# DOT: Dynamic Object Tracking for Visual SLAM

Irene Ballester[1,2], Alejandro Fontán[1,2], Javier Civera [1], Klaus H. Strobl[2] and Rudolph Triebel[2,3]


Fig. 1. **Top row**: ORB-SLAM2 [1] tracks on KITTI [2] images. **Middle row**: ORB-SLAM2 tracks with DOT segmentation masks, which differentiate between moving and static objects. **Bottom row**: ORB-SLAM2 tracks using Detectron2 [3] segmentation masks, encoding all potentially dynamic objects. Note how DOT segments out actually moving objects (e.g., moving cars), while keeping the static ones (e.g., parked cars).

***Abstract*— In this paper we present DOT (Dynamic Object Tracking), a front-end that added to existing SLAM systems can significantly improve their robustness and accuracy in highly dynamic environments. DOT combines instance segmentation and multi-view geometry to generate masks for dynamic objects in order to allow SLAM systems based on rigid scene models to avoid such image areas in their optimizations.**

**To determine which objects are actually moving, DOT segments first instances of potentially dynamic objects and then, with the estimated camera motion, tracks such objects by minimizing the photometric reprojection error. This short-term tracking improves the accuracy of the segmentation with respect to other approaches. In the end, only actually dynamic masks are generated. We have evaluated DOT with ORB-SLAM 2 [1] in three public datasets. Our results show that our approach improves significantly the accuracy and robustness of ORB-SLAM 2, especially in highly dynamic scenes.**

## I. Introduction

Simultaneous Localization and Mapping, commonly known by its acronym SLAM, is one of the fundamental capabilities for the autonomous navigation of robotic platforms [4]. 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 the last two decades, most state-of-the-art systems still assume a static environment, where the relative position between the scene points does not change and the only motion is done by the camera. With this assumption, SLAM models attribute the visual changes exclusively to the relative camera motion. A usual approach [5], [1] is modeling dynamic areas as outliers, ignoring them during the pose tracking and map estimation processes. However, for several frames, until such dynamic areas are discarded as outliers, their data is used in the SLAM optimization, hence introducing errors and inconsistencies in the estimation of the map and the camera poses. Moreover, for feature-based SLAM methods, that track a small number of salient image points, the errors produced by a relatively small number of matches in dynamic areas are relevant and can lead to the system failure. The world and the real applications in which a robot or an AR system must operate is far from being static. We can cite as representative examples the autonomous navigation of cars or drones, AR in crowded scenes or even planetary exploration tasks, where the poor texture makes SLAM systems precarious in the presence of shadows or other robots. Developing SLAM systems that are sufficiently robust to operate in highly dynamic environments is then essential for many applications.

As shown in Fig. 1, this work aims to develop an image processing strategy that improves the robustness of a visual SLAM system in dynamic environments. Our specific contribution is the development of "Dynamic Object Tracking" (DOT), a front-end that combines instance segmentation with multi-view geometry to track the camera motion, as well as the motion of the dynamic objects, using direct methods [6]. The result of this pre-processing is a mask containing the dynamic parts of each image that a SLAM system can use to avoid making correspondences in such regions.

Our experimental results in three different public datasets show that our combination of semantic segmentation and geometry-guided tracking outperforms the state of the art in dynamic scenes. We also find relevant that DOT is implemented as an independent front-end module, and hence easy-to-plug in existing SLAM systems. As DOT includes short-term mask tracking, we avoid the segmentation of all frames in the sequence, with significant savings in computation. Finally, although we tuned and evaluated DOT for the specific domain of car navigation, our strategy would be valid for other applications.

[1]University of Zaragoza, [2]German Aerospace Center (DLR), [3]Technical University of Munich

## II. Related Work

SLAM in dynamic environments is an open research problem with a large scientific bibliography. We will divide the different approaches into three main categories.

The first category, and the most general one, models the scene as a set of non-rigid parts, hence including deformable and dynamic objects [7][8][9]. While this research line is the most general, it is also the most challenging one. In this paper we will assume intra-object rigidity, which is the premise behind the other two categories of dynamic visual SLAM.

