Detecting, Tracking and Eliminating Dynamic Objects in 3D Mapping using Deep Learning and Inpainting

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

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 of its onboard 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, like size, power and cost; but also several challenges like the unobservability of the scale or the state initialization. By using more involved setups, like stereo or RGB-D cameras, these issues are solved and the robustness of the system can be greatly improved.

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

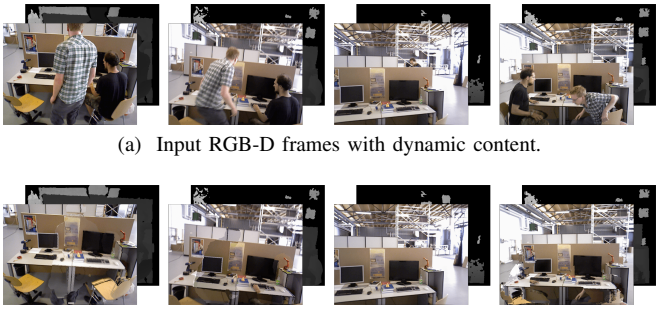
Visual SLAM can be classified into feature-based methods [1], [2], that rely on salient points matching and can only estimate a sparse reconstruction; and direct methods [3], [4], [5], which are able to estimate in principle a completely dense reconstruction by the direct minimization of the photometric error and TV regularization. Some direct methods focus on the high-gradient areas estimating semi-dense maps [6], [7].

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, cars, *etc*. 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.
2) How to prevent the tracking algorithm from using information belonging to dynamic objects.
3) How to prevent the mapping algorithm from including moving objects as part of the 3D map.
4) How to complete the 3D information in the scene occluded by the moving objects.

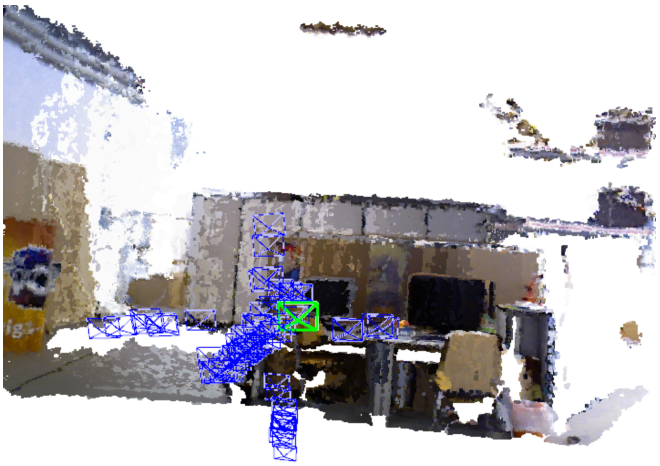
Many applications would be greatly benefited from a solution for this problem, *e.g.*, augmented reality, autonomous vehicles, service robots and medical imaging, among others. In the main, all of them that, for instance, could reuse maps.

In this work we propose an algorithm for dealing with dynamic objects in RGB-D, stereo and monocular SLAM.



(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 removing the dynamic objects.

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

*An extended version of this work has been submitted to the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018).

*This work has been supported by NVIDIA Corporation through the donation of a Titan X GPU, by the Spanish Ministry of Economy and Competitiveness (projects DPI2015-68905-P and DPI2015-67275-P, FPI grant BES-2016-077836), and by the Aragón regional government (Grupo DGA T04-FSE).

Berta Bescós, 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

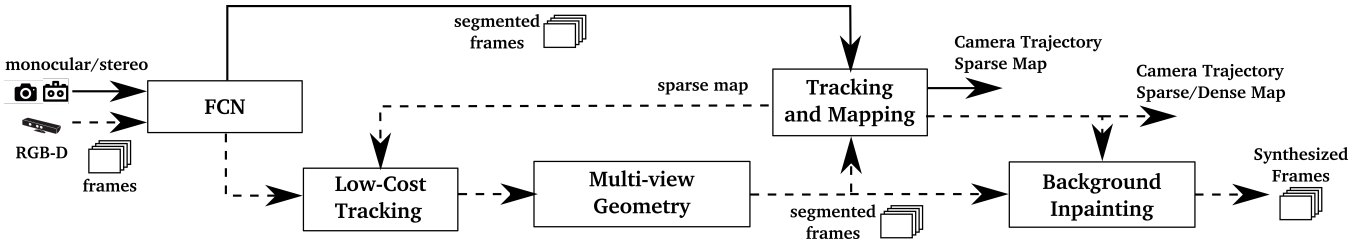


Fig. 2: Block diagram of our methodology. The stereo and monocular images (black continuous line) pass through a FCN for computing the segmentation of the *a priori* dynamic objects before being used for mapping and tracking. In the RGB-D case (black dashed line) a multi-view geometry based approach is added for detecting all dynamic objects, for which we need a low-cost tracking. We do the inpainting of the occluded background once the position of the camera is known.

