DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes

Berta Bescos, José M. Fácil, Javier Civera and José Neira

Abstract—The assumption of scene rigidity is typical in SLAM algorithms. Such a strong assumption limits the use of most visual SLAM systems in populated real-world environments, which are the target of several relevant applications like service robotics or autonomous vehicles. In this paper we present DynaSLAM, a visual SLAM system that, building on ORB-SLAM2 [1], adds the capabilities of dynamic object detection and background inpainting. DynaSLAM is robust in dynamic scenarios for monocular, stereo and RGB-D configurations. We are capable of detecting the moving objects either by multi-view geometry, deep learning or both. Having a static map of the scene allows inpainting the frame background that has been occluded by such dynamic objects. We evaluate our system in public monocular, stereo and RGB-D datasets. We study the impact of several accuracy/speed trade-offs to assess the limits of the proposed methodology. DynaSLAM outperforms the accuracy of standard visual SLAM baselines in highly dynamic scenarios. And it also estimates a map of the static parts of the scene, which is a must for long-term applications in real-world environments.

Index Terms—SLAM, Visual-Based Navigation, Localization
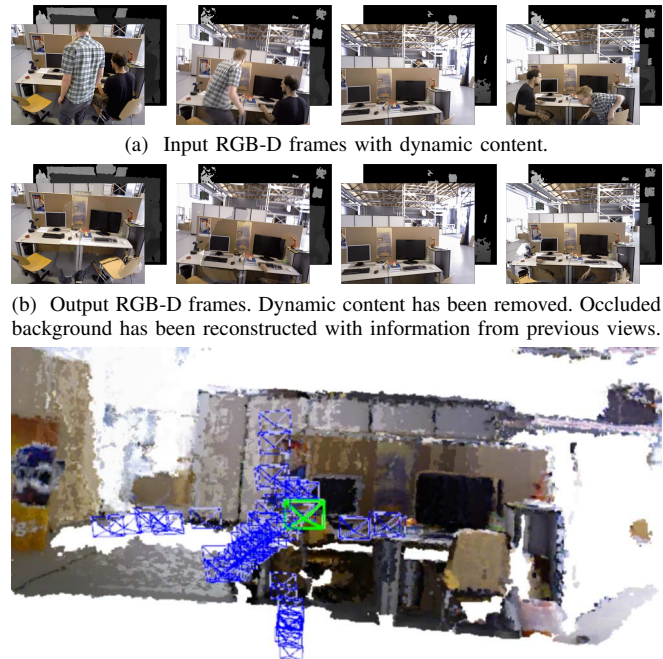
I. INTRODUCTION

SIMULTANEOUS Localization and Mapping (SLAM) is a prerequisite for many robotic applications, for example collision-less navigation. SLAM techniques estimate jointly a map of an unknown environment and the robot pose within such map, only from the data streams of its on-board sensors. The map allows the robot to continually localize within the same environment without accumulating drift. This is in contrast to odometry approaches that integrate the incremental motion estimated within a local window and are unable to correct the drift when revisiting places.

Visual SLAM, where the main sensor is a camera, has received a high degree of attention and research efforts over the last years. The minimalistic solution of a monocular camera has practical advantages with respect to size, power and cost, but also several challenges such as the unobservability of the

Berta Bescos, José M. Fácil, Javier Civera and José Neira are with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza 50018, Spain {bbescos,jmfacil,jcivera,jneira}@unizar.es

(a) Input RGB-D frames with dynamic content.

(b) Output RGB-D frames. Dynamic content has been removed. Occluded background has been reconstructed with information from previous views.

(c) Map of the static part of the scene, after removal of the dynamic objects.

Fig. 1: Overview of DynaSLAM results for the RGB-D case.

scale or state initialization. By using more complex setups, like stereo or RGB-D cameras, these issues are solved and the robustness of visual SLAM systems can be greatly improved.

The research community has addressed SLAM from many different angles. However, the vast majority of the approaches and datasets assume a static environment. As a consequence, they can only manage small fractions of dynamic content by classifying them as outliers to such static model. Although the static assumption holds for some robotic applications, it limits the applicability of visual SLAM in many relevant cases, such as intelligent autonomous systems operating in populated real-world environments over long periods of time.

Visual SLAM can be classified into feature-based methods [2], [3], that rely on salient points matching and can only estimate a sparse reconstruction; and direct methods [4], [5], [6], which are able to estimate in principle a completely dense reconstruction by the direct minimization of the photometric error and TV regularization. Some direct methods focus on the high-gradient areas estimating semi-dense maps [7], [8].

None of the above methods, considered the state of the art, address the very common problem of dynamic objects in the

scene, *e.g.*, people walking, bicycles or cars. Detecting and dealing with dynamic objects in visual SLAM reveals several challenges for both mapping and tracking, including:

1) How to detect such dynamic objects in the images to:
   a) Prevent the tracking algorithm from using matches that belong to dynamic objects.
   b) Prevent the mapping algorithm from including moving objects as part of the 3D map.
2) How to complete the part of the 3D map that is temporally occluded by a moving object.

Many applications would greatly benefit from progress along these lines. Among others, augmented reality, autonomous vehicles, and medical imaging. All of them could for instance safely reuse maps from previous runs. Detecting and dealing with dynamic objects is a requisite to estimate stable maps, useful for long-term applications. If the dynamic content is not detected, it becomes part of the 3D map, complicating its usability for tracking or relocation purposes.