The second category aims to improve the accuracy and robustness of visual SLAM by reconstructing only the static part of a scene. Dynamic objects are segmented out and ignored for camera pose tracking and map estimation. Along this line, DynaSLAM [10], built on top of ORB-SLAM2 [1], aims to estimate a map of the static part of the scene and re-use it 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 [11] 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, which as shown in [10], results in a lower accuracy than the original ORB-SLAM2 in scenes containing potentially mobile objects that are not actually moving (*e.g.*, scenes with many parked cars). The aim of this work is, precisely, to overcome this problem as only those objects that are moving at that precise moment will be labeled as dynamic. Another work that has a similar approach is StaticFusion [12], a dense RGB-D visual SLAM system where segmentation is performed by using the 3D reconstruction of the scene background as a way of propagating the temporal information about the static parts of the scene.

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 [13], MaskFusion [14], DynSLAM [15] and ClusterVO[16]. Their aim is to simultaneously estimate the poses of the camera and multiple dynamic objects. For that purpose, in MID-Fusion [13] and MaskFusion [14] sub-maps of each possible moving object are created and a joint estimation of both the objects and camera poses is carried out.

Most of the systems mentioned [13][14][15][16][10] involve deep learning methods, which in some cases cannot be currently implemented in real-time due to bottleneck imposed by 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, thus enabling its implementation in real time.

## III. DOT

### A. System Overview

Fig. 2 shows an overview of our proposal. The input to DOT are either RGB-D or stereo images at a certain video rate, and its output is a mask encoding the static and dynamic elements of the scene, which can be directly used by SLAM or odometry systems.

The first block (*Instance Segmentation*) corresponds to the CNN that segments out pixel-wise all the potentially dynamic objects. In our experiments, done using autonomous driving datasets, only cars were segmented as potentially moving. As it will be detailed later, since DOT tracks the mask from frame to frame, this operation does not need to be done at every frame.

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

The next block (*Is the object in motion?*) determines, using geometric criteria, whether the objects labeled as potentially 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.

Finally, DOT generates new masks from the estimations of the objects movement (*Mask Propagation*), so not every frame needs to be segmented by the network (see Fig. 3). Given the significant computational load of instance segmentation, this can be an relevant advantage of DOT compared to other state-of-the-art methods.

### B. Instance Segmentation

We use the COCO Instance Segmentation baseline model with Mask R-CNN R50-FPN 3x[17], [18] trained with Detectron2 [3] for the segmentation of all potentially movable instances that are present in an image. The output of the network has been modified to obtain in a single image all the segmentation masks. The image areas that are not classified into the potentially moving categories are given a 'background' label and are considered static in the subsequent blocks.The classes have been restricted to those considered as rigid object and potentially movable, hence excluding humans from the tracking. If other categories were needed, the network could be fine-tuned using these weights as a starting point or trained from scratch with its own dataset.

In order to consistently track the objects across multiple frames we have included a matching step between the masks computed by DOT and the ones provided by the net. New detections not matching any existing object are used to initialize new instances.

### C. Camera and Object Tracking

From the instance segmentation of the previous step, we aim to estimate the motion of the camera and the dynamic objects. Since the motion of the camera and the motion of the objects are coupled in the images, we make the estimation in a two-step process. First we find the pose of the camera as a relative transformation $\mathbf{T}_c \in \mathbf{SE}(3)$ and then we subtract it to estimate the object motion $\mathbf{T}_o \in \mathbf{SE}(3)$. Our optimization is related to the recent approaches of direct visual odometry and SLAM [6], which aim to find the motion that minimizes a photometric reprojection error.

