	9.5151	8.157	6.879	4.899	9.0594	7.5875	5.5511	6.6046

TABLE V

RPE W.R.T. TRANSLATION PART (m) FOR DELTA = 100.0 (m)

Sequence	ORB-SLAM2			YOLACT based SLAM			Improvement
	RMSE	Mean	Std.	RMSE	Mean	Std.	
00	1.116214036	1.009965739	0.475292521	**1.083633568**	0.984576054	0.452627556	2.92%
02	1.126546197	1.052389781	0.401972741	**1.104486106**	1.03036859	0.397781505	1.96%
03	0.927987705	0.890995869	0.259398421	**0.916369562**	0.879714733	0.256583639	1.25%
04	0.510252763	0.496726259	0.116708636	**0.394330949**	0.39405061	0.014866536	22.72%
05	0.636059757	0.61482536	0.162977886	**0.635704848**	0.615142529	0.160375568	0.06%
06	0.762649083	0.718536536	0.25561469	**0.72253913**	0.699121337	0.182461371	5.26%
07	0.579359585	0.559499924	0.150390704	**0.565427798**	0.550443186	0.129309297	2.40%
08	**1.313039786**	1.149518138	0.634571927	1.337771458	1.177969017	0.63405163	-1.88%
09	0.960717741	0.866632721	0.41463997	**0.914582038**	0.846934678	0.345198429	4.80%
10	0.876241916	0.832793644	0.272497049	**0.870783161**	0.828641384	0.267613098	0.62%

TABLE VI

RPE W.R.T. ROTATION ANGLE IN DEGREES (DEG) FOR DELTA = 100.0 (m)

Sequence	ORB-SLAM2			YOLACT based SLAM			Improvement
	RMSE	Mean	Std.	RMSE	Mean	Std.	
00	0.704252	0.601303	0.366613	**0.679778**	0.585505	0.345372	3.48%
02	**0.509458**	0.444547	0.248848	0.513421	0.442288	0.260736	-0.78%
03	0.406495	0.341155	0.221023	**0.332309**	0.29628	0.15049	18.25%
04	0.231235	0.225244	0.052294	**0.222127**	0.218121	0.041994	3.94%
05	0.353356	0.304941	0.178526	**0.350945**	0.307698	0.168773	0.68%
06	0.329738	0.306005	0.122834	**0.246756**	0.216921	0.117617	25.17%
07	0.375483	0.358337	0.112169	**0.372887**	0.355615	0.112171	0.69%
08	0.691854	0.587397	0.365549	**0.653951**	0.54402	0.362896	5.48%
09	0.536663	0.497475	0.20131	**0.516428**	0.431032	0.284447	3.77%
10	0.466912	0.425482	0.19228	**0.422709**	0.372861	0.199143	9.47%

TABLE VII

ATE RMSE (ATE) FOR ORB-SLAM2 AND YOLACT-BASED SLAM (UNIT: m)

Sequence	ORB-SLAM2			YOLACT based SLAM			Improvement
	RMSE	Mean	Std.	RMSE	Mean	Std.	
00	1.278500729	1.140867411	0.577049101	**1.278026448**	1.14440523	0.568936088	0.04%
02	**5.623904973**	4.694145816	3.097305636	5.764694065	4.745755139	3.272538131	-2.50%
03	0.86916426	0.731171335	0.469930835	**0.750049098**	0.633002679	0.402344699	13.70%
04	0.220273415	0.195445462	0.101594532	**0.14927791**	0.136283248	0.060916094	32.23%
05	0.798360021	0.713710476	0.357765397	**0.74637621**	0.67436324	0.319861952	6.51%
06	0.963431985	0.933330686	0.238945643	**0.802975859**	0.783961059	0.173710358	16.65%
07	0.549587316	0.516482002	0.187863141	**0.479924029**	0.449732812	0.167533495	12.68%
08	**3.456896398**	3.109234493	1.510891649	3.509694593	3.207317387	1.425156591	-1.53%
09	3.601034662	2.992111204	2.003676915	**1.527670041**	1.370454689	0.675003481	57.58%
10	1.184657887	1.080961783	0.484701901	**0.87053932**	0.784159376	0.378064521	26.52%

SLAM2 system in the stereo mode. The maximum improvements of the YOLACT-based SLAM in RPE w.r.t. translation part, RPE w.r.t. rotation part, and ATE are 22.72%, 25.17%, and 57.58%, respectively. Sequence 04 contains more moving dynamic objects. The performance improvement of the YOLACT-based SLAM system on sequence 04 is

