DOT: Dynamic Object Tracking for Visual SLAM

Irene Ballester Campos

July 2020

1 Introduction

Simultaneous Localisation and Mapping, commonly known by its acronym SLAM, is one of the fundamental capabilities for the autonomous navigation of robotic platforms [3]. Its goal is the joint estimation of the robot motion and a map of its surroundings, from the information of its embedded sensors. Visual SLAM, for which the sensors are mainly, or exclusively, cameras, is one of the most challenging yet relevant configurations.

Despite the significant advances in SLAM in recent years, most systems still assume a static environment, where the relative position between the 3D points in the scene remains unchanged, the only movement being that of the camera. Following this fundamental assumption, camera-pose estimation algorithms attribute the changes between two images exclusively to the relative transformation due to camera displacements. Therefore, they can not account for the effects of moving objects. At best, some algorithms can detect and treat them as outliers [15, 16] to be ignored during the pose tracking and map estimation process. However, this does not prevent that, during the time interval elapsed until their detection as moving objects, the associated information is integrated into the estimation assuming scene rigidity, introducing errors and inconsistencies in the pose and map estimations. Moreover, for those visual SLAM approaches that base the pose tracking on the matching of a small number of key-points, the errors generated by dynamic elements can be fatal and even lead to system failure.

The world of real applications in which a robot must operate is, in general, far from being completely static: autonomous navigation of vehicles such as cars or drones, augmented reality applications or terrestrial and even planetary exploration tasks (where the lack of identifiable characteristics in the images makes SLAM systems precarious in the presence of shadows or other robots). It is therefore necessary to develop visual SLAM systems with the necessary robustness to operate in highly dynamic environments. This was the motivation for this work, which is aimed at developing an image processing strategy that improves the robustness of a visual SLAM system in environments containing moving objects. As a result, we developed “Dynamic Object Tracking” (DOT), a front-end system that combines semantic instances with multi-view geometry to estimate the movement of the camera as well as that of scene objects using direct methods [4]. The result of the pre-processing is a mask encoding both static and dynamic parts of each image fed into the SLAM system, so as to not use the correspondences found in the dynamic regions. The study includes an experimental validation specifically designed to evaluate the system’s ability to effectively reduce the errors associated with SLAM mapping and motion estimation.

The main contributions of our proposed system can be summarised as:

- Significant improvement in the robustness and accuracy of the coupled SLAM system in highly dynamic environments.
- Independence with respect to the particular SLAM system, which makes it a versatile front-end that can be adapted with minimal integration work to any state-of-art visual odometry or SLAM system.
- Unlike other systems, it can be implemented to operate in real time, since DOT allows semantic segmentation to be performed at a lower frequency than that of the camera.

- Robustness against neural net segmentation errors.

The methods developed in this work have been optimised for the specific domain of car navigation. Outside this scenario, the proposed strategy would still be valid at a high level, but parts of it may need some adaptations (e.g. training of semantic segmentation).

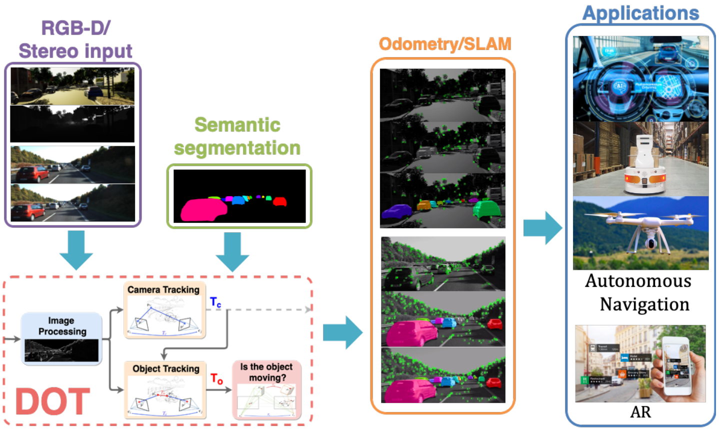

Figure 1: Overview of DOT.

2 Related Work

SLAM in dynamic environments is still an open research problem. The different works in the literature can be divided into three main categories.

The first of the categories, and the most general one, models the scene as a set of non-rigid parts, hence including deformable and dynamic objects [17, 11, 12]. While this research line is relevant because of its generality and potential applications, it also poses significant challenges mainly related to deformation models. In this work, we consider that the world is composed of a variable number of rigid solids, which is the premise behind the other two categories of dynamic visual SLAM.

The second category of works within dynamic visual SLAM aims to improve the accuracy and robustness of a SLAM system by reconstructing only the static part of a scene. Dynamic objects are segmented out (and ignored for mapping purposes), in order to prevent the tracking algorithm from using correspondences belonging to these objects. Along this line of work, DynaSLAM [1], built on top of ORB-SLAM2 [16], aims to estimate static maps that can be reused in long-term applications. Dynamic objects are removed by combining 1) semantic segmentation for potentially moving objects, and 2) multi-view geometry for detecting inconsistencies in the rigid model. Mask R-CNN [8] is used for semantic segmentation, which detects and classifies the objects in the scene into different categories, some of which have been pre-set as potentially dynamic (e.g., car or person). DynaSLAM was designed to mask out all the potentially mobile objects in the scene. This results in a lower accuracy than the original ORB-SLAM2 in scenes containing potentially mobile objects that are not actually moving (e.g., with many cars parked) since removing image tracks located on the potentially moving, but actually static, objects impacts negatively on the camera path estimation process. The aim of this work is, precisely, to overcome this problem as only those objects that are moving at that precise moment will be labelled as dynamic.

Finally, the third line of work in dynamic visual SLAM, which goes beyond the segmentation and suppression of dynamic objects, includes works such as MID-Fusion [20] and MaskFusion [18]. Their aim is to reconstruct the background of the scene and also to estimate the movement of the different dynamic objects. For that purpose, sub-maps of each possible moving object are created and a joint estimation of both the objects and camera poses is carried out.