**Optimization.** We do Gauss-Newton for both the camera pose and the subsequent object motion estimation

$$(\mathbf{J}^T \mathbf{\Sigma}_r^{-1} \mathbf{J})\mathbf{x} = -\mathbf{J}^T \mathbf{\Sigma}_r^{-1} \mathbf{r}, \tag{1}$$

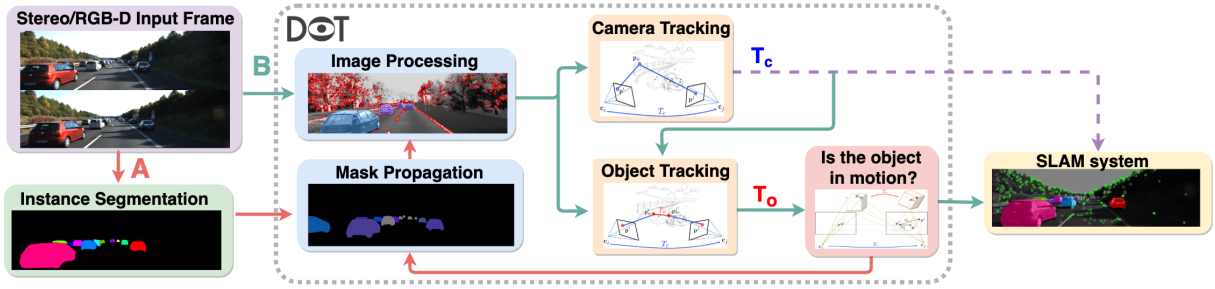


Fig. 2. **Overview of DOT**. Path A (red), shows the processing for frames that get a segmentation mask from the network. Path B (green), shows the processing for frames that will acquire a segmentation mask geometrically propagated by DOT.

where $\mathbf{J} \in \mathbb{R}^{n\times 6}$ contains the derivatives of the residual function (equations (3) and (5)) and $\mathbf{\Sigma}_r \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing the covariances of the photometric residuals $\mathbf{r} \in \mathbb{R}^n$. The Lie-algebras pose-increments $\widehat{\mathbf{x}}_{\mathfrak{se}(3)} \in \mathfrak{se}(3)$, with $\widehat{\cdot}_{\mathfrak{se}(3)}$ being the mapping operator from the vector to the matrix representation of the tangent space [19], are expressed as a vector $\mathbf{x} \in \mathbb{R}^6$. We update the transformations using left matrix multiplication and the exponential map operator $\exp(\cdot)$. Both are initialized with a constant velocity model and a multi-scale pyramid image to aid convergence.

**Camera tracking.** The camera motion is estimated using a sparse subset of high gradient pixels from static scene areas $\mathcal{P}$ and multi-view constraints [20], assuming that the camera calibration and points depths are known. The projection of a static point $\mathbf{p} \in \mathcal{P}$ from its pixel coordinates $\mathbf{p}^j$ in the reference frame $F_j$ to its corresponding coordinates $\mathbf{p}^i$ in the frame $F_i$ is as follows:

$$\mathbf{p}^i = \Pi(\mathbf{T}_c \Pi^{-1}(\mathbf{p}^j, z_j)), \tag{2}$$

where $\Pi$ and $\Pi^{-1}$ correspond to perspective projection and back-projection models, respectively, and $z_j$ is the depth of the point in the reference frame $F_j$.

The camera pose is optimized by minimizing the photometric reprojection error

$$\sum_{p\in\mathcal{P}} \left\| I_j(\mathbf{p}^j) - I_i(\Pi(\exp(\widehat{\mathbf{x}}_{\mathfrak{se}(3)})\mathbf{T}_c \Pi^{-1}(\mathbf{p}^j, z_j))) \right\|_\gamma, \tag{3}$$

which is computed as the sum of all intensity differences between points in their reference frame and their projection into the frame being tracked. We use the Huber norm $\gamma$.