TABLE VIII
TIME ANALYSIS OF THE SLAM SYSTEMS

Parts of the SLAM system		YOLACT based SLAM	DynaSLAM	DOS-SLAM
Slightly Tracking		**1.6ms**	1.64ms	-
Multi-View Geometry (RGB-D)		**25.2ms**	284.83ms	26.3ms
Background Inpainting (RGB-D)		**210ms**	213ms	-
Dense Optical Flow	with CPU	195ms	-	-
	with GPU	4.3ms	-	-
Instance Segmentation Network (YOLACT with GPU)		**32ms**	205ms	36.77ms

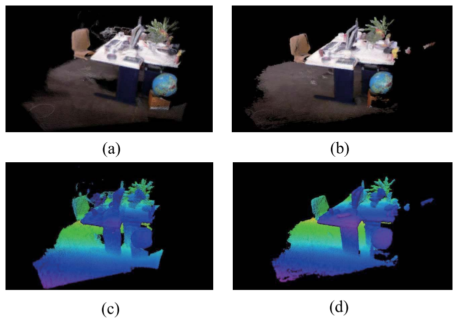
Fig. 9. Point cloud map and the octree map. (a) and (c) Point cloud map and the octree map reconstructed from keyframes, respectively, without updated keyframes and filters. (b) and (d) Point cloud map and the octree map reconstructed from updated keyframes, with statistical filters and voxel filters, which have neater boundaries and less noise.

the most obvious, which are 22.72%, 3.94%, and 32.23%, respectively.

C. Time Analysis

The YOLACT-based SLAM is expected to be a real-time or near real-time SLAM. The average calculation time of the key parts of the system on the TUM dataset has been recorded in Table VIII. The segmentation speed of YOLACT is about 30 FPS with GPU. YOLACT-based SLAM not only expands various dynamic detection methods for different applications but also has an obvious advantage in operation time.

The Background Inpainting cannot be real time. Background inpainting is only used for visualization in our article; it is an independent and optional module. In real-time applications that must be repaired, the display refresh rate needs to be reduced to 4–5 Hz.

Since the instance segmentation network and the multiview geometry are close to real time, the multithreading technology can perform two computing tasks at the same time to ensure real-time or near real-time performance.

D. Point Cloud Map and OctoMap

As shown in Fig. 9, the static feature points are introduced to construct the 3-D point cloud map, and the statistical filters and voxel filters are applied to remove noise in the point cloud. The point cloud map has been converted into the octree map, which facilitates further navigation and path planning.

VI. CONCLUSION

A detection framework equipped with visual-based measurement methods has been proposed in this article in order to accurately detect dynamic objects in DSs. First, in order to filter out dynamic feature points, the frames have been segmented by the YOLACT so that the feature points belonging to the *a priori* dynamic objects will be dropped. The geometric constraints have been introduced to further filter the missing dynamic points. Second, to supplement the dynamic objects missed by YOLACT, the optical flow has been introduced to detect dynamic objects driven by humans. The *a priori* mask is used to calculate the segmentation threshold of optical flow. Third, to serve visualization applications, background inpainting has been expanded to restore the features occluded by dynamic objects. In order to verify the effectiveness of the dynamic detection method, the proposed method has been embedded in the visual SLAM system to improve its performance in dynamic environments. The experimental results have proved that the YOLACT-based SLAM has good performance in DSs.

In future research, an accurate and fast segmentation network is the basis of dynamic object detection; simplifying software or increasing hardware computing power is conducive to the operation of the detect method.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] S. Shirmohammadi and A. Ferrero, "Camera as the instrument: The rising trend of vision based measurement," *IEEE Instrum. Meas. Mag.*, vol. 17, no. 3, pp. 41–47, Jun. 2014.
[2] H. Deng, Q. Fu, Q. Quan, K. Yang, and K.-Y. Cai, "Indoor multi-camera-based testbed for 3-D tracking and control of UAVs," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 6, pp. 3139–3156, Jun. 2020.
[3] L. Yang, J. Fan, Y. Liu, E. Li, J. Peng, and Z. Liang, "Automatic detection and location of weld beads with deep convolutional neural networks," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021.
[4] W. Xie, P. X. Liu, and M. Zheng, "Moving object segmentation and detection for robust RGBD-SLAM in dynamic environments," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–8, 2021.
[5] X. Ban, H. Wang, T. Chen, Y. Wang, and Y. Xiao, "Monocular visual odometry based on depth and optical flow using deep learning," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, Art. no. 2501619.
[6] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.
[7] J. Engel, T. Schps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," *Proc. 13th Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2014, pp. 834–849.
[8] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, Feb. 2014.
[9] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual SLAM and structure from motion in dynamic environments: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 1–36, Jun. 2018.

[10] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nov. 2007, pp. 225–234.

[11] F. Chhaya, D. Reddy, S. Upadhyay, V. Chari, M. Z. Zia, and K. M. Krishna, "Monocular reconstruction of vehicles: Combining SLAM with shape priors," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Stockholm, Sweden, May 2016, pp. 5758–5765.