Many of the mentioned systems involve deep learning methods, which in some cases can not be currently implemented in real-time due to the limited frequencies of the segmentation network. The contribution developed in this work eliminates the requirement to segment all the frames, which allows the system to be independent of the segmentation frequency of the network, so enabling its implementation in real time.

3 DOT

3.1 System Overview

Fig. 2 is an overview of our proposed system. DOT receives as input either RGB-D or stereo images at a certain video rate, and produces a mask encoding the static and dynamic elements of the scene, which can be directly used by SLAM or odometry systems.

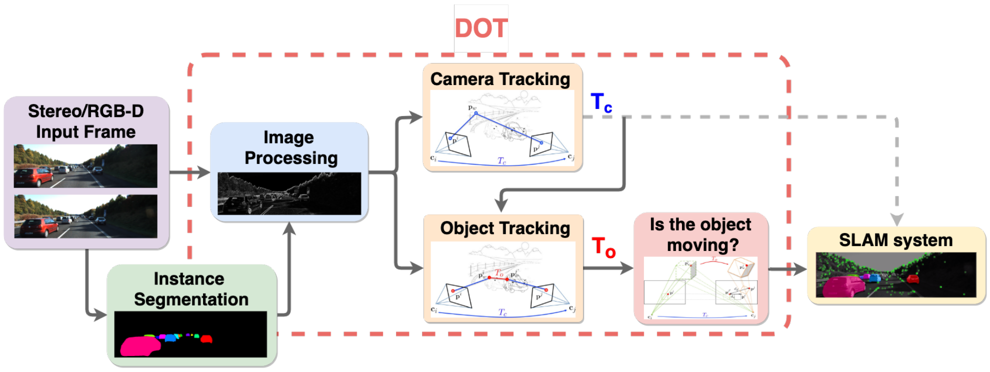

Figure 2: Block diagram of DOT.

The first block (*Instance Segmentation*) corresponds to the CNN that segments out pixelwise all the dynamic objects (in our experimental part, only vehicles are considered). As explained below, the frequency at which the network operates does not need to be that of the video, but can be lower.

The image processing block (*Image processing*) extracts and separates the points belonging to static regions of the image and the points that are in dynamic objects. Camera tracking is estimated by using only the static part of the scene. From this block, and taking into account the camera pose, the movement of each of the objects segmented by the network is calculated independently (*Object tracking*).

The last block (*Is the object moving?*) determines, from geometric calculations, whether the objects previously labelled as dynamic by the network are indeed moving. This information is used to update the masks encoding the static and dynamic regions of each frame and to feed the linked odometry/SLAM visual system.

3.2 Instance Segmentation

We use Mask R-CNN [8] to carry out instance segmentation of all potentially movable objects that are present in the scene. The net output has been modified to obtain in a single image all the segmentation masks. What has not been classified into one of its categories is labelled as 'background' and is considered as static in the subsequent blocks.

We use the Mask R-CNN implementation [14] along with the pre-trained Ms COCO [13] weights. The classes have been restricted to those considered as potentially movable, excluding humans since people tracking is beyond the scope of this paper. In case other categories were needed, the net could be fine-tuned using these weights as a starting point or trained from scratch with its own dataset.

3.3 Camera and Object Tracking

This part of the system estimates the pose of the segmented objects over time. For that purpose, first we need to calculate the motion of the camera $\mathbf{T}_c$ and then, we substract it to track the motion of each object $\mathbf{T}_o$.

3.3.1 Camera Tracking

Camera motion can be estimated from the static points of the scene by using multi-view geometry equations [7] when the camera calibration and point correspondence are known. Figure 3 represents the camera at two instants of time j and i, identified in the figure as c_j and c_i, and the projection of a fixed point $\mathbf{p}$ in both frames.

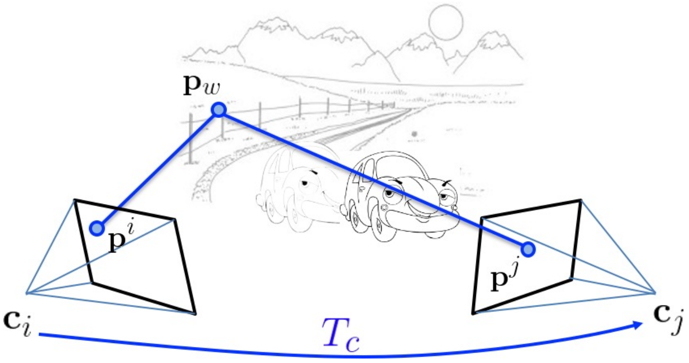

Figure 3: Projection model of a static point.

The diagram in Figure 4 shows the projection process of a point from its coordinates in the image F_j to the corresponding coordinates in the image F_i, as a function of the camera calibration $\mathbf{K}$, the projection model Π (*pinhole*) and the depth of the point z_c.

$$\begin{tikzcd}
 & \text{Pixel coordinates } \mathbf{p} & \text{Normalized 2D image coordinates } \mathbf{p}_n & \text{3D camera coordinates } \mathbf{p}_c & \text{3D global coordinates } \mathbf{p}_w \\
F_j: & \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}_j \arrow[r, "\mathbf{K}^{-1}"] & \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix}_j \arrow[r, "z_c"] & \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}_j \arrow[rd, "\mathbf{T}_j^w(\mathbf{R}_j^w, \mathbf{t}_j^w) = \mathbf{I}_4"] & \\
 & & & & \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}_w \arrow[ld, "\mathbf{T}_w^i(\mathbf{R}_w^i, \mathbf{t}_w^i) = \mathbf{T}_c"'] \\
F_i: & \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}_i & \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix}_i \arrow[l, "\mathbf{K}"'] & \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}_i \arrow[l, "\frac{1}{z_c}"'] &
\end{tikzcd}$$

Figure 4: Diagram of the projection process of a static point in the image between frames taken at moments j and i.

The process described in the diagram above can be condensed into the equation 1. It expresses the projection of a static point $\mathbf{p}$, from its pixel coordinates in the frame F_j to its corresponding coordinates in the frame F_i:

$$\mathbf{p}^i = \Pi(\mathbf{T}_c \Pi^{-1}(\mathbf{p}^j, z_j)). \tag{1}$$

where $\mathbf{T}_c$ describes the relative camera motion, Π and Π^{-1} correspond to perspective projection and back-projection, respectively, and z_j is the depth of the point in the frame j.

