SLAMANTIC - Leveraging Semantics to Improve VSLAM in Dynamic Environments

Matthias Schörghuber[1] Daniel Steininger[1] Yohann Cabon[2] Martin Humenberger[2]
Margrit Gelautz[3]
[1]Austrian Institute of Technology [2]NAVER LABS Europe [3]Vienna University of Technology
firstname.lastname@{[1]ait.ac.at,[2]naverlabs.com,[3]tuwien.ac.at}

Abstract

In this paper, we tackle the challenge for VSLAM of handling non-static environments. We propose to include semantic information obtained by deep learning methods in the traditional geometric pipeline. Specifically, we compute a confidence measure for each map point as a function of its semantic class (car, person, building, etc.) and its detection consistency over time. The confidence is then applied to guide the usage of each point in the mapping and localization stage. Points with high confidence are used to verify points with low confidence in order to select the final set of points for pose computation and mapping. Furthermore, we can handle map points whose state may change between static and dynamic (a car can be parked or in motion). Evaluating our method on public datasets, we show that it can successfully solve challenging situations in dynamic environments which cause state-of-the-art baseline VSLAM algorithms to fail and that it maintains performance on static scenes. Code is available at github.com/mthz/slamantic

1. Introduction

The accuracy of estimated camera poses in a visual simultaneous localization and mapping (VSLAM) algorithm relies on valid geometric representations of the observed environment. During operation, a VSLAM algorithm extends its map (3D scene model) by adding new 3D measurements which are generated by estimating the depth of image points or areas captured from different viewpoints at different points in time. If an object has moved in between these points in time, the triangulation of image points representing the object does not yield to the correct distance from the camera and therefore does not lead to the correct camera pose. Hence, reliable results can usually only be achieved in static and distinctive (in terms of texture and structure) environments. This shortcoming is the main challenge for

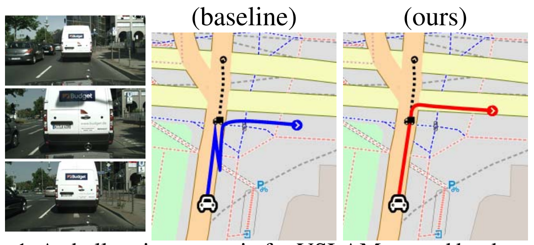

Figure 1. A challenging scenario for VSLAM caused by dynamics: The car with the observing camera is approaching a truck which has stopped at a crossing (left). When the truck gradually starts moving, the baseline VSLAM (middle) fails because it wrongly estimates a backward motion. Using semantics, our proposed approach (right) is able to cope with such situations.

VSLAM algorithms in real-world scenarios.

In order to cope with dynamic scenes, many algorithms assume that image points originating from static scene elements are dominant. Such approaches do not treat image points from dynamic elements separately, but assume that these points violate the geometric model and can therefore be classified as outliers in the pose estimation or bundle adjustment step and, hence, excluded from further computation.

In the presence of minor dynamic objects such as a single car or a person within texture-rich environments, the dominant static scene assumption combined with RANSAC-based outlier detection is often sufficient. Traditional approaches start to fail when dynamic 3D-points become dominant. Particular problems arise in scenes where objects are first considered as static, which allows 3D-points from these objects to become an integral part of the map, and becoming gradually dynamic afterwards. An illustrative example of such a scenario is shown in Figure 1, where a car is approaching a truck that has stopped at an intersection. When the truck starts moving again, the rear car falsely interprets this as backward motion.

Advances in scene understanding, especially semantic segmentation [13], enable new strategies for overcoming this problem. A popular one is to use semantic information to mask out classes which are assumed to be dynamic, such as persons or cars [3, 14, 2]. However, parked cars are a valuable source, in some cases even the primary source, of high-quality 3D-points for local odometry estimation. Furthermore, semantic information may not be available at camera framerate or not always provide accurate data. If an image region is assigned to a wrong semantic class, it could be unnecessarily excluded from the pose estimation which can be critical in sparsely textured environments. The current solution to this problem is to explicitly detect motion [25] within the scene using, *e.g.*, optical flow [1, 21, 32].

Contrarily, we propose a novel way of tackling this challenge by integrating semantic information into geometric, feature-based VSLAM methods without the need for motion detection. We combine semantic class label assignments with map point observation consistency in order to estimate a reliability measure for each 3D-point and utilize it during the pose estimation and map optimization steps. Furthermore, our method is able to handle hybrid cases such as cars and does not require the availability of semantic labels for each individual frame.