**Object tracking.** Once $\mathbf{T_c}$ has been estimated, the pose of each potentially dynamic object can be estimated analogously by using a subset of image high gradient pixels $\mathcal{Q}_o$ belonging to such object. Modelling the potentially dynamic object as a solid with pose $\mathbf{T_o}$, the projection of each point $\widetilde{\mathbf{p}}$ in the frame $F_j$ to its coordinates in frame $F_i$ is:

$$\widetilde{\mathbf{p}}^i = \Pi(\mathbf{T}_c \mathbf{T}_o \Pi^{-1}(\widetilde{\mathbf{p}}^j, z_j)). \tag{4}$$

Analogously to equation 3, we estimate $\mathbf{T}_o$ by minimizing the following photometric reprojection error

$$\sum_{\widetilde{p}\in\mathcal{Q}} \left\| I_j(\widetilde{\mathbf{p}}^j) - I_i(\Pi(\mathbf{T}_c \exp(\widehat{\mathbf{x}}_{\mathfrak{se}(3)})\mathbf{T}_o \Pi^{-1}(\widetilde{\mathbf{p}}^j, z_j))) \right\|_\gamma. \tag{5}$$

**Algorithm 1** Dynamic Object Tracking

```
function OBJECT TRACKING(P, Q, O)
                                          ▷ P = static points
                                          ▷ Q = dynamic points
                                          ▷ O = set of objects
    mask ← ∅                              ▷ Dynamic mask to be computed

    {T_c , φ_c} ← track camera (P)        ▷ Camera Tracking
    if φ_c < th_φ then return ∅
    end if

    for object in O do                    ▷ Object Tracking
        if is visible (object , T_c) then
            {T_o , φ_o} ← track object (T_c , Q_o , mask)
            if φ_o < th_φ then break
            end if
            object ← outlier rejection (φ_o)
            mask ← update mask (object)
            mask ← is object moving? (object)
        end if
    end for

    return mask
end function
```

### *D. Tracking quality, outliers and occlusions*

Occlusions, changes in lighting conditions and segmentation errors have a significant effect in the accuracy of the objects and camera poses. As seen in algorithm 1, we developed several strategies that we apply after the object tracking step to reduce their impact.

**Tracking quality.** The appearance of dynamic objects changes significantly, increasing the tracking error. We used the Pearson's correlation coefficient $\phi_o \in [-1,1]$ to model appearance similarity. This metric reflects the degree of linear correlation between the reference intensities and their corresponding estimates, hence being invariant to gain and offset changes. Note that this can also be applied to camera tracking $\phi_c$, although changes in the background appearance are usually less pronounced.

**Outlier rejection.** Commonly, outliers are rejected using an absolute threshold for the photometric error. More sophisticated works [6] adapt it according to the median residual, the motion blur or the lighting changes. As shown in Fig. 4, we propose to set a threshold relative to the linear relation between intensities, so the errors are independent to photometric changes in the image.

**Occlusions.** The dynamic objects might occlude each other.

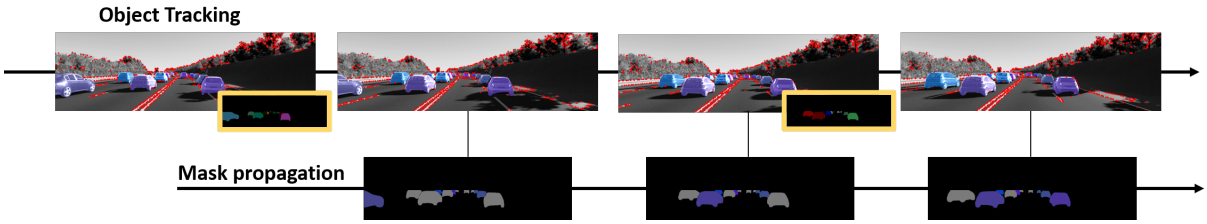


Fig. 3. **Sample results**. The upper row shows features and object tracks. Note how the network segmentations (framed in yellow) are not necessary in all frames. The lower row shows the masks propagated by DOT that encode the motion classification: in motion (color), static (black) and yet not observed (gray).

Removing the occluded parts as outliers was not sufficient in our experiments. We implemented a strategy consisting of tracking the objects from the closest to the farthest, updating their respective masks sequentially. In this manner, we update in every iteration the points of the farther objects that have been occluded by closer ones.

### *E. Is the object in motion?*

This 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, to be used by SLAM or odometry systems, are the masks that store the areas of the image occupied by dynamic objects and whether they are in motion or not. The masks are obtained by projecting the pixels of each object into the new frame using $\mathbf{T}_c$ and $\mathbf{T}_o$ estimated in the previous step.