Once the relationships between the pixel coordinates at different times are known, in order to estimate the camera movement, an equivalent non-linear optimization problem must be solved. The problem unknowns are the translation and the rotation of the camera and the cost function is the reprojection error; in other words, the rotation and translation which the camera has been subjected to are those which place the projections of the points in their correct correspondences between images.

In this work, photometric error has been used as a reprojection error, which places our method within the category of direct methods [4]. Unlike indirect methods, such as ORB-SLAM2, direct methods do not perform an intermediate feature extraction, but directly use the pixel intensity to estimate the camera pose by minimizing the photometric error. Equation 2 defines the photometric error, e_c, as the sum of the difference between the intensities of the point in the frame being tracked and the point in the reference frame. For the calculation of the camera pose, only the points that have been marked as static, P_s, are used:

$$e_c = \sum_{\mathbf{p} \in P_s} |I_j(\mathbf{p}^j) - I_i(\Pi((\mathbf{T}_c \Pi^{-1}(\mathbf{p}^j, z_j)))|_\gamma. \tag{2}$$

3.3.2 Object tracking

Once the transformation matrix of the camera $\mathbf{T_c}$ has been found, the pose of each one of the potentially movable objects can be estimated in a similar way by using the points belonging to each object.

The model for relating the coordinates of a point belonging to a moving object between two frames is an extension of the one explained above for the camera tracking. In this case, as shown in Figure 5 (compare with the figure 3), the difference between the coordinates of the point in two images taken at different moments, p_i and p_j, is not only due to the movement of the camera but also to the movement of the object, expressed by its transformation matrix, $\mathbf{T}_o$.

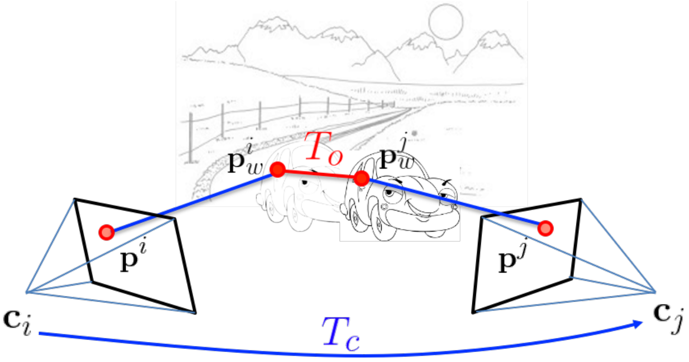

Figure 5: Projection model of a dynamic point.

The relations between the different coordinate systems are identical to those presented above for a static point, with the difference that now it is necessary to take into account that the point $\widetilde{\mathbf{p}}_w$ no longer remains stationary ($\widetilde{\mathbf{p}}_w^j \neq \widetilde{\mathbf{p}}_w^i$), but moves following the rigid body motion equations. The relation between pixel coordinates in different frames of a dynamic point, $\widetilde{\mathbf{p}}$, belonging to an potentially movable object is shown in Fig. 6.

$$\begin{tikzcd}
 & \text{Pixel coordinates } \widetilde{\mathbf{p}} & \text{Normalized 2D image coordinates } \mathbf{p}_n & \text{3D camera coordinates } \mathbf{p}_c & \text{3D global coordinates } \mathbf{p}_w \\
F_j: & \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}_j \arrow[r, "\mathbf{K}^{-1}"] & \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix}_j \arrow[r, "z_c"] & \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}_j \arrow[r, "\mathbf{I}_4"] & \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}_j \arrow[d, "\mathbf{T}_o"'] \\
F_i: & \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}_i & \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix}_i \arrow[l, "\mathbf{K}"'] & \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}_i \arrow[l, "\frac{1}{z_c}"'] & \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}_i \arrow[l, "\mathbf{T}_c"']
\end{tikzcd}$$

Figure 6: Diagram of the projection process of a dynamic point in the image between frames taken in the instants j and i.

Accordingly, the equation that relates the pixel coordinates of a dynamic point between frames taken in the instants j and i can be rewritten as:

$$\widetilde{\mathbf{p}}^i = \Pi(\mathbf{T}_c \mathbf{T}_o \Pi^{-1}(\widetilde{\mathbf{p}}^j, z_j)). \tag{3}$$

Finally, since we have already calculated the camera pose $\mathbf{T}_c$, the only remaining unknown in the equation 3 is the movement of the object $\mathbf{T}_o$. It can be estimated, as for the static points, by minimising the sum of re-projection photometric errors, e_o, of the points that belong to each object,

P_o:

$$e_o = \sum_{\mathbf{p}\in P_o} |I_j(\mathbf{p}^j) - I_i(\Pi(\mathbf{T}_c\mathbf{T}_o\Pi^{-1}(\tilde{\mathbf{p}}^j, z_j)))|_\gamma. \tag{4}$$

3.3.3 Optimization method

The resolution of the non-linear problem of camera movement is usually solved with small modifications from the Gauss-Newton iterative optimization method. In this work we have used a pre-existing implementation of the problem with the *Ceres* library. *Ceres* offers a versatile and efficient C++ solution for the resolution of non-linear problems that includes, among other tools, the calculation of the Jacobians by self-differentiation of the error. However, to make the algorithm faster and more robust we use an implementation that incorporates the analytically calculated Jacobians.

3.4 Is the object moving?

The last block receives as input the transformation matrices of the camera, $\mathbf{T}_c$, and the objects, $\mathbf{T}_o$, and estimates whether the objects are moving or not. Its output are the dynamic object masks, to be used by SLAM or odometry systems.

It could be thought that deviations of the object transformation $\mathbf{T}_o$ from the identity will indicate that the object is moving. However, in practice, the results after the optimisation are affected by the correspondence noise, which makes it very difficult to distinguish which part of the variations is due to noise in the results and which part is due to a change in the position of the object. In this work we chose to determine the movement of the objects using 2D measurements in the images. Fig. 7 shows the projections of an object if it is static (green) or if it is moving (orange).