We evaluate our approach on a representative selection of difficult scenarios identified in public datasets. We show that our method is able to cope with dynamic environments and particularly challenging situations that cannot be handled by the current state-of-the-art, while maintaining performance in predominantly static environments. We implemented our approach on top of ORB-SLAM2 [20], and make our available at *github.com/mthz/slamantic*.

2. Related Work

VSLAM algorithms can be seen as a mature research field, since the technology has proven itself in numerous real-world applications, such as mobile robotics. Traditional VSLAM methods are either based on feature matching [15, 20], image alignment [9], or patch alignment [10, 8]. To reduce drift and to directly recover scale, some methods integrate inertial measurements [4, 16], [23, 22] or utilize learning-based techniques [29, 30, 18]. A more comprehensive overview about the past, present, and future of SLAM is given by Cadena *et al.* [6].

While impressive results have been achieved in controlled environments with slowly moving cameras, more research is required to increase robustness in less constrained environments. Saptura *et al.* [25] present a thorough survey of VSLAM in dynamic environments and explain in detail multiple ways of addressing this problem. A popular method of dealing with scene dynamics in VSLAM is to detect and exclude motion within an image. This can be done with optical flow [1, 21, 32], optical flow + semantic labeling [33], deep learning [31] or background subtraction methods [28, 17]. Other methods rely on multiple independent cameras [35] or RGBD data [19]. Contrarily, our approach does not need explicit motion detection within the image itself, but uses semantic information to compute the confidence of a 3D-point regarding its dynamics. In the following, we highlight the most important works which we consider relevant for our method.

DynaSLAM [3] first uses instance segmentation to mask out potential dynamic objects and then performs a geometric motion detection step on the remaining static scene elements. This allows to cope with situations where classes that are considered to be static are actually moving (*e.g.* a book which is carried). In contrast, our approach is motivated by the inverse case of using potentially dynamic classes while actually being static (*e.g.* a parked car). This allows us to maximize the area of the image being used for VSLAM. In particular, we do not need an additional motion detection step but we are able to leverage the implicit geometric verification of VSLAM. Contrary to DynaSLAM, our approach adds only a couple of milliseconds to the overall processing time.

In the work of Kaneko *et al.* [14], semantic classes are used to mask-out features which are labeled as sky or car. Their main motivation is to better distribute features within scenes which are static and rich in visual information. The method relies on perfect semantic labeling of the input image and was tested only on synthetic data.

Barnes *et al.* [2] show a method for robust monocular visual odometry in urban environments. They compute a map without dynamic objects using multiple scene traversals. This static-only map is used to train a convolutional neural network (CNN) which infers a mask that is used to differentiate between long-term static and dynamic elements. While this work presents a very interesting approach to handle dynamic scenes with machine learning, it requires multiple 3D mapping runs in the target environment and it cannot distinguish between, *e.g.*, parked and moving cars.

Rosen *et al.* [24] proposes a bayesian filter with a survival time prior for each 3D-point. They suggest using semantics to select an empirically characterized prior for a particular class. This is comparable to our approach of defining a class-specific dynamics factor. While their main goal is to estimate a long-term probability for a 3D-point, our approach aims to improve online VSLAM and the consideration of state changes during observation.

To the best of our knowledge, none of the existing methods is able to handle hybrid cases without explicit motion detection, which is one of the main contributions of our approach.

3. Semantic VSLAM

We aim to improve the performance of feature-based monocular or stereo VSLAM in dynamic environments. In a feature-based VSLAM such as [20], the world is represented as a graph consisting of frames and 3D-points as nodes with their observations as edges. This graph is optimized by minimizing the reprojection errors of the observations. In the case of erroneous correspondences or scene structure changes caused by dynamic elements, the reprojection error increases. If the error is above a certain threshold, the 3D-point is labeled as an outlier and consequently removed.

More precisely, a central assumption is that the more often a 3D-point passes the optimization and implicit geometric verification process, the higher is the probability that this point is reliable. Thus, too many violations of this assumption will very likely cause VSLAM to fail. We suggest solving this problem by introducing a confidence measure $df \in \mathbb{R}[0,1]$ for each 3D-point which reflects the uncertainty caused by dynamics in the scene and we call it the dynamics factor. It will be used during pose estimation to differentiate static from potentially dynamic and dynamic 3D-points under consideration of their semantic class assignment.