In this work we propose an on-line algorithm to deal with dynamic objects in RGB-D, stereo and monocular SLAM. This is done by adding a front-end stage to the state-of-the-art ORB-SLAM2 system [1], with the purpose of having a more accurate tracking and a reusable map of the scene. In the monocular and stereo cases our proposal is to use a CNN to pixel-wise segment the *a priori* dynamic objects in the frames (*e.g.*, people and cars), so that the SLAM algorithm does not extract features on them. In the RGB-D case we propose to combine multi-view geometry models and deep-learning-based algorithms for detecting dynamic objects and, after having removed them from the images, inpaint the occluded background with the correct information of the scene (Fig. 1).

The rest of the paper is structured as follows: section II discusses related work, section III gives the details of our proposal, section IV details the experimental results, and section V presents the conclusions and lines for future work.

II. RELATED WORK

Dynamic objects are, in most SLAM systems, classified as spurious data and therefore neither included in the map nor used for camera tracking. The most typical outlier rejection algorithms are RANSAC (*e.g.*, in ORB-SLAM [3], [1]) and robust cost functions (*e.g.*, in PTAM [2]).

There are several SLAM systems that address more specifically the dynamic scene content. Within feature-based SLAM methods, some of the most relevant are:

- Tan *et al.* [9] detect changes that take place in the scene by projecting the map features into the current frame for appearance and structure validation.
- Wangsiripitak and Murray [10] track known 3D dynamic objects in the scene. Similarly, Riazuelo *et al.* [11] deal with human activity by detecting and tracking people.
- More recently, the work of Li and Lee [12] uses depth edges points, which have an associated weight indicating its probability of belonging to a dynamic object.

Direct methods are, in general, more sensitive to dynamic objects in the scene. The most relevant works specifically designed for dynamic scenes are:

- Alcantarilla *et al.* [13] detect moving objects by means of a scene flow representation with stereo cameras.
- Wang and Huang [14] segment the dynamic objects in the scene using RGB optical flow.
- Kim *et al.* [15] propose to obtain the static parts of the scene by computing the difference between consecutive depth images projected over the same plane.
- Sun *et al.* [16] calculate the difference in intensity between consecutive RGB images. Pixel classification is done with the segmentation of the quantized depth image.

All the methods –both feature-based and direct ones– that map the static scene parts only from the information contained in the sequence [1], [3], [9], [12], [13], [14], [15], [16], [17], fail to estimate lifelong models when an *a priori* dynamic object remains static, *e.g.*, parked cars or people sitting. On the other hand, Wangsiripitak and Murray [10], and Riazuelo *et al.* [11] would detect those *a priori* dynamic objects, but would fail to detect changes produced by static objects, *e.g.*, a chair a person is pushing, or a ball that someone has thrown. That is, the former approach succeeds in detecting *moving* objects, and the second one in detecting several *movable* objects. Our proposal, DynaSLAM, combines multi-view geometry and deep learning in order to address both situations. Similarly, Anrus *et al.* [18] segment dynamic objects by combining a dynamic classifier and multi-view geometry.

III. SYSTEM DESCRIPTION

Fig. 2 shows an overview of our system. First of all, the RGB channels pass through a CNN that segments out pixel-wise all the *a priori* dynamic content, *e.g.*, people or vehicles.

In the RGB-D case, we use multi-view geometry to improve the dynamic content segmentation in two ways. First, we refine the segmentation of the dynamic objects previously obtained by the CNN. Second, we label as dynamic new object instances that are static most of the time (*i.e.*, detect *moving* objects that were not set to *movable* in the CNN stage).

For that purpose, it is necessary to know the camera pose, for which a low-cost tracking module has been implemented to localize the camera within the already created scene map. These segmented frames are the ones which are used to obtain the camera trajectory and the map of the scene. Notice that if the moving objects in the scene are not within the CNN classes, the multi-view geometry stage would still detect the dynamic content, but the accuracy might decrease.

Once this full dynamic object detection and localization of the camera have been done, we aim to reconstruct the occluded background of the current frame with static information from previous views. These synthetic frames are relevant for applications like augmented and virtual reality, and place recognition in lifelong mapping.
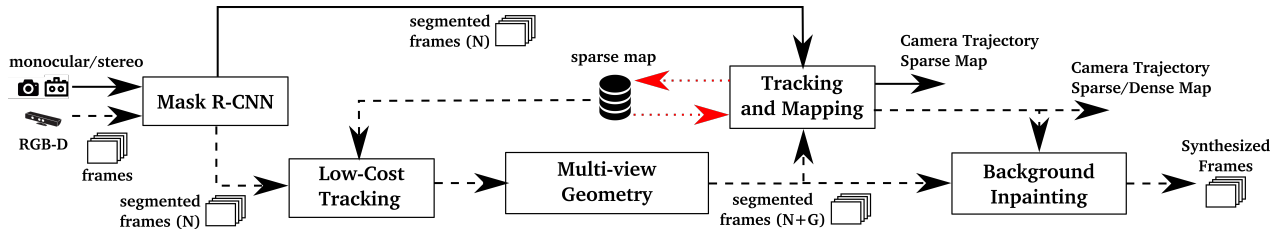
Fig. 2: Block diagram of our proposal. In the stereo and monocular pipeline (black continuous line) the images pass through a Convolutional Neural Network (Mask R-CNN) for computing the pixel-wise semantic segmentation of the *a priori* dynamic objects before being used for the mapping and tracking. In the RGB-D case (black dashed line) a second approach based on multi-view geometry is added for a more accurate motion segmentation, for which we need a low-cost tracking algorithm. Once the position of the camera is known (Tracking and Mapping output), we can inpaint the background occluded by dynamic objects. The red dotted line represents the data flow of the stored sparse map.