Observing the object motion directly in $\mathbf{T}_o$ generates, due to the propagated image noise, difficulties in establishing absolute thresholds that determine whether an object is in motion. In this work we chose to observe the motion of the objects using 2D image measurements. We denote our metric as *dynamic disparity*, being the distance in pixels between the projection of the point as if it were static $\mathbf{p}^i$ and its actual projection $\widetilde{\mathbf{p}}^i$. For each object we compute the median $d_d = \text{med}\{\,||\mathbf{p}^i, \widetilde{\mathbf{p}}^i||, \forall \widetilde{\mathbf{p}} \in \mathcal{Q}\}$ of the dynamic disparities of its points $\widetilde{\mathbf{p}} \in \mathcal{Q}$:

The 3D motion of a point produces different image motions depending on 1) its image coordinates, 2) its depth, and 3) the relative angle between the directions of the object and the camera motions.

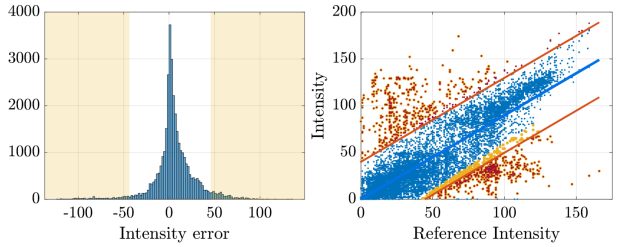


Fig. 4. **Outlier rejection**. **Left:** histogram of photometric errors for an object. The shaded area corresponds to the points removed with a constant threshold. **Right:** Linear relation between intensities. Note the different points labeled as outliers by absolute (yellow) and relative (red) thresholds due to the changing photometry.

Fig. 5. **Disparity vs Entropy**. Comparison of the dynamic disparities produced by different objects in motion. Note how observations with high entropy values (brighter red) produce larger shifts of image pixels.

From the non-linear pose optimization (see eq. (1)) we can derive the uncertainty in the estimation of the motion of the object $\mathbf{\Sigma}_x = (\mathbf{J}^T \mathbf{\Sigma}_r^{-1} \mathbf{J})^{-1}$. Assuming a k-dimensional Gaussian distribution, its differential entropy is $H(\mathbf{x}_o) = \frac{1}{2}\log((2\pi e)^k |\mathbf{\Sigma}_{x_o}|)$.

The differential entropy can be seen as the pose uncertainty derived from the photometric residuals minimization. In other words, motion observations of high entropy will result in large shifts of image pixels (see Fig. 5). Conversely, observations with low entropy will produce small image disparities.

Based on this, the algorithm for classifying the movement of objects works as follows. We compare dynamic disparities against a variable threshold $\Delta d = f(H(x))$ that grows smoothly with the entropy. We label as "in motion" all those objects whose dynamic disparity exceeds this threshold ($d_d > \Delta d$). For every value below an entropy threshold $H_{min}$ we assume the object motion cannot be observed. Therefore, labeling an object as static requires that the motion is observable ($H(x) > H_{min}$) and that the median of the dynamic disparity is less than the variable threshold ($d_d < \Delta d$).

While selecting the optimal functional formulation would require further study, this expression meets the requirements and has shown good results in this work (see section IV). Fig. 3 is an example of the mask propagated by DOT. Objects labeled as "in motion" are represented in colour, while those labeled as "static" disappear in black. The cars represented in gray are those which cannot be determined as being static neither dynamic.

### *F. Mask propagation*

To relate instances from different frames for the same 3D object, DOT overlaps the mask produced by the neural network and the one propagated by projecting the pixels from the previous frame with multiview equations and the calculated camera and object poses.

**State propagation.** Relating new semantic instances to pre-existing objects allows us to predict their motion (which is critical for fast moving objects). In addition, the state of the motion can be maintained in the case of an object moving to a position where the motion is not observable (see Section III-C).