As mentioned above, the number of observations over time of the same 3D-point is a powerful measure of the reliability of the point. Instead of just counting the observations, we suggest using them to adjust the confidence of a 3D-point being static depending on the associated semantic class. Intuitively, classes such as *road* or *building* are considered to be more likely static and need fewer observations than 3D-points from *people* or classes such as *cars* which are considered hybrid.
The dynamics factor df is defined as

$$df = \max(df_{obs} + s \cdot df_{label}, df_{label}, 0) \qquad (1)$$

and consists of three components. The first is the main component which depends on the number of observations (df_{obs}) and semantic (df_{label}) information associated with a 3D-point. The scalar s allows the scaling of the influence of the semantic information. By applying the maximum function, term df_{label} imposes a semantic class dependent lower-bound to cope with hybrid classes. The third component ensures that df does not become negative.

Observation Term The term df_{obs} includes the observation information of a 3D-point. It lowers the value df with an increasing number of observations which indicates a higher probability of a point being static. We model this with

$$df_{obs}(N_{obs}) = -\frac{N_{obs}(N_{obs}-1)}{k} + d, \qquad (2)$$

where N_{obs} is the number of observations, $k \in \mathbb{R} > 0$ is an arbitrary scalar defining the declination rate, and d is set to the desired value at $N_{obs} = 1$. Because the first observations are the most critical with the highest uncertainty, we select a quadratic function to gradually adjust the dynamics factor df while enabling a fast convergence to its lower-bounds. The line "no label" in figure 2 shows the curve of the function df_{obs} for k=20 and d=0.5. A low k would delay the use of the 3D-point while a high value would result in its immediate use. A good choice lies in between because a 3D point should become part of the potential-dynamic group while it is still observed. We set the declination parameter k of the observation term df_{obs} to 20 to achieve a saturation at around 4 observations. All mentioned parameters were kept constant throughout all experiments.

Semantic Term The term df_{label} integrates the semantic class information into the dynamics factor df. The semantic information consists of a set of labels such as *person*, *car*, *road* or *building*. Table 1 shows the label set of the Cityscapes [7] and Virtual-KITTI [11] datasets.

While the dynamics assignment of classes such as *person* and *building* appear clear, hybrid classes exist which can be both static or dynamic. For example, in the case of autonomous driving and a front-facing camera, moving cars are an erroneous source whereas parked cars are often the primary source of high-quality 3D-points. Therefore a binary classification between static and dynamic based on the semantic class, which is used in mask-out approaches such as [2, 14, 3], is not sufficient, which motivates our hybrid approach. For each semantic class, we specify the assumed dynamics using the factor $ld_L \in \mathbb{R}[-1,1]$ (label dynamics factor). It describes the likelihood of a label to originate from a static ($ld_L < 0$) or dynamic ($ld_L > 0$) object. Our label dynamics factor assignments are also shown in Table 1. The assignment is based on the nature of the class since it roughly categorizes available labels. It was kept constant through all experiments.

A further aspect which has to be considered is that the semantic label may not always be correctly detected. Thus, we introduce the label consistency $lc \in \mathbb{R}[0,1]$ for each 3D-point as

$$lc = \begin{cases} lp_L & N_{obs} = 1 \\ \frac{N_L}{N_{obs}} & N_{obs} \geq 2 \end{cases}, \qquad (3)$$

where lp_L is the label probability obtained from the semantic labeling which we use as initialization value for a 3D-point with a single observation. In the case of multiple observations, we define the consistency being the ratio of the number of observations N_L in which the 3D-point is assigned to a label and the number of total observations N_{obs}. If the ratio is below 50%, we initiate a re-labeling which selects the label with the most occurrences.
This leads to the final semantic term

$$df_{label} = ld_L \cdot lc. \qquad (4)$$

Summarizing, in the first term of Equation 1 the semantic term df_{label} adjusts the base function df_{obs} according to the label dynamics factor ld_L and its label consistency lc. To allow a distinction between static and hybrid classes, a semantic class dependent lower-bound is imposed with df_{label} in the second term of Equation 1.