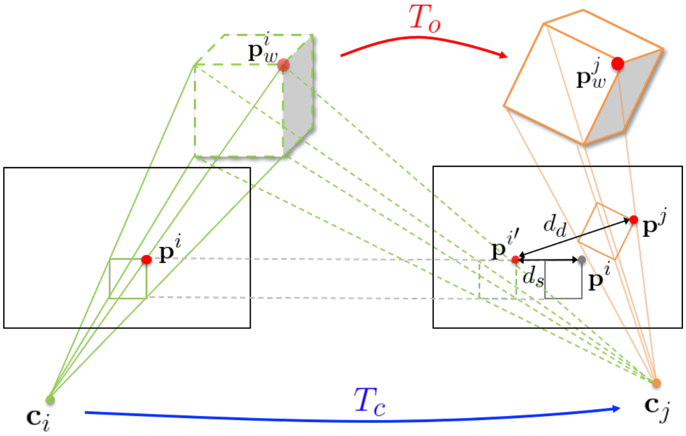

Figure 7: Projection diagram of a point belonging to a dynamic object in two different frames j and i.

To determine whether an object is in motion we use a metric that has been named as *dynamic disparity*:

$$d_d = \left|\left|\mathbf{p}^i, \tilde{\mathbf{p}}^i\right|\right|, \tag{5}$$

which is the distance between the projection of the point as if it was static and its real projection. To determine if the object is moving, the median of the dynamic disparities of all the points belonging to object must be greater than a threshold θ_d. It should be noted that this definition of disparity does not correspond to the one commonly used term in stereo vision.

If an object is moving, but this results in a dynamic sub-pixel disparity, the algorithm will mislabel it as static. This constraint affects objects depending on 1) the position of the point in the image, 2) its depth, and 3) the relative angle between the directions of object and camera movements. In order to be able to detect whether an object is static, its relative movement to the camera has to be adequate so that all three possible directions of movement are observable. To do so, we calculate the motion in the image due to the relative movement with the camera $(\Delta x_c, \Delta y_c, \Delta z_c)$:

$$d_x = f_x \frac{\Delta x_c}{z_j}, \qquad d_y = f_y \frac{\Delta y_c}{z_j}, \qquad d_z = \frac{\Delta z_c}{z_j(z_j + \Delta z_c)} \sqrt{(f_x x_j)^2 + (f_y y_j)^2}. \tag{6}$$

If the value that we call *static disparity* $d_s = \min(d_x, d_y, d_z)$ is above a certain threshold θ_s we can say that the object staticity is observable and that, therefore, if the median dynamic disparity of all points is below than a certain value θ_d the object is not moving.

A summary of these considerations can be found in Figure 8. In the particular case that at a certain instant t we cannot determine the movement of the object, but we have already made previous observations, then we can propagate the information from the immediately preceding observation.

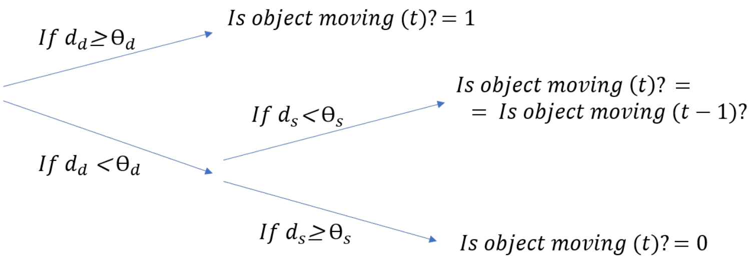
$If\ d_d \geq \Theta_d$
$Is\ object\ moving\ (t)? = 1$
$If\ d_s < \Theta_s$
$Is\ object\ moving\ (t)? =$
$= Is\ object\ moving\ (t-1)?$
$If\ d_d < \Theta_d$
$If\ d_s \geq \Theta_s$
$Is\ object\ moving\ (t)? = 0$

Figure 8: Diagram of logical relationships for the detection process of moving objects.

In principle, the minimum threshold value for θ_d would be 1 pixel, but in practice a final consideration must be added. Since the camera and objects movement estimations have some uncertainty associated, this uncertainty must also be taken into account when setting the thresholds. Thus, θ_d should be equal to 1 pixel when such uncertainties are negligible, but should grow continuously as they increase. The accuracy of motion estimates can be quantified in terms of Pearson's correlation coefficients for the camera, ϕ_c, and the objects, ϕ_o. These coefficients, which can vary between 0 and 1, measure the degree of linear correlation between the intensities of the reference points and their corresponding estimates at a later instant (1 and 0 would indicate perfect linear relationship and no correlation, respectively). The function adopted in this work to relate θ_d to these coefficients is the following:

$$\theta_d = \frac{1}{\phi_c \phi_o} \tag{7}$$

While selecting the optimal functional form would require further study, this expression meets the requirements (gradual increase from 1 to infinity as the coefficients are reduced) and has shown good results in this work. It should be noted that this method offers robust behaviour to changes in lighting with respect to the reference frame, which could invalidate other alternatives based on intensity analysis, but which in principle would not affect the correlation coefficients.

Therefore, labelling an object as static requires the median of the disparity to be below one pixel when the estimations are accurate, but this limit relaxes as the associated uncertainty increases.

Figure 9 is an example of the mask propagated by DOT. Objects labelled as "moving" are represented in colour, while those labelled as "static" disappear in black. The cars represented in grey are those for which we cannot determine its movement.

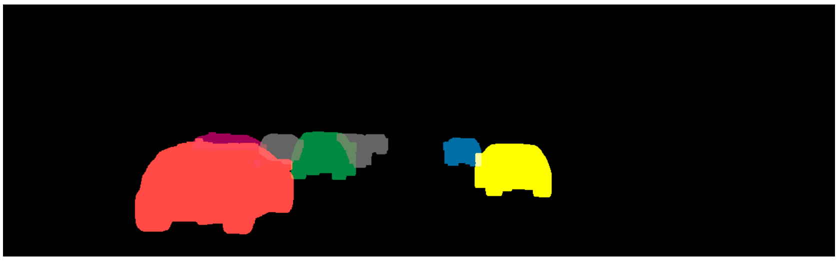

Figure 9: Example of the resulting mask calculated by DOT.