**Independent segmentation.** Our proposal allows the propagation of semantic segmentation masks from an initial seed over time and space, eliminating the need to segment each frame. Running the neural network at a lower frequency facilitates real-time object tracking on low-end platforms. As an additional benefit, DOT is able to fill the gaps in which the network temporarily loses instantiation of an object between consecutive images.

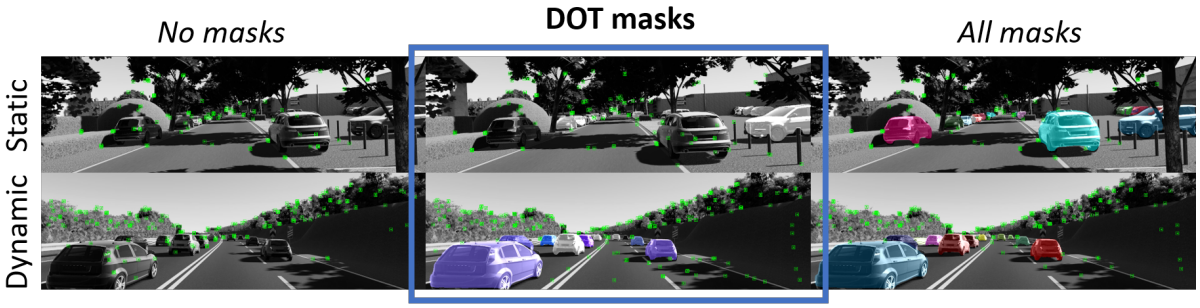


Fig. 6. **Content adaptation**. **Left**: *No masks*. **Centre**: *DOT masks*. **Right**: *All masks*. The top row shows a static scene in which *All masks* discards all points in static objects. In the bottom row, *No masks* tracks features on moving objects, which may cause failures. DOT successfully identifies parked cars as static and moving ones as dynamic.

## IV. Experimental results

**Baselines.** Our experiments estimate the camera trajectory using ORB-SLAM2 [1] in three different configurations:

*No masks*: ORB-SLAM2 is run using the authors' implementation on unmodified images. Image features can be initialized in the whole image (including areas belonging to moving objects).

*DOT masks*: in addition to the images, ORB-SLAM2 receives as input the masks containing potentially dynamic objects currently in motion. We modified ORB-SLAM2 so as not to extract points from such moving objects.

*All masks*: all the masks obtained by the segmentation network are applied in ORB-SLAM2. Hence, all potentially dynamic objects are removed without checking if they are actually moving.

**Sequence subsets.** We evaluated these configurations for three subsets from the KITTI Vision Benchmark Suite [2]. We used data from Virtual KITTI [21], [22], a synthetic dataset composed of 5 sequences virtually cloned from KITTI [2], KITTI *Odometry*, a predefined subset of sequences specially designed for the development and evaluation of visual odometry systems, as well as a selection of sequences from the KITTI *raw* section with a high number of moving objects [23]. We used RGB-D ORB-SLAM2 in Virtual KITTI, as this dataset provides synthetic depth images. In other sets, we used stereo ORB-SLAM2. The ground truth for the real sequences is given by a GPS localization system.

**Evaluation metrics.** In order to account for non-deterministic effects, we run each configuration $10\times$ per sequence and report median values. All the experiments were run in a laptop with an Intel Core i5 processor and 8GB of RAM memory. The absolute trajectory error (ATE) [24] is the root-mean squared error (RMSE) of the estimated position of all frames with respect to the GPS ground truth after both trajectories have been aligned. For an easier comparison, the average of the errors is normalized by the value obtained with DOT on each sequence $\bar{\varepsilon}_{norm} = \frac{1}{n}\sum_{i=0}^{n} \frac{\varepsilon_i}{\varepsilon_{DOT}}$.

The right columns in Table I show the ATE normalized by the best ATE in each sequence among the three configurations. Thus, 1 identifies the best result, while values $>1$ are indicative of poorer performance. The color scale indicates the relative position between the best result (green) and the worst (red).