In the monocular and stereo cases, the images are segmented by the CNN so that keypoints belonging to the *a priori* dynamic objects are neither tracked nor mapped.

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

A. Segmentation of Potentially Dynamic Content using a CNN

For detecting dynamic objects we propose to use a CNN that obtains a pixel-wise semantic segmentation of the images. In our experiments we use Mask R-CNN [19], which is the state of the art for object instance segmentation. Mask R-CNN can obtain both pixel-wise semantic segmentation and the instance labels. For this work we use the pixel-wise semantic segmentation information, but the instance labels could be useful in future work for the tracking of the different moving objects. We use the TensorFlow implementation by Matterport[1].

The input of Mask R-CNN is the RGB original image. The idea is to segment those classes that are potentially dynamic or movable (person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra and giraffe). We consider that, for most environments, the dynamic objects likely to appear are included within this list. If other classes were needed, the network, trained on MS COCO [20], could be fine-tuned with new training data.

The output of the network, assuming that the input is an RGB image of size $m \times n \times 3$, is a matrix of size $m \times n \times l$, where l is the number of objects in the image. For each output channel $i \in l$ a binary mask is obtained. By combining all the channels into one, we can obtain the segmentation of all dynamic objects appearing in one image of the scene.

B. Low-Cost Tracking

After the potentially dynamic content has been segmented, the pose of the camera is tracked using the static part of the image. Because the segment contours usually become high-gradient areas, salient point features tend to appear. We do not consider the features in such contour areas.

[1]https://github.com/matterport/Mask_RCNN

The tracking implemented at this stage of the algorithm is a simpler and therefore computationally lighter version of the one in ORB-SLAM2 [1]. It projects the map features in the image frame, searches for the correspondences in the static areas of the image, and minimizes the reprojection error to optimize the camera pose.

C. Segmentation of Dynamic Content using Mask R-CNN and Multi-view Geometry

By using Mask R-CNN, most of the dynamic objects can be segmented and not used for tracking and mapping. However, there are objects that cannot be detected by this approach because they are not *a priori* dynamic, but movable. Examples of the latest are a book carried by someone, a chair that someone is moving, or even furniture changes in long-term mapping. The approach utilized for dealing with these cases is detailed in this section.

For each input frame, we select the previous keyframes that have the highest overlaps. This is done by taking into account

(a) Keypoint x' belongs to a static object ($z' = z_{proj}$).

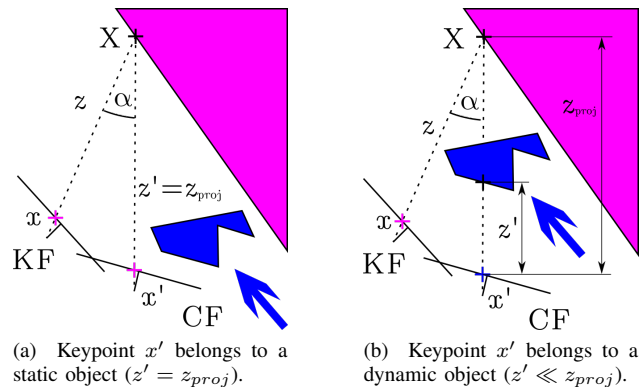(b) Keypoint x' belongs to a dynamic object ($z' \ll z_{proj}$).

Fig. 3: Keypoint x from the Key Frame (KF) is projected into the Current Frame (CF) using its depth and camera pose, resulting in point x' with depth z'. The projected depth z_{proj} is then computed. A pixel is labeled as dynamic if the difference $\Delta z = z_{proj} - z'$ is greater than a threshold τ_z.

(a) Using Multi-view Geometry. (b) Using Deep Learning. (c) Using Geometry and Deep Learning.
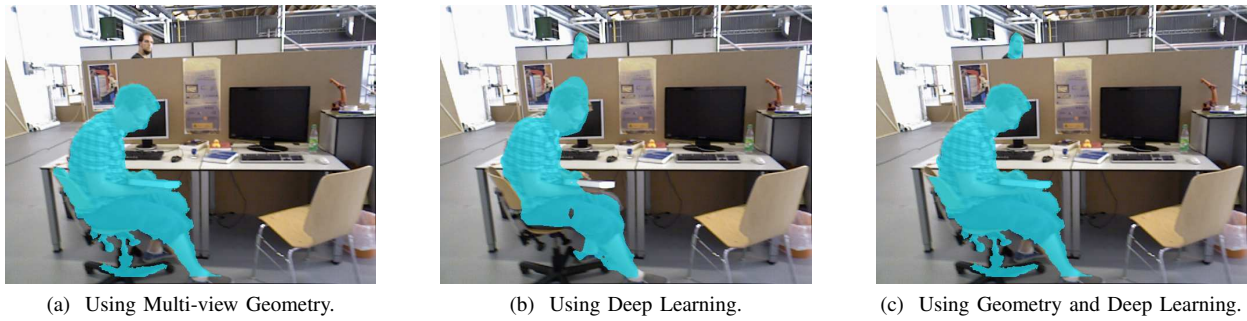
Fig. 4: Detection and segmentation of dynamic objects using multi-view geometry (left), deep learning (middle), and a combination of both geometric and learning methods (right). Notice that Fig. 4a cannot detect the person behind the desk, Fig. 4b cannot segment the book carried by the person, and the combination of the two (Fig. 4c) is the best performing.