4 Experimental results

Description of the experiment. Although the potential applications of DOT cover a wide spectrum ranging from object detection to augmented reality or autonomous driving among others, in this work an intensive evaluation has been made to demonstrate to what extent "knowing the movement of objects" can improve the accuracy of a SLAM system. The experiment consists in estimating the trajectories of a camera using a state-of-art SLAM system along with three different configurations to evaluate if, indeed, the one in which DOT has been implemented provides the best results in both accuracy and robustness. The SLAM system used in this work is ORB-SLAM2 [16].

Configurations. The three configurations chosen to evaluate the performance achieved with DOT are:

- *No masks*: The SLAM system works with its original implementation on the unmodified images. A static scene is assumed, so that all the points in the images (including those belonging to moving objects) can be selected by ORB-SLAM2[16] to estimate the camera movement.

- *DOT masks*: This is the method developed in this work, in which the SLAM system receives as input, in addition to the images, the instances generated by the net and modified by DOT to detect the moving cars. In this way, ORB-SLAM2 [16] does not extract points from objects that have been tagged as dynamic by the net and have been confirmed by DOT to be in motion.

- *All Masks*: This method provides SLAM with all the masks obtained by the neural network without verifying that the objects segmented as potentially mobile are indeed in motion. In this way, all potentially mobile objects are eliminated.

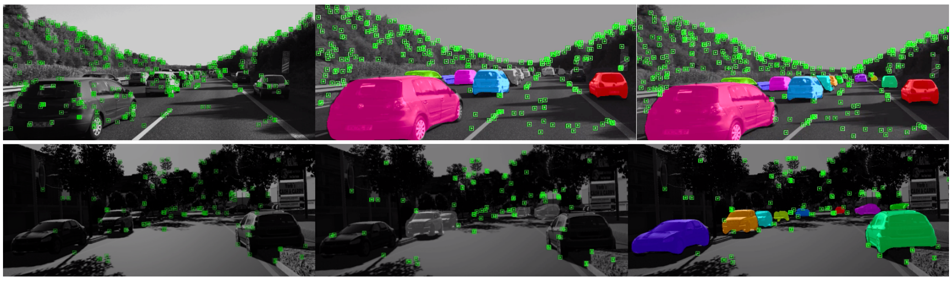

Figure 10: Sample results for the three studied configurations. Left: *No masks*. Centre: *DOT masks*. Right: *All Masks*. In colour, the cars eliminated for the processing with *DOT masks* and *All Masks*.

Evaluation. As usual in experiments with real-time SLAM systems, in order to take into account their non-deterministic nature, we have run each configuration 10 times for each sequence and the values reported are the median of the results obtained for each case. All the experiments have been carried out in a laptop computer with an Intel Core i5 processor and 8GB of RAM memory.

Four different metrics are used to assess performance: the absolute translation error, ATE, and the relative pose error, RPE, proposed in [19], and the average relative translation t_{trans} and rotation t_{rot} errors proposed in [6]. To compare DOT with respect to the other two configurations, the average of the errors normalised by the value obtained with DOT, $\overline{\varepsilon}_{norm}$, is calculated for each sequence:

$$\overline{\varepsilon}_{norm} = \frac{1}{n}\sum_{i=0}^{n} \frac{\varepsilon_i}{\varepsilon_{DOT}} \tag{8}$$

KITTI *Dataset.*

We evaluate DOT accuracy and robustness in three public datasets designed for research in autonomous driving. Two of them are part of the KITTI Vision Benchmark Suite [6], which contains stereo sequences of urban and road scenes recorded from a car, where accurate ground truth is provided by a GPS localisation system. The third dataset used for the evaluation is V-KITTI [5] [2], a synthetic dataset virtually cloned from KITTI *tracking* [6].

4.1 V-KITTI dataset

Virtual KITTI [5] [2] is a synthetic dataset composed of 5 sequences generated from KITTI [6] for autonomous driving research. To obtain the trajectories, we have run the RGB-D version of ORB-SLAM2, which receives as input the RGB colour image and its corresponding depths image.

The ATE values in Table 1 show that, on average, the DOT results are 92.6% and 37.8% better than those obtained with *No masks* and *All masks*, respectively. In addition, DOT scores best for 3 of the 5 sequences evaluated.

Seq.	ATE [m]			ATE/ATE$_{\text{best}}$		
	No masks	DOT	All Masks	No masks	DOT	All Masks
01	**1.10**	1.14	1.38	1.00	1.04	1.26
02	0.16	0.14	**0.10**	1.60	1.43	1.00
06	0.11	**0.07**	0.08	1.67	1.00	1.18
18	4.77	**1.00**	1.50	4.79	1.00	1.51
20	29.42	**9.12**	13.54	3.23	1.00	1.49
$\bar{\varepsilon}_{norm}$	192.6%	100.0%	137.8%			

Table 1: Comparison of the DOT results in V-KITTI sequences with the reference methods, *No masks* and *All Masks*. Left: ATE [m]. Right: ATE normalised by the best value of each sequence.

Content scene adaptation. The right columns in Table 1 shows the ATE values normalised by the most accurate value in each sequence among the three configurations. Thus, a value equal to 1 identifies the method with the best result, while values > 1 are indicative of poorer performance. The colour scale indicates the relative position of the errors between the best result (green) and the worst (red). The dominance of the green colour in the DOT column reveals the clear superiority of this option with respect to the other two. This is thought to evidence that while the use of masks may be convenient, the accuracy is significantly improved if only the objects that have been verified to be in motion are disregarded. Furthermore, these results demonstrate that the DOT approach automatically adapts the processing algorithm to the best option, both for static scenes and in cases with moving elements.

Due to the great variety of casuistry produced by dynamic objects, a detailed analysis of the most relevant sequences is presented below.