[12] D. Gutierrez-Gomez, W. Mayol-Cuevas, and J. J. Guerrero, "Inverse depth for accurate photometric and geometric error minimisation in RGB-D dense visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 83–89.

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

[14] A. Kundu, K. M. Krishna, and J. Sivaswamy, "Moving object detection by multi-view geometric techniques from a single camera mounted robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, St. Louis, MO, USA, Oct. 2009, pp. 4306–4312.

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

[16] J. Ji, S. Li, J. Xiong, P. Chen, and Q. Miao, "Semantic image segmentation with propagating deep aggregation," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 12, pp. 9732–9742, Dec. 2020.

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

[18] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robot. Auto. Syst.*, vol. 117, pp. 1–16, Jul. 2019.

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

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

[21] Z. Zhang, J. Zhang, and Q. Tang, "Mask R-CNN based semantic RGB-D SLAM for dynamic scenes," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Hong Kong, Jul. 2019, pp. 1151–1156.

[22] L. Zhao, Z. Liu, J. Chen, W. Cai, W. Wang, and L. Zeng, "A compatible framework for RGB-D SLAM in dynamic scenes," *IEEE Access*, vol. 7, pp. 75604–75614, 2019.

[23] W. Zhang, Q. Chen, W. Zhang, and X. He, "Long-range terrain perception using convolutional neural networks," *Neurocomputing*, vol. 275, pp. 781–787, Jan. 2018.

[24] Y. Ai, T. Rui, M. Lu, L. Fu, S. Liu, and S. Wang, "DDL-SLAM: A robust RGB-D SLAM in dynamic environments combined with deep learning," *IEEE Access*, vol. 8, pp. 162335–162342, 2020.

[25] C. Yu *et al.*, "DS-SLAM: A semantic visual SLAM towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 1168–1174.

[26] S. Han and Z. Xi, "Dynamic scene semantics SLAM based on semantic segmentation," *IEEE Access*, vol. 8, pp. 43563–43570, 2020.

[27] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 9156–9165.

[28] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT++: Better real-time instance segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Aug. 5, 2020, doi: 10.1109/TPAMI.2020.3014297.

[29] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 740–755.

[30] Z. Pan, Y. Jin, X. Jiang, and J. Wu, "An FPGA-optimized architecture of real-time farneback optical flow," in *Proc. IEEE 28th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, May 2020, p. 223.

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

[32] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[33] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast odometry and scene flow from RGB-D cameras based on geometric clustering," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3992–3999.

[34] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Proc. 11th Conf. Robot.-Sci. Syst.*, 2015, pp. 13–17.

[35] M. Runz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 4471–4478.

[36] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background reconstruction for dense RGB-D SLAM in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 3849–3856

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

[38] Y. Wang and S. Huang, "Motion segmentation based robust RGB-D SLAM," in *Proc. 11th World Congr. Intell. Control Autom.*, Jun. 2014, pp. 3122–3127.

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

[40] H. Xu, S. Zhang, and P. Liu, "DOS-SLAM: A real-time dynamic object segmentation visual SLAM system," in *Proc. 2nd Int. Conf. Algorithms, Comput. Artif. Intell.*, 2019, pp. 85–90.

[41] L. Cui and C. Ma, "SDF-SLAM: Semantic depth filter SLAM for dynamic environments," *IEEE Access*, vol. 8, pp. 95301–95311, 2020.

[42] L. Cui and C. Ma, "SOF-SLAM: A semantic visual SLAM for dynamic environments," *IEEE Access*, vol. 7, pp. 166528–166539, 2019.

[43] S. Yang, G. Fan, L. Bai, R. Li, and D. Li, "MGC-VSLAM: A meshing-based and geometric constraint VSLAM for dynamic indoor environments," *IEEE Access*, vol. 8, pp. 81007–81021, 2020.

Jianfang Chang received the master's degree in control theory and control application from Tianjin University, Tianjin, China, in 2018, where he is currently pursuing the Ph.D. degree with the School of Electrical and Information Engineering.

His current research areas encompass neural networks, deep learning, robot operating systems, and simultaneous localization and mapping.

Na Dong received the Ph.D. degree in control theory and control application from Nankai University, Tianjin, China, in 2011.

She is currently an Associate Professor with the School of Electrical and Information Engineering, Tianjin University, Tianjin. Her current research areas encompass intelligent control algorithms, heuristic optimization algorithms, neural networks, data-driven control, deep learning, and image processing.

Donghui Li is currently a Professor with the School of Electrical and Information Engineering, Tianjin University, Tianjin, China.

His current research areas encompass intelligent control algorithms, power electronic transformation, and fault diagnosis.