3D-Points Grouping By using the dynamics factor df, we aim to differentiate 3D-points based on their estimated level of dynamics. A 3D-point originating from a pedestrian is considered a non-reliable source of geometric scene information. Contrarily, a point corresponding to a building or pole will not move during observation and is thus assumed to be reliable. 3D-points from hybrid classes, however, can be either static or dynamic and might change their state during observation. This motivates to split the 3D-points into three groups based on their computed dynamics factor df:

$$\begin{aligned} df \leq 0.25 & \quad static\ (S), \\ 0.25 < df \leq 0.5 & \quad static\text{-}dynamic\ (SD), \\ df > 0.5 & \quad dynamic\ (D). \end{aligned} \tag{5}$$

Figure 2 shows examples of the dynamics factor df computation on a static ($house$, ld_L=-1), hybrid (car, ld_L=0.5) and dynamic ($person$, ld_L=1) class with a low (L_{lc}) and high (H_{lc}) label consistency lc. In this example, the semantic labeling sequence for H_{lc} labeled lines is L, L, L, L, L with L denoting a correct label assignment. The low confident lines L_{lc} are assigned a labeling sequence which includes occurrences of a wrong label O and are set to L, O, L, O, L. The example uses a declination factor of k=20, the value for $df_{obs}(1)$ is set to 0.5 (d=0.5) and the semantic influence factor to s=0.5. The group definition from Equation 5 is visualized as background color in Figure 2 ($green$=$static$, $blue$=$static$-$dynamic$, red=$dynamic$).

Without a specific label assignment, df starts at the upper-bound of the $static$-$dynamic$ group and with an increasing number of observations it may change to the $static$ group. In case of a 3D-point of label $house$ with its corresponding label dynamics factor (ld_L=-1), the 3D-point is initially part of the group $static$ but it might temporally switch to other groups due to inconsistent label assignments. Contrarily, a 3D-point of label $person$ with a high dynamic assumption (ld_L=1) remains within the $dynamic$ group. In hybrid cases such as car, the 3D-points are first assumed to be $dynamic$ and change their group assignment to $static$-$dynamic$ with an increasing number of consistent observations. The lower-bound imposed by df_{label} prevents 3D-points of this group from being classified as $static$.

Dynamics Factor Application The three groups are used for pose estimation performed during the tracking stage of VSLAM as indicated in Figure 3. In the first step, the matches between 2D image features and 3D-points are split into $static$, $static$-$dynamic$ and $dynamic$, according to their dynamics factor df. The $static$ group is used to estimate an initial pose. With this pose, the matches of the $static$-$dynamic$ group are validated based on the reprojection error. The final pose is computed using matches between pixel locations from the group $static$ and valid 3D-points from the group $static$-$dynamic$.

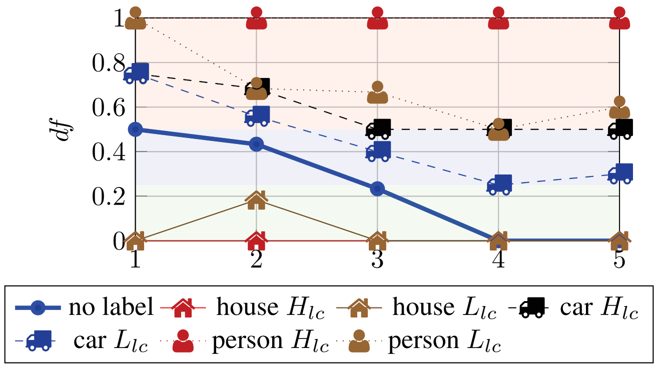

Figure 2. Examples of dynamics factor (df) computation: no label, static (house), hybrid (car), and dynamic (person). H_{lc} means high and L_{lc} means low label consistency lc. The background color refers to the group definitions of $static$ (green), $static$-$dynamic$ (blue) and $dynamic$ (red). Persons remain in the $dynamic$ group. Cars move to and remain in the $static$-$dynamic$ group with an increasing number of observation. 3D-points from houses require fewer observations to become part of the $static$ group.

Since the dynamics factor df has a lower-bound on the semantic term df_{label}, a hybrid class such as car remains in the $static$-$dynamic$ (or $dynamic$) group. The $static$-$dynamic$ group allows to cope with situations where 3D-points from hybrid classes have become a valid geometric part of the map ($e.g.$ parked car) but change their state within the mission ($e.g.$ car starts moving). Furthermore, this allows accurate pose estimation even in cases with fewer matches from static than dynamic 3D-points, where RANSAC-based algorithms fail.

If a frame is selected for keyframe computation (mapping), all $dynamic$ matches are additionally validated to allow the mapping algorithm the generation of new observations and therefore update of the dynamics factor df.

If too few matches from static 3D-points are available for computing a pose, or if the pose estimation does not succeed, the algorithm tries to estimate the pose with the $static$ and $static$-$dynamic$ matches without validation. If the pose estimation still does not succeed, the pose is estimated without considering the dynamics factor.