Static sequences (sequence 1): *urban scene with several cars parked on both sides of the road and few cars in motion.* Considering again the ATE results for sequence 1 in Table 1, the highest accuracy is achieved with the method *No masks* (1.10 m) and the worst for *All masks* (1.38 m). The fact that no car moves along the sequences means that they are practically static and, therefore, it benefits the *No masks* configuration, which considers them as such. As shown in Figure 11, this configuration can extract points from a larger area, resulting in a better accuracy of the estimated camera movement. Since DOT verifies that there are no moving objects, it allows ORB-SLAM2 to use the points located in the parked cars, thereby reducing the loss of useful information and achieving an accuracy that is very close to the best solution (1.14 m).

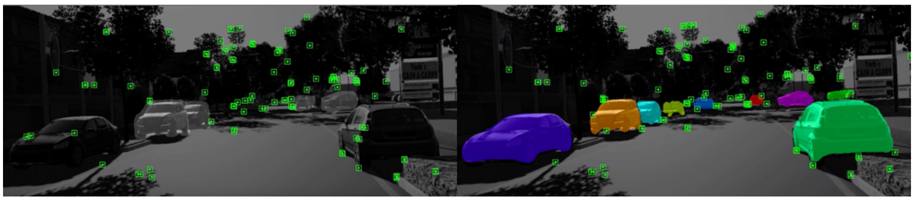

Figure 11: Sequence 1 images from the V-KITTI dataset in ORB-SLAM2. Left: *DOT masks*. Right.: *All Masks*.

Dynamic sequences (sequences 6 y 18): *extra-urban road scenes where all the vehicles are in motion.* The fact that all the vehicles in the scene are moving dramatically violates the assumption of scene rigidity implicit in ORB-SLAM2 to calculate the camera movement. As it can be seen in Table 1, both *All masks* and *DOT* yield the best results in this sequence. The high dynamism of the objects in sequence 18 causes even the failure of ORB-SLAM2 at estimating the camera movement in 6 out of 10 trials (only 56% of the trajectory could be estimated in those cases).

Partially dynamic sequences (sequences 2 and 20): *intermediate situation, with stationary and moving cars. Sequence 20 shows a traffic jam on a motorway where a large part of the image is occupied by moving vehicles, but also including some cars that are not moving.* While not using any mask increases the error due to misusing points located on moving objects, applying all masks without verifying that the object is in motion causes loss of information, especially when a large part of the scene is occupied by vehicles. Such scenarios clearly demonstrate the versatility achieved by DOT, as well as the significant improvements that can be achieved if the actual motion state of the objects is known.

4.2 KITTI *Odometry*

KITTI *Odometry* is a set of KITTI sequences specially designed for the development and evaluation of visual odometry systems. In this case, the inputs to ORB-SLAM2 are the images recorded by a stereo pair of cameras and the timestamp list.

Table 2 displays the ATE values for 11 sequences evaluated in the different configurations. According to these results, DOT obtains an overall performance which is 12.7% and 30.3% better than the *No masks* and *All Masks* methods, respectively.

Seq.	ATE [m]			$\mathrm{ATE}/\mathrm{ATE}_{\mathrm{best}}$		
	No masks	DOT	All Masks	No masks	DOT	All Masks
0	**1.77**	1.80	2.08	1.00	1.02	1.18
1	**6.37**	7.71	8.45	1.00	1.21	1.33
2	3.72	**3.70**	3.84	1.01	1.00	1.04
3	0.40	0.40	0.40	1.00	1.01	1.00
4	0.27	0.26	**0.24**	1.12	1.09	1.00
5	0.40	**0.39**	0.45	1.03	1.00	1.14
6	**0.63**	0.68	0.67	1.00	1.08	1.07
7	0.52	**0.51**	**0.51**	1.01	1.00	1.00
8	**3.04**	3.24	3.78	1.00	1.07	1.24
9	2.65	**0.98**	3.80	2.71	1.00	3.89
10	**1.23**	1.29	1.26	1.00	1.05	1.02
$\bar{\varepsilon}_{norm}$	112.7%	100.0%	130.3%			

Table 2: Comparison of the DOT results in KITTI *Odometry* sequences with the reference methods: *No masks* and *All Masks*. Left: ATE [m]. Right: ATE normalised by the best value of each sequence.

The values of t_{trans} and t_{rot}, listed in Table 3, reflect that this group of sequences, compared to V-KITTI, contains much less dynamic elements, so the use of masks is detrimental. The *All masks* configuration produces higher errors in both parameters: on average, t_{trans} is around 0.81 m/100 m and t_{rot} is around 0.24 °/100 m, to be compared with 0.76 m/100 m and 0.23 °/100 m, respectively, for *No masks*. DOT automatically adapts the masks to the actual dynamism of the scene, with results very similar to those of *No masks* (t_{trans} around 0.77 m/100 m and t_{rot} 0.23 °/100 m). Although the magnitude of the differences among methods is relatively small, the observed behaviour again confirms the versatility of the strategy proposed in DOT.

GPS accuracy. According to the dataset specifications, the ground truth camera poses collected by the GPS is accurate to within 10 cm. Therefore, it can be concluded that no significant differences exist between the three configurations in sequences 3, 4, 5, 6, 7 and 10. This is thought to be a consequence of the small number of moving objects, as well as of the rich texture of the images, which provides a large number of static points for estimating camera movement.