both the distance and the rotation between the new frame and each of the keyframes, similarly to Tan *et al.* [9]. The number of overlapping keyframes has been set to 5 in our experiments, as a compromise between computational cost and accuracy in the detection of dynamic objects.

We then compute the projection of each keypoint x from the previous keyframes into the current frame, obtaining the keypoints x', as well as their projected depth z_{proj}, computed from the camera motion. Notice that the keypoints x come from the features extractor algorithm used in ORB-SLAM2. For each keypoint, whose corresponding 3D point is X, we calculate the angle between the back-projections of x and x', *i.e.*, their parallax angle α. If this angle is greater than $30°$, the point might be occluded, and will be ignored from then on. We observed that, in the TUM dataset, for parallax angles greater than $30°$ static objects were considered as dynamic due to their viewpoint difference. We obtain the depth of the remaining keypoints in the current frame z' (directly from the depth measurement), taking into account the reprojection error, and we compare them with z_{proj}. If the difference $\Delta z = z_{proj} - z'$ is over a threshold τ_z, keypoint x' is considered to belong to a dynamic object. This idea is shown in Fig. 3. To set the threshold τ_z, we manually tagged the dynamic objects of 30 images within the TUM dataset, and evaluated both the precision and recall of our method for different thresholds τ_z. By maximizing the expression $0.7 \times Precision + 0.3 \times Recall$, we concluded that $\tau_z = 0.4m$ is a reasonable choice.

Some of the keypoints labeled as dynamic lay on the borders of moving objects, and might cause problems. To avoid this, we use the information given by the depth images. If a keypoint is set as dynamic, but a patch around itself in the depth map has high variance, we change the label to static.

So far, we know which keypoints belong to dynamic objects, and which ones do not. To classify all the pixels belonging to dynamic objects, we grow the region in the depth image around the dynamic pixels [21]. An example of a RGB frame and its corresponding dynamic mask can be seen in Fig. 4a.

The results of the CNN (Fig. 4b) can be combined with those of this geometric method for full dynamic object detec-tion (Fig. 4c). We can find strengths and limitations in both methods, hence the motivation for their combined use. For geometric approaches, the main problem is that initialization is not trivial because of its multi-view nature. Learning methods and their impressive performance using a single view, do not have such initialization problems. Their main limitation though is that objects that are supposed to be static can be moved, and the method is not able to identify them. This last case can be solved using multi-view consistency tests.

These two ways of facing the moving objects detection problem are illustrated in Fig. 4. In Fig. 4a we see that the person in the back, which is potentially a dynamic object, is not detected. There are two reasons for this. First, the difficulties that RGB-D cameras face when measuring the depth of distant objects. And second, the fact that reliable features lie on defined, and therefore nearby, parts of the image. Albeit, this person is detected by the deep learning method (Fig. 4b). Apart from this, on one hand we see in the Fig. 4a that not only is detected the person in the front of the image, but also the book he is holding and the chair he is sitting on. On the other hand, in the Fig. 4b the two people are the only objects detected as dynamic, and also their segmentation is less accurate. If only the deep learning method is used, a *floating book* would be left in the images and would incorrectly become part of the 3D map.

Because of the advantages and disadvantages of both meth-ods, we consider that they are complementary and therefore their combined use is an effective way of achieving accurate tracking and mapping. In order to achieve this goal, if an object has been detected with both approaches, the segmen-tation mask should be that of the geometrical method. If an object has only been detected by the learning based method, the segmentation mask should contain this information too. The final segmented image of the example in the previous paragraph can be seen in the Fig. 4c. The segmented dynamic parts are removed from the current frame and from the map.

D. Tracking and Mapping

The input to this stage of the system contains the RGB and depth images, as well as their segmentation mask. We extract ORB features in the image segments classified as static. As the segment contours are high-gradient areas, the keypoints falling in this intersection have to be removed.

E. Background Inpainting

For every removed dynamic object, we aim at inpainting the occluded background with static information from previous views, so that we can synthesize a realistic image without moving content. We believe that such synthetic frames, containing the static structure of the environment, are useful for applications such as virtual and augmented reality, and for relocation and camera tracking after the map is created.

Since we know the position of the previous and current frames, we project into the dynamic segments of the current frame the RGB and depth channels from a set of all the previous keyframes (the last 20 in our experiments). Some gaps have no correspondences and are left blank: some areas cannot be inpainted because their correspondent part of the scene has not appeared so far in the keyframes, or, if it has appeared, it has no valid depth information. These gaps cannot be reconstructed with geometrical methods and would need a more elaborate inpainting technique. Fig. 5 shows the resulting synthetic images for three input frames from different sequences of the TUM benchmark. Notice how the dynamic content has been successfully segmented and removed. Also, most of the segmented parts have been properly inpainted with information from the static background.