4. Experimental Evaluation

Datasets VSLAM algorithms are often evaluated on a selection of well-known datasets such as KITTI [12], TUM-

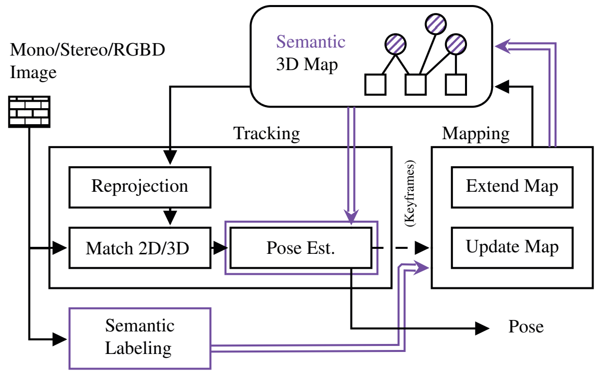

Figure 3. Schematic overview of the proposed integration of semantic information (purple) into a feature-based VSLAM.

RGBD [26] or EuRoC [5]. However, since these datasets and benchmarks were designed to evaluate traditional approaches and were recorded in static or low-dynamic environments only, they rarely cover scenarios which violate fundamental assumptions of traditional VSLAM. Hence there is limited data with ground truth available that contains test cases which can be used for quantitative evaluation of our semantic VSLAM approach. For example, the well-known KITTI-Odometry benchmark dataset only contains very few sequences with persons or moving cars which therefore are not suited to evaluate our approach. Only TUM-RGBD contains a few explicit dynamic scenes, which we use for comparison with the state-of-the-art in Section 4.3. As a consequence, we selected the Cityscapes [7] dataset as our primary dataset for evaluation in the target domain because it contains complex dynamic scenes.

We started by using the synthetic dataset Virtual-KITTI [11] (VKITTI[1]) to test concept and implementation. For real-world experiments, we provide a qualitative analysis using hand-picked short sequences of the Cityscapes dataset, where traditional VSLAM algorithms completely fail and a quantitative evaluation on the full Cityscapes sequence recorded in Frankfurt.

In order to compare our method with the related work, on the one hand we use the dynamic scenes of the TUM-RGBD dataset, and on the other hand we use the primarily static environment of the KITTI dataset.

Semantic Labeling The goal of semantic labeling is to assign a class label for each pixel of an input image. Our semantic labeling module is based on the implementation of Yu *et al*. [34]. Their concept of Deep Layer Aggregation includes an efficient scale space integration, while simultaneously reducing the number of network parameters. We applied the provided *dla-34* model with a down-sampling rate of 2, trained and validated it on the 3475 densely labeled images of Cityscapes. The model distinguishes between 19 semantic labels (Table 1) and achieves a mean-intersection-over-union (mIoU) score of 75.1 on the Cityscapes validation data [34]. The algorithm performs at an average processing speed of 8.9 fps on a single NVIDIA GTX 1080 Ti. Note that experiments on the TUM-RGBD dataset were performed using MaskRCNN [13] for semantic segmentation because our *dla-34*-based model is not trained for indoor scenes.

Label	ld_L
Road, Building, Traffic Sign, Traffic Light, Pole, Guard Rail^V, Wall^C, Fence^C, Sidewalk^C	-0.5
Terrain, Vegetation, Tree^V	-0.2
Sky, Void^C, Undefined^V, Misc^V	0
Truck, Car, Van^V, Bus^C, Train^C, Motorcycle^C, Bicycle^C	0.5
Person^C, Rider^C	1.0

Table 1. Label dynamics factor (ld_L) assignment to the semantic classes of the Virtual-KITTI [11] and Cityscapes [7] datasets. ^{C}Cityscapes only. ^{V}Virtual-KITTI only.

Figure 6 shows qualitative semantic labeling results on sequence B and D. While sequence B depicts a high labeling accuracy and is representative for most frames of the Cityscapes dataset, D shows an extreme failure case. This is caused by the advertisements painted on the tram. To cope with these cases, our method does not rely on single label assignments but utilizes a consistency value which is updated over time (Equation 3) as well as the observation information of 3D-points (Equation 2). Additionally, we normalize the labeling scores to generate a probability map for initialization (lp_L in Equation 3).

VSLAM Methods For easy comparison with the state of the art, we implemented our method on top of ORB-SLAM2 [20]. All our evaluations are conducted using a stereo- or RGBD-VSLAM setup. However, our semantics-based approach can be applied to monocular input data as well. In our experiments the semantic labeling is performed on the left camera image only.