This is done by adding a front-end stage to the state-of-the-art ORB-SLAM2 system [8], 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 Fully Convolutional Network (FCN) to detect the *a priori* dynamic objects in the frames, *e.g.*, people, cars, trucks, *etc*, 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).

II. SYSTEM DESCRIPTION

Fig. 2 shows an overview of our system. First of all, the RGB channels pass through a FCN that segments out 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. Firstly, we can refine the segmentation of the dynamic objects. Secondly, we can label as dynamic new object instances that are static most of the times (*i.e.*, detect *moving* objects that were not set to *movable* in the FCN 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.

Once this full dynamic objects detection and the localization of the camera have been done, we aim to reconstruct the occluded background of the current frame with static information from previous views.

In the monocular and stereo cases, the images are segmented out by the FCN so that the features extractor algorithm does not work with keypoints belonging to the *a priori* dynamic objects in both the tracking and mapping threads.

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

A. Segmentation of Potential Dynamic Content using FCN

For detecting dynamic objects we propose to use a Fully Convolutional Network (FCN). In our experiments we use Mask R-CNN [9], which extends Faster R-CNN [10] by adding a branch that predicts the segmentation mask in parallel with the already existing branch for bounding box recognition. He *et al.* have not published their code yet, but different researchers have implemented variations [11].

The input of the FCN is the RGB original image. The idea is to segment those classes that are potentially dynamic or movable (*e.g.*, people, dog, cat, car). The output of the network, assuming that the input is a RGB image of arbitrary 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 channel $i \in l$ a binary mask is obtained. By combining all the channels into one image we would obtain the segmentation of all those dynamic objects appearing in one image of the scene.

B. Low-Cost Tracking

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

The tracking implemented in this stage of the algorithm is a low-cost version of the one in Mur and Tardos [8]. 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 FCN and Multi-view Geometry

By using Mask R-CNN, most of the dynamic content can be segmented and therefore not used for both the tracking and the mapping. However, some objects cannot be detected by this approach because they are not *a priori* dynamic, but are 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. This approach for dealing with these changes in the scene is detailed in this section.

For each input frame, we select those previously processed keyframes that have the maximum overlapping of the scene with itself. This is done taking into account both the distance and the rotation between the new frame and each of the keyframes, similarly to Tan *et al.* [12]. The number of selected keyframes has been set to five in our experiments.

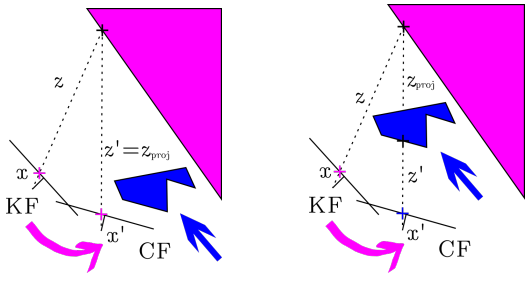
We then compute the projection of each extracted keypoint x of the keyframes into the current frame, obtaining the keypoints x', as well as their projected depth z_{proj}. For each

keypoint we calculate the angle between the rays connecting x and x' with their corresponding 3D map point X, i.e., their parallax angle α. If this angle is greater than 30°, its corresponding keypoints might be subject to occlusions, and will be ignored from now on. Experimentally, we have proved that, in the indoor scenarios of the TUM dataset, with a parallax angle greater than 30° some static objects were considered as dynamic due to the view point difference. We obtain the depth of the remaining keypoints in the current frame z', taking into account the reprojection error, and we compare them with the already computed projected depth of the keyframes z_{proj}. If the difference $\Delta z = z_{proj} - z'$ is over a threshold τ_z the keypoint x' is considered to belong to a dynamic object. This idea can be seen in a schematic manner in the diagram of Fig. 3. In order to set the threshold τ_z we have manually tagged the dynamic objects of a few 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 have concluded that $\tau_z = 0.4m$ is a reasonable choice.