**Tracking accuracy.** Accoding to ATE in Table I, for V-KITTI sequences, DOT improves performance by 92.6% and 37.8% with respect to the *No masks* and *All masks* configurations, respectively. In addition, DOT scores best for 3 of the 5 sequences evaluated.

TABLE I

DOT against baselines (*No masks* and *All masks*).

| | ATE [m] | | | ATE/ATE$_{best}$ | | |
|---|---|---|---|---|---|---|
| V-KITTI | 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% | | | |
| KITTI *Odometry* | 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% | | | |
| KITTI *Raw* | 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 % | | | |

For the 11 trajectories of KITTI *Odometry*, DOT is 12.7% and 30.3% more accurate than *No masks* and *All masks*, respectively. Compared to V-KITTI, this group of sequences contains less dynamic elements, so the use of masks is even detrimental. According to the dataset specifications, the ground truth camera poses collected by the GPS are accurate to within 10 cm. 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 the camera motion.

The differences between sequences and methods are more evident in the last section of Table I, KITTI *Raw*, characterized by an abundance of moving objects. Overall, DOT achieves improvements of 142.3% in ATE over *No masks* and 15.9 % over *All masks*.

Note how discarding dynamic objects in sequence 1003-0047 significantly reduces the tracking errors. The sequences 0926-0009, 0929-0004 and 1003-0047 were cloned to generate the V-KITTI synthetic sequences (1, 18 and 20). As expected, since the scenes contents are identical, so is the qualitative analysis of the results.

The color scale in Table I confirms that DOT approaches the best solution even when it is not the most accurate (green). So, while the use of masks may be convenient, the accuracy is significantly improved if only the objects actually in motion are removed. These results demonstrate that DOT consistently achieves good performance both for static and dynamic scenes.

**Adaptation to scene content.** Fig. 6 illustrates two situations that may affect SLAM accuracy. The high dynamism of the scene in the lower row (V-KITTI 20), with all vehicles in motion, violates the rigidity assumption of ORB-SLAM2, and makes the system fail. Similarly, moving objects in VKITI 18 causes tracking failure of ORB-SLAM2 in 6 out of 10 trials (only 56% of the trajectory could be estimated in those cases). The upper row shows an urban scene with cars parked on both sides (V-KITTI 01). Contrary to the previous case, the worst results were obtained with all the segmentation masks, since many points with reliable information are removed for tracking. ATE results for this sequence show that extracting points from a larger area results in more accurate estimated trajectories.

This analysis shows that, whereas using dynamic object masks is beneficial, using them without verifying if the object is actually in motion leads to information losses, especially in scenes with many vehicles. DOT achieves a trade-off between those two opposing scenarios by estimating the motion state of the objects, significantly enhancing robustness and accuracy.

**Loop closure.** The loop closure module of ORB-SLAM2 reduces the drift and inaccuracies due to dynamic objects or to the removal of parked vehicles. We have observed that ORB-SLAM2 running with *DOT masks* is able to close the loop 6 out of 10 runs in sequence 9 of KITTI *Odometry*, while none was closed with *All masks*. This explains a part of the variability in the results and appears as an additional advantage afforded by DOT.

**Segmentation errors.** Compared to other approaches, DOT reduces segmentation errors. Neural networks sometimes mislabel static objects (*e.g.*, traffic signs or buildings) as dynamic, but DOT corrects this error by re-tagging the object as static (see Fig. 7).Also, when the network does not fire in one of the sequence frames, DOT is able to fill the gap by propagating the object mask.

**Mask propagation.** As explained in section III-F, our approach allows reducing the frequency of network segmentation by propagating pre-existing masks in the intermediate frames. Figure 8 shows the number of correctly labeled pixels minus mislabeled ones (ground truth in black) on every frame of V-KITTI when DOT uses 100% of Detectron2 segmentations (red), 50% (blue), 33% (yellow) and 25% (green). Note how the masks stay accurate when being propagated except when tracking failures occur or a moving object enters the scene between segmentations (see also intersection over union on V-KITTI in Table II). This may be exploited to reduce computation time, which can be a practical advantage, specially for high frequency image streams.