Another application of these synthesized frames would be the following: if the frames dynamic areas are inpainted with the static content, the system can work as a SLAM system under the staticity assumption using the inpainted images.

IV. EXPERIMENTAL RESULTS

We have evaluated our system in the public datasets TUM RGB-D and KITTI and compared to other state-of-the-art SLAM systems in dynamic environments, using when possible results published in the original papers. Furthermore we have compared our system against the original ORB-SLAM2 to quantify the improvement of our approach in dynamic scenes. In this case, the results for some sequences were not published and we have ourselves completed their evaluation. Mur and Tardós [1] propose to run each sequence five times and show median results, to account for the non-deterministic nature of the system. We have run each sequence ten times, as dynamic objects are prone to increase this non-deterministic effect.

A. TUM Dataset

The TUM RGB-D dataset [22] is composed of 39 sequences recorded with a Microsoft Kinect sensor in different indoor scenes at full frame rate (30Hz). Both the RGB and the depth

Sequence	DynaSLAM (N)	DynaSLAM (G)	DynaSLAM (N+G)	DynaSLAM (N+G+BI)
$w\_halfsphere$	**0.025**	0.035	**0.025**	0.029
w_xyz	**0.015**	0.312	**0.015**	**0.015**
w_rpy	0.040	0.251	**0.035**	0.136
w_static	0.009	0.009	**0.006**	0.007
$s\_halfsphere$	**0.017**	0.018	**0.017**	0.025
s_xyz	0.014	**0.009**	0.015	0.013

TABLE I: Absolute trajectory RMSE [m] for several variants of DynaSLAM (RGB-D).

images are available, together with the ground-truth trajectory, the latest recorded by a high-accuracy motion-capture system. In the sequences named *sitting* (*s*) there are two people sitting in front of a desk while speaking and gesticulating, *i.e.*, there is a low degree of motion. In the sequences named *walking* (*w*), two people walk both in the background and the foreground and sit down in front of the desk. This dataset is highly dynamic and therefore challenging for standard SLAM systems. For both types of sequences *sitting* (*s*) and *walking* (*w*) there are four types of camera motions: (1) halfsphere (half): the camera moves following the trajectory of a 1-meter diameter half sphere, (2) xyz: the camera moves along the x-y-z axes, (3) rpy: the camera rotates over roll, pitch and yaw axes, and (4) static: the camera is kept static manually.

We use the absolute trajectory RMSE as the error metric for our experiments, as proposed by Sturm *et al.* [22].

The results of different variations of our system for six sequences within this dataset are shown in Table I. Firstly, DynaSLAM (N) is the system in which only Mask R-CNN segments out the *a priori* dynamic objects. Secondly, in DynaSLAM (G) the dynamic objects have been only detected with the multi-view geometry method based on depth changes. Thirdly, DynaSLAM (N+G) stands for the system in which the dynamic objects have been detected combining both the geometrical and deep learning approaches. Finally, we have considered interesting to analyze the system shown in Fig. 6. In this case (N+G+BI), the background inpainting stage (BI) is done before the tracking and mapping. The motivation for this experiment is that, if the dynamic areas are inpainted with the static content, the system can work as a SLAM system under the staticity assumption using the inpainted images. In this proposal, the ORB features extractor algorithm works both in the real and reconstructed areas of the frames, finding matches with the keypoints of the previously processed keyframes.

According to Table I, the system (N+G) that uses learning and geometry is the most accurate one in most sequences. The improvement over (N) comes from the segmentation of *movable* objects and refinement of the dynamic segments. The system (G) has higher error because it needs motion and its segmentation is only accurate after a small delay, during which the dynamic content introduces some error in the estimation.

Adding the background inpainting stage (BI) before the localization of the camera (Fig. 6) usually leads to less accuracy in the tracking. The reason is that the background

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2860039, IEEE Robotics and Automation Letters

6   IEEE ROBOTICS AND AUTOMATION LETTERS. PREPRINT VERSION. ACCEPTED JULY, 2018

(a) RGB original images.

(b) Depth original image.