Loops closure. Seven out of the 11 sequences contain loops along their trajectory. When ORB-SLAM2 detects that the camera has reached a point where it has been previously, the trajectory estimated until that point is optimised to make it consistent with the loop closure condition. This ORB-SLAM2 module minimises the drift caused by poor estimation and therefore mitigates inaccuracies produced by dynamic objects or by the removal of useful information from stationary vehicles. The loop is not always identified in all repetitions of a given case, which may result in broader error variability due to the enormous difference in path drift depending on the loop detection. The results

Seq.	No masks		DOT masks		All masks	
	t_{trans} [m/100m]	t_{rot} [°/100m]	t_{trans} [m/100m]	t_{rot} [°/100m]	t_{trans} [m/100m]	t_{rot} [°/100m]
0	**0.76**	0.23	0.77	**0.22**	0.84	0.24
1	**0.77**	**0.23**	0.78	**0.23**	0.85	0.24
2	**0.81**	0.23	0.82	0.23	0.86	0.23
3	0.82	0.23	0.82	0.23	0.82	0.23
4	**0.82**	0.23	**0.82**	0.23	0.86	0.23
5	**0.76**	**0.22**	**0.76**	**0.22**	0.80	0.23
6	**0.71**	0.22	**0.71**	0.22	0.74	0.22
7	0.71	0.22	**0.70**	0.22	0.73	0.22
8	0.74	0.23	**0.73**	0.23	0.77	0.23
9	0.76	**0.23**	**0.75**	**0.23**	0.79	0.25
10	0.76	**0.23**	**0.75**	**0.23**	0.80	0.25
$\overline{\varepsilon}_{norm}$	99.8%	100.92%	100.0%	100.0%	106.2%	103.7%

Table 3: Comparison of t_{trans} [m/100m] and t_{rot} [°/100m] sequences with the reference methods: *(No masks)* and using all masks obtained from the net *(All Masks)*. Left: ATE [m]. Right: ATE normalised by the best value of each sequence.

indicate that DOT tends to increase the ability of ORB-SLAM2 to successfully identify loops. Sequence 9 is a clear example in this regard. Whereas the implementation with DOT closed the loop 6 out of the 10 runs, none was identified when using *All masks*; this clearly suggests that the recovery of some objects ignored when all masks are applied is very helpful for the SLAM system to correctly identify the scene. Some improvement is also observed with respect to the *No masks* configuration, which closed the loop in 4 occasions; whereas the difference with DOT might not be meaningful, it could point to some improvement related to discarding information from non-static objects.

Neural network errors. (sequence 01): *road scene where all the vehicles are in motion.* Considering the high amount of dynamic content in this scene, we would expect the highest accuracy to be achieved with the *All masks* method. However, the results seem to be swapped and the static method (*No masks*) is the one achieving the best accuracy. Such inconsistency is attributed to the fact that the network sometimes mislabels static objects, that could be used to estimate camera movement (e.g. traffic signs or buildings), as dynamic objects. As shown in Figure 12, DOT corrects this effect by re-tagging the object as a static one.

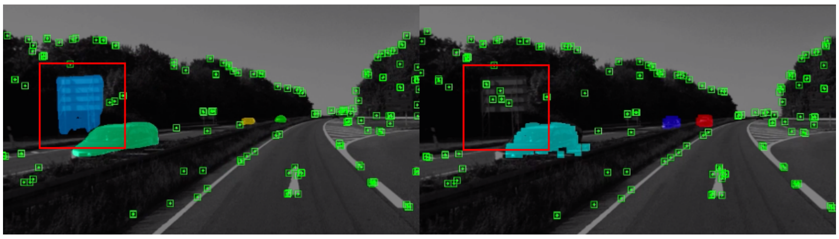

Figure 12: Comparison between the configuration of *All Masks* and *DOT masks* where we can observe the robustness of DOT against network segmentation errors.

4.3 KITTI *Raw*

The following set of sequences has been selected from the *raw* section of the KITTI dataset [6] for their high concentration of moving objects [9]. As for KITTI *Odometry*, we have run the stereo version of ORB-SLAM2.

As shown in Table 4, differences between sequences and methods are more evident with this set of sequences, characterised by a large number of moving objects. Overall, DOT achieves improvements of 142.3% in ATE accuracy over *No masks* and 15.9 % over the *All masks* method. It is somewhat surprising that most sequences produced similar results with DOT and *No masks* for this set, although the case 1003-0047 clearly demonstrates that the moving objects must be discarded in order to avoid such large errors. The improvement with respect to *All masks* is evident in cases 0926-0009/0101 and 1003-0047, whereas both methods display similar results in the rest of the cases.

Seq.	ATE [m]			ATE/ATE$_{\text{best}}$		
	No masks	DOT	All Masks	No masks	DOT	All Masks
0926-0009	**1.23**	1.24	1.44	1.00	1.01	1.17
0926-0013	**0.26**	**0.26**	0.27	1.00	1.00	1.03
0926-0014	0.86	0.82	**0.78**	1.11	1.06	1.00
0926-0051	0.37	**0.36**	0.37	1.02	1.00	1.02
0926-0101	**8.66**	10.26	12.37	1.00	1.18	1.43
0929-0004	0.32	**0.30**	**0.30**	1.08	1.03	1.00
1003-0047	13.81	**1.25**	2.23	11.01	1.00	1.78
$\bar{\varepsilon}_{norm}$	242.3%	100.0%	115.9 %			

Table 4: Comparison of the DOT results in KITTI *Raw* sequences with the reference methods: *(No masks)* and using all masks obtained from the net *(All Masks)*. Left: ATE [m]. Right: ATE normalised by the best value of each sequence.

Equivalent virtual sequences The sequences 0926-0009, 0929-0004 and 1003-0047 were used to generate the V-KITTI synthetic sequences (1, 18 and 20). As expected, given that the content of the scenes are identical, so is the qualitative analysis of the results. In sequences 0926-0009 and 0929-0004, DOT correctly interprets the dynamism of the scenes so that their results are close to the best one. The use of masks in sequence 1003-0047 dramatically improves accuracy because it is the most dynamic scene and DOT also outperforms *All Masks)* because it keeps information from the cars stuck in the traffic jam.

5 Contributions and future work

The DOT system, the development of which has begun in this work, has generated results that are believed to be relevant in their field and which will be presented at a conference (WACV21). Specific plans are also underway to release the code so that it is accessible to the scientific community. However, this work is still under development and is being extended to seek improvements in accuracy and robustness. In addition to the accuracy improvements achieved described in the previous chapter, in this section, we summarize other aspects of DOT that can be considered as relevant contributions, along with some ideas for future work.

5.1 Real-time implementation

The high computational cost involved in the use of neural networks prevents their real-time implementation on low-power platforms (such as mobile phones or drones). In fact, current SLAM work in dynamic environments [1][20] does not include semantic segmentation in normal operation, but rather performs it through image pre-processing.