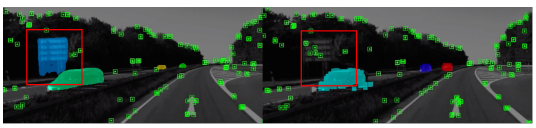

Fig. 7. **Segmentation error**. Comparison between *All masks* and *DOT masks*. Notice that an incorrect segmentation from Detectron2 (the sign in the red square is assigned a car label) is correctly classified as static by DOT.

TABLE II

IoU IN THE V-KITTI DATASET FOR DIFFERENT SEGMENTATION RATES.

| Rate | Seq01 | Seq02 | Seq06 | Seq18 | Seq20 |
|---|---|---|---|---|---|
| 1.0 | 0.88 | 0.88 | 0.84 | 0.90 | 0.89 |
| 0.5 | 0.74 | 0.83 | 0.67 | 0.85 | 0.84 |
| 0.33 | 0.72 | 0.80 | 0.60 | 0.85 | 0.81 |
| 0.25 | 0.69 | 0.78 | 0.55 | 0.84 | 0.81 |

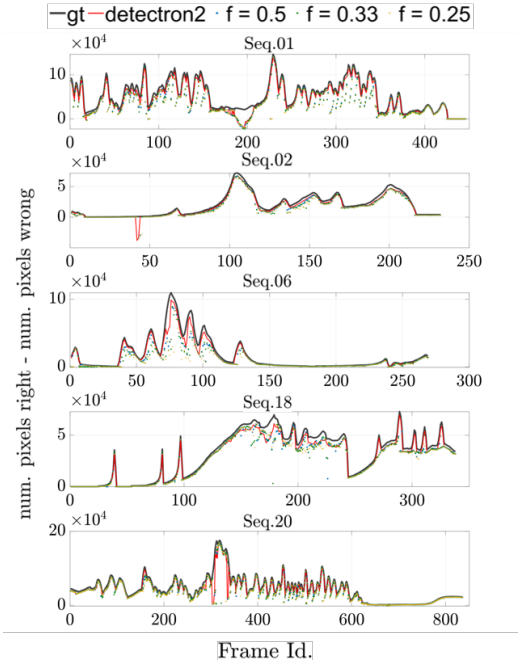


Fig. 8. **Mask propagation**. Number of correctly labeled pixels minus mislabeled ones respect to the ground truth (black) in V-KITTI, when DOT uses the segmentation network all frames (red), 50% (blue), 33% (yellow) and 25% (green) of them.

## V. CONCLUSIONS

DOT is a novel front-end algorithm for SLAM systems that robustly detects and tracks moving objects by combining instance segmentation and multi-view geometry equations. Our evaluation with ORB-SLAM2 in three public datasets for autonomous driving research [2], [21], [22] demonstrates that DOT-generated object motion information allows us to segment the dynamic content, significantly improving its robustness and accuracy.

The independence of DOT from SLAM 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, DOT mask tracking reduces the rate at which segmentation should be done, reducing the computational needs with respect to the state of the art.

## REFERENCES


[1] 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, p. 1255–1262, 2017. [Online]. Available: http://dx.doi.org/10.1109/TRO.2017.2705103

[2] 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.

[3] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[4] 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*, vol. 32, no. 6, pp. 1309–1332, 2016.

[5] 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.

[6] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[7] 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.

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

[9] 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.

[10] B. Bescós, J. M. Fácil, J. Civera, and J. Neira, "DynSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes," *CoRR*, vol. abs/1806.05620, 2018. [Online]. Available: http://arxiv.org/abs/1806.05620

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

[12] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments," in *2018 ICRA*. IEEE.

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

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

[15] I. A. Barsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust dense mapping for large-scale dynamic environments," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2018.8462974

[16] 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.

[17] 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.

[18] 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].

[19] H. Strasdat, "Local accuracy and global consistency for efficient visual slam," Ph.D. dissertation, Department of Computing, Imperial College London, 2012.

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

[21] 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*, 2016, pp. 4340–4349.

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

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

[24] 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*. IEEE, 2012, pp. 573–580.