(c) Inpainted RGB images.
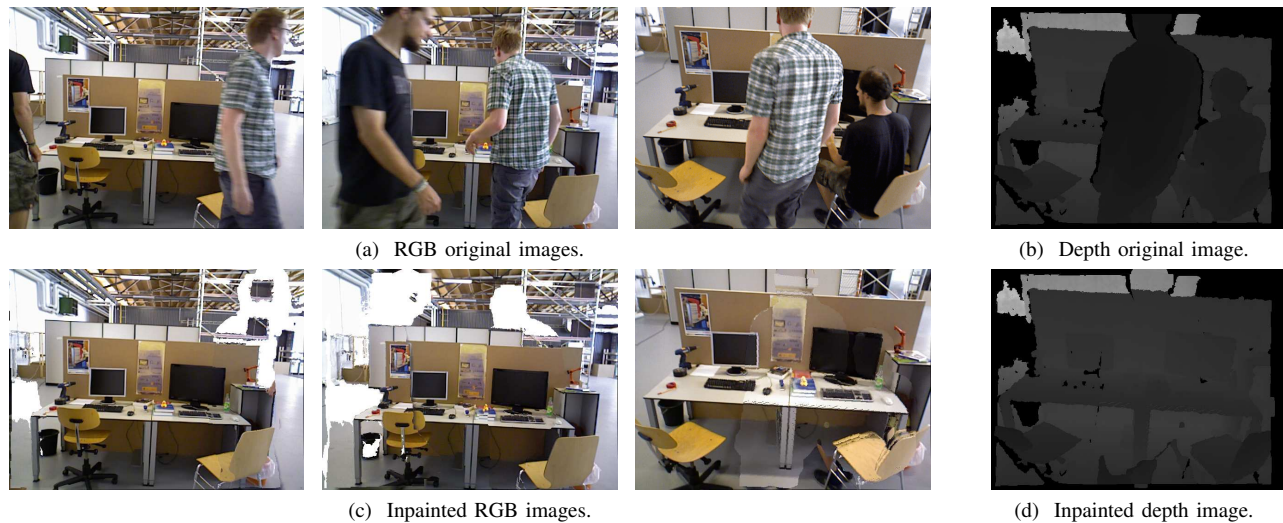
(d) Inpainted depth image.

Fig. 5: Qualitative results of our approach. In Fig. 5a we show three RGB input frames, and in Fig. 5c we show the output of our system, in which all dynamic objects have been detected and the background has been reconstructed. Figs. 5b and 5d show respectively the depth input and output, which has also been processed. Figure best viewed in electronic format.

reconstruction is strongly correlated with the camera poses. Hence, for sequences with purely rotational motion (*rpy*, *halfsphere*), the estimated camera poses have a greater error and lead to a non-accurate background reconstruction. The background inpainting stage (BI) should be done therefore once the tracking stage is finished (Fig. 2). The main accomplishment of the background reconstruction is seen in the synthesis of the static images (Fig. 5) for applications such as virtual reality or cinematography. The DynaSLAM results shown from now on are from the best variant, that is, (N+G).

Table II shows our results on the same sequences, compared against RGB-D ORB-SLAM2. Our method outperforms ORB-SLAM2 in highly dynamic scenarios (*walking*), reaching an error similar to that of the original RGB-D ORB-SLAM2 system in static scenarios. In the case of low-dynamic scenes (*sitting*) the tracking results are slightly worse because the tracked keypoints find themselves further than those belonging to dynamic objects. Albeit, DynaSLAM's map does not contain the dynamic objects that appear along the sequence. Fig. 7 shows an example of the estimated trajectories of DynaSLAM and ORB-SLAM2, compared to the ground-truth.

Table III shows a comparison between our system and several state-of-the-art RGB-D SLAM systems designed for dynamic environments. In account for the effectiveness of our and the state-of-the-art approaches for motion detection

Sequence	ORB-SLAM2 (RGB-D) [1]	DynaSLAM (N+G) (RGB-D)		
	median	median	min	max
$w\_halfsphere$	0.351	**0.025**	0.024	0.031
w_xyz	0.459	**0.015**	0.014	0.016
w_rpy	0.662	**0.035**	0.032	0.038
w_static	0.090	**0.006**	0.006	0.008
$s\_halfsphere$	0.020	**0.017**	0.016	0.020
s_xyz	**0.009**	0.015	0.013	0.015

TABLE II: Comparison of the RMSE of ATE [m] of DynaSLAM against ORB-SLAM2 for RGB-D cameras. To account for the non-deterministic nature of the system, we show the median, minimum and maximum error of ten runs.

(independently of the utilized SLAM system), we also show the respective improvement values against the original SLAM system used in every case. DynaSLAM significantly outper-
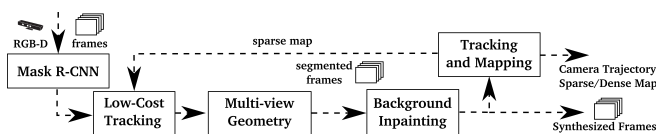

Fig. 7: Ground truth and trajectories estimated by DynaSLAM and ORB-SLAM2 in the TUM sequence $fr3/walking\_xyz$.

Fig. 6: Block diagram of RGB-D DynaSLAM (N+G+BI).

Sequence	Depth Edge SLAM [12]	Motion Segmentation DSLAM [14]			Motion Removal DVO-SLAM [16]			DynaSLAM (N+G) (RGB-D)		
		w/o Motion Detection	w/ Motion Detection	Improvement w/ MD	w/o Motion Detection	w/ Motion Detection	Improvement w/ MD	w/o Motion Detection [1]	w/ Motion Detection	Improvement w/ MD
	[m]	[m]	[m]	[%]	[m]	[m]	[%]	[m]	[m]	[%]
w_half	0.049	0.116	0.055	52.59%	0.529	0.125	76.32%	0.351	**0.025**	**92.88%**
w_xyz	0.060	0.202	0.040	80.20%	0.597	0.093	84.38%	0.459	**0.015**	**96.73%**
w_rpy	0.179	0.515	0.076	85.24%	0.730	0.133	81.75%	0.662	**0.035**	**94.71%**
w_stat	0.026	0.470	0.024	**94.89%**	0.212	0.066	69.06%	0.090	**0.006**	93.33%
s_half	0.043	-	-	-	0.062	0.047	**23.70%**	0.020	**0.017**	15.00%
s_xyz	0.040	-	-	-	0.051	0.048	**4.55%**	0.009	**0.015**	X

TABLE III: Absolute trajectory RMSE [m] of DynaSLAM against state-of-the-art RGB-D SLAM systems in dynamic scenes. To evaluate the effectiveness of the specific module addressing dynamic content, we report the improvement with respect to the original SLAM systems (w/o Motion Detection). Our results are estimated using Mask R-CNN and multi-view geometry.

forms all of them in all sequences (both high and low dynamic ones). The error is, in general, around 1-2 cm, similar to that of the state of the art in static scenes. Our motion detection approach also outperforms the other methods.