Some of the keypoints that have been previously set to dynamic lay on the borders of moving objects. To avoid this, we use the information given by the depth images. If a keypoint is considered to be dynamic, but a patch around itself in the depth map has a high variance, this keypoint will no longer be tagged as dynamic.

So far, we know which keypoints belong to dynamic objects, and which ones do not. In order to classify all the pixels belonging to these objects, we grow the region around those dynamic pixels in the depth image. An example of a segmented image projected on the RGB frame can be seen in Fig. 4a. Our results show small misalignments due to both the time difference between RGB and depth images, and the depth discontinuities inside a moving object itself. We dilate the segmented regions in order to avoid both effects.

The results of the FCN (Fig. 4b) can be fused together with those of this geometric method for a full dynamic objects detection (Fig. 4c). We can find advantages and disadvantages in both dynamic objects detection methods. Firstly, using geometric approaches, the main problem is that initialization is not trivial because of its multi-view nature.



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

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

Fig. 3: The dynamic keypoints are detected if the difference between z_{proj} and z' is greater than a threshold Δz.

Learning methods and their impressive performance using a single view, do not have such initialization problems. On the other hand, the main limitation of the deep learning method is that objects that are supposed to be static can be moved, and the method is not able to identify them. This causes defects on the reconstruction of the scene. Such defects can be solved by checking the multi-view compatibility.

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. This is due to both the difficulties that RGB-D cameras face when measuring the depth of objects that are far, and 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 methods, we consider they are complementary and therefore the combined use of both is an effective way to achieve an accurate tracking and mapping. In order to achieve this goal, if an object has been detected with both approaches, the segmentation 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 explained above can be seen in the Fig. 4c. These segmented dynamic parts are then removed from the current frame.

D. Tracking and Mapping

The input to this stage of the system contains the RGB and depth images, as long 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.

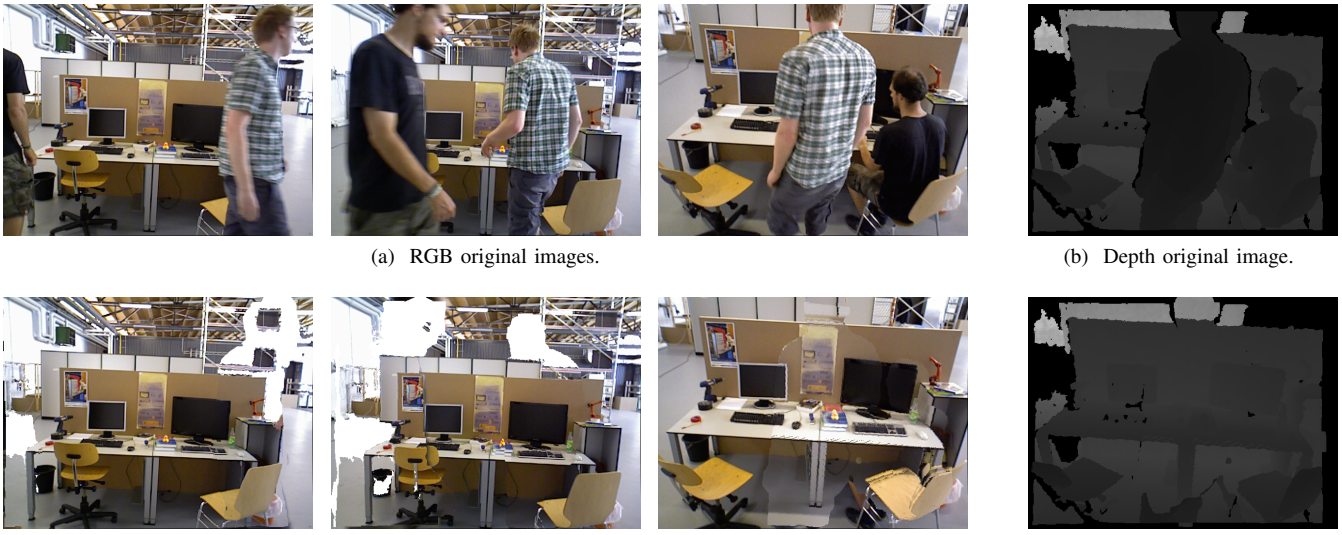
E. Background Inpainting

For every dynamic object that is removed, we also aim at inpainting the occluded background with static information from previous views, so that we can synthesize a realistic image without moving content for virtual reality applications.