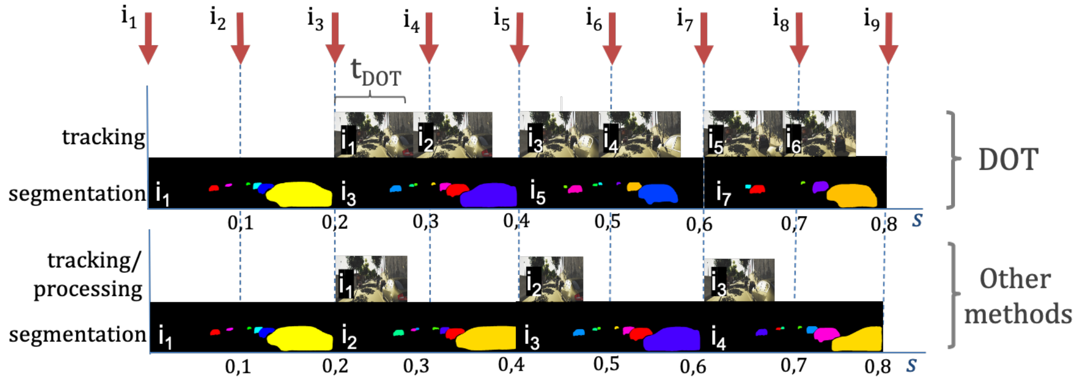

Figure 13: Timeline for DOT performance and comparison with other methods.

As illustrated in Figure 13, our implementation allows for segmentation to be performed every few frames, as DOT can generate new masks from the motion estimation of objects in the scene. The masks provided by a segmentation net are very accurate at low frequencies, when the available computation time is long enough. However, attempts to optimize network performance can significantly compromise the segmentation quality. Therefore, the combination of a neural network operating at low frequency to provide high-quality segmentation masks together with the estimation of the objects movement in the intermediate frames appears as an optimal solution.

5.2 Integration with Odometry/Visual Lam

One aspect of the existing algorithm that might be worth reviewing is the double estimation of the camera movement. In addition to the initial estimation carried out by DOT, this calculation is performed again in the linked SLAM system.

However, this apparent lack of efficiency provides the advantage of keeping DOT's versatility to be combined with any state-of-the-art visual odometry or SLAM system. Nevertheless, for its implementation in specific applications, a full integration would certainly be beneficial.

5.3 Intensive evaluation

In this work the evaluation has focused on demonstrating to what extent "knowing the movement of objects" improves the accuracy of a SLAM system. However, the good results obtained suggest that it would be interesting to carry out additional evaluations in order to try to compare the DOT/ORB-SLAM2 combination with other dynamic SLAM systems that are part of the current state of the art [1][10][18][20].

Furthermore, although the datasets used in this work have been designed for autonomous driving of vehicles, the DOT formulation is perfectly valid for other types of applications with different types of dynamic objects, such as robotic manipulation in structured scenarios (e.g. in offices) or navigation of autonomous drone flight systems. In certain cases, it would be necessary to retrain the segmentation network, but for many others, the application to other sequences seems immediate.

Finally, despite having achieved good accuracy, the improvement in DOT motion estimation could be refined using other strategies, such as the combined use of features with direct methods or the implementation of a kinematic model that adapts the constraints based on the information extracted from the semantic instantiation (for example, if the network has segmented the object as a "vehicle", the roll or tilt angles could be constrained).

5.4 Semantic segmentation methods

DOT relies on Mask R-CNN as a method of semantic segmentation. Yet, there are other neural networks that, apart from the instance segmentation, can provide other type of information, for example, about the movement of objects. Another possible technique would be to feed back the segmentation network with the information on movement generated by DOT. For example, if we are able to estimate the position of the object in the next frame, the network could use this information to search for the object in this region, thereby allowing it to optimize its performance.

6 Conclusions

DOT is a novel front-end algorithm for SLAM systems that combines semantic segmentation with multi-view geometry to estimate camera and object motion using direct methods.

The evaluation of DOT in combination with ORB-SLAM2 in three public datasets for autonomous driving research [6][5][2] demonstrates that DOT-generated object motion information allows the SLAM system to adapt to the scene content and to significantly improve its performance, in terms of both accuracy and robustness.

The independence of DOT from SLAM system makes it a versatile front-end that can be adapted with minimal integration work to any state-of-art visual odometry or SLAM system. In addition, DOT allows semantic segmentation (typically involving high computational cost) to be performed at a lower frequency than the camera, which unlike other systems enables real-time implementation.

We believe that extending the capabilities of DOT can provide significant benefits not only for use in autonomous driving but also for a much broader range of applications in robotics or augmented reality.

References

[1] B. Bescós, J. M. Fácil, J. Civera, and J. Neira. DynSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes. *CoRR*, abs/1806.05620, 2018.

[2] Y. Cabon, N. Murray, and M. Humenberger. Virtual KITTI 2, 2020.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[4] J. Engel, V. Koltun, and D. Cremers. Direct Sparse Odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[5] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2016.

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? the KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[9] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu. ClusterVO: Clustering Moving Instances and Estimating Visual Odometry for Self and Surroundings, 2020.

[10] J. Huang, S. Yang, Z. Zhao, Y.-K. Lai, and S.-M. Hu. ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation. 2019.

[11] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. VolumeDeform: Real-time Volumetric Non-rigid Reconstruction. October 2016.

[12] J. Lamarca, S. Parashar, A. Bartoli, and J. Montiel. Defslam: Tracking and mapping of deforming scenes from monocular sequences. *arXiv preprint arXiv:1908.08918*, 2019.

[13] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft COCO: Common Objects in Context, 2014.

[14] I. Matterport. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow, 2019. URL: https://github.com/matterport/Mask_RCNN [Online. Accedido el 03/12/2019].

[15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[16] 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*, 33(5):1255–1262, 2017.

[17] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[18] M. Rünz, M. Buffier, and L. Agapito. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects, 2018.

[19] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012.

[20] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger. MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM, 2018.