ORB-SLAM, the monocular version of ORB-SLAM2, is generally more accurate than the RGB-D one in dynamic scenes, due to their different initialization algorithms. RGB-D ORB-SLAM2 is initialized and starts the tracking from the very first frame, and hence dynamic objects can introduce errors. ORB-SLAM delays the initialization until there is parallax and consensus using the staticity assumption. Hence, it does not track the camera for the full sequence, sometimes missing a substantial part of it, or even not initializing.

Table IV shows the tracking results and percentage of the tracked trajectory for ORB-SLAM and DynaSLAM (monocular) in the TUM dataset. The initialization in DynaSLAM is always quicker than that of ORB-SLAM. In fact, in highly dynamic sequences, ORB-SLAM initialization only occurs when the moving objects disappear from the scene. In conclusion, although the accuracy of DynaSLAM is slightly lower, it succeeds in bootstrapping the system with dynamic content and producing a map without such content (see Fig. 1), to be re-used for long-term applications. The reason why DynaSLAM is slightly less accurate is that the estimated trajectory is longer, and there is therefore room for accumulating errors.

Sequence	ORB-SLAM [1]		DynaSLAM (Monocular)	
	ATE [m]	% Traj	ATE [m]	% Traj
$fr3/walking\_halfsphere$	**0.017**	87.16	0.021	**97.84**
$fr3/walking\_xyz$	**0.012**	57.63	0.014	**87.37**
$fr2/desk\_with\_person$	**0.006**	95.30	0.008	**97.07**
$fr3/sitting\_xyz$	**0.007**	91.44	0.013	**100.00**

TABLE IV: Absolute trajectory RMSE [m] and percentage of successfully tracked trajectory for both ORB-SLAM and DynaSLAM (monocular).

B. KITTI Dataset

The KITTI Dataset [23] contains stereo sequences recorded from a car in urban and highway environments. Table V

shows our results in the eleven training sequences, compared against stereo ORB-SLAM2. We use two different metrics, the absolute trajectory RMSE proposed in [22], and the average relative translation and rotation errors, proposed in [23]. Table VI shows the results in the same sequences for the monocular variants of ORB-SLAM and DynaSLAM.

Sequence	ORB-SLAM2 (Stereo) [1]			DynaSLAM (Stereo)		
	RPE [%]	RRE [°/100m]	ATE [m]	RPE [%]	RRE [°/100m]	ATE [m]
KITTI 00	**0.70**	**0.25**	**1.3**	0.74	0.26	1.4
KITTI 01	**1.39**	**0.21**	10.4	1.57	0.22	**9.4**
KITTI 02	**0.76**	**0.23**	**5.7**	0.80	0.24	6.7
KITTI 03	0.71	0.18	0.6	**0.69**	0.18	0.6
KITTI 04	0.48	0.13	0.2	**0.45**	**0.09**	0.2
KITTI 05	0.40	0.16	0.8	0.40	0.16	0.8
KITTI 06	0.51	**0.15**	0.8	**0.50**	0.17	0.8
KITTI 07	**0.50**	**0.28**	0.5	0.52	0.29	0.5
KITTI 08	1.05	0.32	3.6	1.05	0.32	**3.5**
KITTI 09	**0.87**	**0.27**	3.2	0.93	0.29	**1.6**
KITTI 10	**0.60**	**0.27**	**1.0**	0.67	0.32	1.2

TABLE V: Comparison of the RMSE of the ATE [m], the average of the RPE [%] and the RRE [°/$100m$] of DynaSLAM against ORB-SLAM2 system for stereo cameras.

Sequence	ORB-SLAM [1]	DynaSLAM (Monocular)
KITTI 00	**5.33**	7.55
KITTI 02	**21.28**	26.29
KITTI 03	**1.51**	1.81
KITTI 04	1.62	**0.97**
KITTI 05	4.85	**4.60**
KITTI 06	**12.34**	14.74
KITTI 07	**2.26**	2.36
KITTI 08	46.68	**40.28**
KITTI 09	6.62	**3.32**
KITTI 10	8.80	**6.78**

TABLE VI: Absolute trajectory RMSE [m] for ORB-SLAM and DynaSLAM (monocular).

Note that the results are similar in both the monocular and stereo cases, but the former is more sensitive to dynamic objects and therefore to the additions in DynaSLAM. In some sequences the accuracy of the tracking is improved when not using features belonging to *a priori* dynamic objects,

i.e., cars, bicycles, *etc.* An example of this would be the sequences KITTI 01 and KITTI 04, in which all vehicles that appear are moving. In the sequences in which most of the recorded cars and vehicles are parked (hence static), the absolute trajectory RMSE is usually bigger since the keypoints used for tracking are more distant and usually belong to low-texture areas (KITTI 00, KITTI 02, KITTI 06). However, the loop closure and relocalization algorithms work more robustly since the resulting map only contains structural objects, *i.e.*, the map can be re-used and work in long-term applications.