First, we project into the dynamic segments both the weighted color and depth from a set of all the previous keyframes (the last twenty in our experiments). Secondly, some gaps have no correspondences and are left blank: it can be due to the difference of field of view between the cameras –two pixels in the keyframes might correspond to three pixels in the input frame–. Besides, another reason why some areas can not be inpainted is because their correspondent part of the scene has not appeared so far in the keyframes, or, if has appeared, it has no valid depth information. The first

(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.



(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.

ones usually have a small size and can easily be inpainted with the color and the depth from the neighbors. The second ones can not be reconstructed with geometrical methods and would need a more elaborated inpainting technique.

Fig. 5 shows the synthetic images for three input frames from different sequences of the TUM RGB-D benchmark. The first row contains three RGB and one depth original images, and the second row shows the inpainted ones. 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 background.

III. EXPERIMENTAL RESULTS

We evaluate our system using the TUM RGB-D dataset and compare to other state-of-the-art SLAM systems in dynamic environments. Besides, we compare our evaluation to the original ORB-SLAM2 system to quantify the improvement of our approach in dynamic scenes. Mur and Tardos [8] propose to run each sequence five times and show median results, to account for the non-deterministic nature of the system. In our case, we run each sequence ten times, as dynamic objects are prone to exalt this effect.

A. TUM Dataset

The TUM RGB-D dataset [13] sequences have been recorded using a Microsoft Kinect sensor in different indoor scenes at full frame rate (30Hz). Both the RGB and the depth images are available, together with the ground-truth, obtained from a high-accuracy motion-capture system. In the sequences named *sitting* there are two people sitting in a desk while speaking and gesticulating, *i.e.*, there is very few motion. In the sequences named *walking*, two people walk both in the background and the foreground. This dataset is highly dynamic and therefore challenging for standard

Sequence	Our system (N)	Our system (G)	Our system (N+G)	Our system (N+G+BI)
walking_halfsphere	**0.025**	0.035	**0.025**	0.029
walking_xyz	**0.015**	0.312	**0.015**	**0.015**
walking_rpy	0.040	0.251	**0.035**	0.136
walking_static	0.009	0.009	**0.006**	0.007
sitting_halfsphere	**0.017**	0.018	**0.017**	0.025
sitting_xyz	0.014	**0.009**	0.015	0.013

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

SLAM systems. For both types of sequences *sitting* and *walking* there are four camera movements: 1) **halfsphere** The camera moves following the trajectory of a 1m 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 for the experiments, as error metric, the absolute trajectory RMSE that has been proposed by Sturm *et al.* [13].

The results of different variations of our system for six different sequences within this dataset are shown in Table I. Firstly, Our system (N) stands for the system in which only the Fully Convolutional Network (FCN) segments out the *a priori* dynamic objects. Secondly, in Our system (G) the dynamic objects have been only detected with the multi-view geometry method based on depth changes. Thirdly, Our system (N+G) stands for the system in which the dynamic objects have been detected combining both the geometrical and the deep learning approaches. Finally, we have considered interesting to analyze the system that is shown in Fig. 6. In this case (N+G+BI), the background inpainting stage (BI) is to be 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 error.

Adding the background inpainting stage (BI) before the localization of the camera usually leads to less accuracy

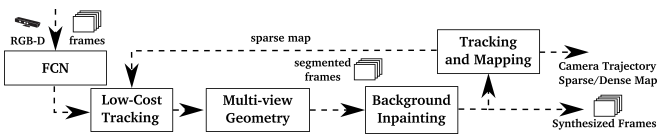


Fig. 6: Block diagram of RGB-D Our system (N+G+BI).

in the tracking. The reason is that that the background reconstruction is strongly correlated with the camera poses. Hence, in those sequences in which there are pure rotation movements (*rpy* mainly, and *halfsphere*), the estimated positions of the cameras 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 Our system results shown from now on are from the best variant, that is, (N+G).

Table II shows our results in the same six 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 scenarios ($sitting$) the tracking results are slightly worse because the tracked keypoints find themselves further than those belonging to dynamic objects. Albeit, Our system's map does not contain the dynamic objects that appear along the sequence. Fig. 7 shows an example of the estimated trajectories of Our system and ORB-SLAM2, compared against the ground-truth. Our system trajectory is almost indistinguishable from that of the ground-truth.

Sequence	ORB-SLAM2 (RGB-D) [8]	Our system (N+G) (RGB-D)
walking_halfsphere	0.351	**0.025**
walking_xyz	0.459	**0.015**
walking_rpy	0.662	**0.035**
walking_static	0.090	**0.006**
sitting_halfsphere	0.020	**0.017**
sitting_xyz	**0.009**	0.015

TABLE II: Comparison of the RMSE of ATE [m] of Our system against ORB-SLAM2 for RGB-D cameras.