As future work, it is interesting to make a distinction between those *movable* and *moving* objects, by using only RGB information. If a car is detected by the CNN (*movable*) but is not currently moving, its corresponding keypoints should be used for the local tracking, but should not be in the map.

C. Timing Analysis

To complete the evaluation of our proposal, Table VII shows the average computational time for its different stages. Note that DynaSLAM is not optimized for real-time operation. However, its capability for creating life-long maps of the static scene content are also relevant for running on offline mode.

Sequence	Low-Cost Tracking [ms]	Multi-view Geometry [ms]	Background Inpainting [ms]
w_halfsphere	1.69	333.68	208.09
w_rpy	1.59	235.98	183.56

TABLE VII: DynaSLAM average computational time [ms].

Mur *et al.* show real-time results for and ORB-SLAM2 [1]. He *et al.* [19] report that Mask R-CNN runs at 195 ms per image on a Nvidia Tesla M40 GPU.

The addition of the multi-view geometry stage is an additional slowdown, due mainly to the region growth algorithm. The background inpainting also introduces a delay, which is another reason why it should be done after the tracking and mapping stage, as it has been shown in Fig. 2.

V. CONCLUSIONS

We have presented a visual SLAM system that, building on ORB-SLAM, adds a motion segmentation approach that makes it robust in dynamic environments for monocular, stereo and RGB-D cameras. Our system accurately tracks the camera and creates a static and therefore reusable map of the scene. In the RGB-D case, DynaSLAM is capable of obtaining the synthetic RGB frames with no dynamic content and with the occluded background inpainted, as well as their corresponding synthesized depth frames, which might be together very useful for virtual reality applications. We include a video showing the potential of DynaSLAM [2].

The comparison against the state of the art shows that DynaSLAM achieves in most cases the highest accuracy.

[2]https://youtu.be/EabI_goFmQs

In the TUM Dynamic Objects dataset, DynaSLAM is currently the best RGB-D SLAM solution. In the monocular case, our accuracy is similar to that of ORB-SLAM, obtaining however a static map of the scene with an earlier initialization.

In the KITTI dataset DynaSLAM is slightly less accurate than monocular and stereo ORB-SLAM, except for those cases in which dynamic objects represent an important part of the scene. However, our estimated map only contains structural objects and can therefore be re-used in long-term applications.

Future extensions of this work might include, among others, real-time performance, an RGB-based motion detector, or a more realistic appearance of the synthesized RGB frames by using a more elaborate inpainting technique, *e.g.*, the one used by Pathak *et al.* [24] by the use of GANs.

REFERENCES

[1] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE T-RO*, 2017.
[2] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *ISMAR*, pp. 225–234, 2007.
[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE T-RO*, 2015.
[4] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Joint Pattern Recognition Symposium*, 2010.
[5] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *ICCV*, IEEE, 2011.
[6] G. Graber, T. Pock, and H. Bischof, "Online 3D reconstruction using convex optimization," in *ICCV Workshops*, IEEE, 2011.
[7] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *ECCV*, pp. 834–849, Springer, 2014.
[8] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
[9] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular SLAM in dynamic environments," in *ISMAR*, pp. 209–218, 2013.
[10] S. Wangsiripitak and D. W. Murray, "Avoiding moving outliers in visual SLAM by tracking moving objects," in *ICRA*, pp. 375–380, IEEE, 2009.
[11] L. Riazuelo, L. Montano, and J. M. M. Montiel, "Semantic visual SLAM in populated environments," *ECMR*, 2017.
[12] S. Li and D. Lee, "RGB-D SLAM in Dynamic Environments Using Static Point Weighting," *IEEE RA-L*, vol. 2, no. 4, pp. 2263–2270, 2017.
[13] P. F. Alcantarilla, J. J. Yebes, J. Almazán, and L. M. Bergasa, "On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments," in *ICRA*, 2012.
[14] Y. Wang and S. Huang, "Motion segmentation based robust RGB-D SLAM," in *WCICA*, pp. 3122–3127, IEEE, 2014.
[15] D.-H. Kim and J.-H. Kim, "Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment," *IEEE T-RO*, 2016.
[16] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *RAS*, 2017.
[17] A. Concha and J. Civera, "DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence," in *IEEE/RSJ IROS*, 2015.
[18] R. Ambrus, J. Folkesson, and P. Jensfelt, "Unsupervised object segmentation through change detection in a long term autonomy scenario," in *Humanoid Robots (Humanoids)*, IEEE, 2016.
[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *arXiv preprint arXiv:1703.06870*, 2017.
[20] 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 *ECCV*, 2014.
[21] N. L. Gerlach, G. J. Meijer, D.-J. Kroon, E. M. Bronkhorst, S. J. Bergé, and T. J. J. Maal, "Evaluation of the potential of automatic segmentation of the mandibular canal," *BJOMS*, 2014.
[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IROS*, 2012.
[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *IJRR*, vol. 32, no. 11, pp. 1231–1237, 2013.
[24] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016.