Table III shows, in six challenging dynamic sequences, a comparison between our system and several state-of-the-art RGB-D SLAM systems designed for dynamic environments. Our system significantly outperforms all of them in all

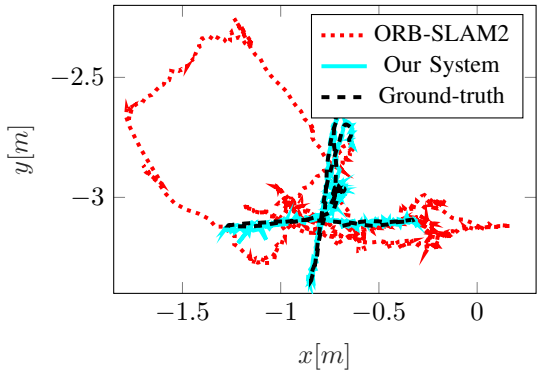


Fig. 7: Estimated trajectory with Our system, ORB-SLAM2 and ground-truth in the TUM sequence $fr3/walking\_xyz$.

Sequence	Depth Edge SLAM [14]	Motion Segmentation DSLAM [15]	Motion Removal DVO-SLAM [16]	Our system (N+G) (RGB-D)
$walking\_halfsphere$	0.049	0.055	0.125	**0.025**
$walking\_xyz$	0.060	0.040	0.093	**0.015**
$walking\_rpy$	0.179	0.076	0.133	**0.035**
$walking\_static$	0.026	0.024	0.066	**0.006**
$sitting\_halfsphere$	0.043	-	0.047	**0.017**
$sitting\_xyz$	0.040	-	0.048	**0.015**

TABLE III: Comparison of the absolute trajectory RMSE [m] of our method against the state-of-the-art RGB-D SLAM systems for dynamic scenes. Our tracking results are estimated with the FCN and the multi-view geometry stages (N+G).

sequences (high and low dynamic). The error is, in general, similar to that of the state of the art in static scenes (1-2 cm).

Monocular ORB-SLAM in highly dynamic scenes is, in general, more accurate than RGB-D SLAM. The reason is the initialization algorithm of ORB-SLAM and RGB-D SLAM. RGB-D SLAM 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 Our system (monocular) in some sequences of the TUM dataset. The initialization in Our system 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 image. In conclusion, although the accuracy of Our system is slightly worse, it succeeds in bootstrapping the system with dynamic content and producing a map free of such content (see Fig. 1), to be re-used for long-term applications.

Sequence	ORB-SLAM [8]		Our system (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: RMSE of ATE [m] and the tracked trajectory percentage for Our system and ORB-SLAM (monocular).

IV. CONCLUSIONS

We have presented a full visual SLAM system for dynamic environments that can use monocular, stereo and RGB-D cameras. Our system is able to accurately track the camera and create a static and therefore reusable map of the scene. In the RGB-D case, Our system 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.

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

In the TUM Dynamic Objects dataset, Our system is 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 and an earlier initialization.

Future extensions of this work might include a real time performance, a RGB-based motion detector, or a more realistic appearance of the synthesized RGB frames.

REFERENCES

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

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Joint Pattern Recognition Symposium*, pp. 11–20, Springer, 2010.

[4] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2320–2327, IEEE, 2011.

[5] G. Graber, T. Pock, and H. Bischof, "Online 3D reconstruction using convex optimization," in *2011 IEEE International Conference on Computer Vision Workshops*, pp. 708–711, IEEE, 2011.

[6] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision*, pp. 834–849, Springer, 2014.

[7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[8] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *arXiv preprint arXiv:1703.06870*, 2017.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[11] Matterport, "Mask RCNN." https://github.com/matterport/MaskRCNN, 2017.

[12] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular SLAM in dynamic environments," in *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pp. 209–218, 2013.

[13] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 573–580, IEEE, 2012.

[14] S. Li and D. Lee, "RGB-D SLAM in Dynamic Environments Using Static Point Weighting," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2263–2270, 2017.

[15] Y. Wang and S. Huang, "Towards dense moving object segmentation based robust dense RGB-D SLAM in dynamic scenarios," in *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*, pp. 1841–1846, IEEE, 2014.

[16] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robotics and Autonomous Systems*, vol. 89, pp. 110–122, 2017.