4.1. Virtual-KITTI

We selected the scenes 18 and 20 from the synthetic dataset Virtual-KITTI [11] representing interesting examples of our target domain of dynamic environments in autonomous driving, namely road scenes with dense traffic. Figure 4 shows exemplary frames of the sequences. The unmodified VSLAM (baseline) algorithm (ORB-SLAM2) was able to compute a pose for all frames of the sequences and does not indicate an erroneous behaviour. However, the comparison with the ground truth reveals a significant drift. Figure 5 shows a bird's-eye view of the sequences compar-

[1]https://europe.naverlabs.com/research/computer-vision/proxy-virtual-worlds/

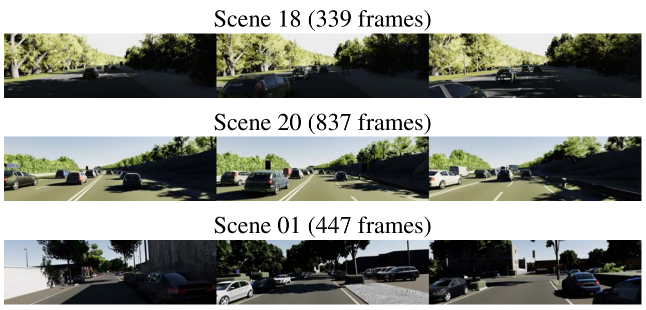

Figure 4. Exemplary frames of scenes in variation “clone” from the Virtual-KITTI [11] dataset. Scene 18 and 20: dense highway traffic. Scene 01: mainly static environment.

ing ground truth (*gt*), baseline VSLAM (*b*) and ours (*df*). Especially in Scene 20, the difference between the ground truth and the baseline VSLAM is visible while our proposed method *df* aligns closely to the ground truth.

Table 2 shows the results using the metrics of the KITTI odometry benchmark tools [12] and absolute trajectory error (ATE) from TUM-RGBD benchmark tool [27]. The figures confirm the plots in Figure 5. For Scenes 18 and 20 we achieved an improvement on the translational $trans.err.$ and rotational $rot.err.$ with the applied dynamics factor *df*. The scenes provide only minor rotational movements, hence the generally low rotational error. Since ground truth semantic labels are available, we compute a lower-bound for our approach (m) by masking-out all image areas labeled as car (because these are the only dynamic areas in Scenes 18 and 20). The result confirms our assumption that pose estimation is more likely to fail in the presence of dynamic objects.

We also evaluated our approach on Scene 01, which mainly consists of static objects except for a few non-dominant oncoming and crossing cars, hence the low error in Table 2. In Scene 01, one of the primary sources for good and valid 3D-points are parked cars. This observation is confirmed by the increased translational error associated with the VSLAM configuration with masked-out cars (m) compared to the baseline (b). Since only negligible dynamic elements are present, we do not expect a major difference between the baseline VSLAM (b) and applied dynamics factor (*df*). Because 3D-points from cars require more observations to become valid and reliable 3D-points, we expect a slightly worse performance on the error metrics in this scene. The results in Table 2 support this expectation, which shows that our method does not significantly influence the performance in static environments while being able to successfully handle highly dynamic scenes.

4.2. Cityscapes

The Cityscapes [7] dataset provides sequences from a car equipped with a stereo-camera, GPS and car-odometry (speed) captured in German cities. As opposed to the Virtual-KITTI dataset, Cityscapes features more challenging scenes including pedestrians and occurrences of objects which become dynamic during observation such as the example from Figure 1. For our evaluations, we use the Frankfurt stereo sequence consisting of approximately 100k frames. Its GPS trajectory is shown in Figure 9.

Figure 6 shows frames of four short sequences for qualitative analysis where, caused by the static-environment assumption, the baseline VSLAM was not able to estimate a valid trajectory. Figure 7 plots the bird's-eye view of the estimated car trajectories, while Figure 8 compares the ground truth and estimated speed derived from the VSLAM trajectory. In Sequence A (frame 42400-43400), which was presented in the Introduction, the test-car equipped with a stereo camera approaches a traffic-light controlled crossing and stops behind a truck. While the car is approaching and the truck becomes the dominant scene object within the camera image, 3D-points originating from the truck are added to the map. After the traffic light turns green, the truck starts moving forwards. Since the 3D-points of the truck are part of the map, the baseline VSLAM (b) estimates a backward motion as shown in Figure 7 and 8. In sequence B (frame 13110-13400) a car is waiting at a

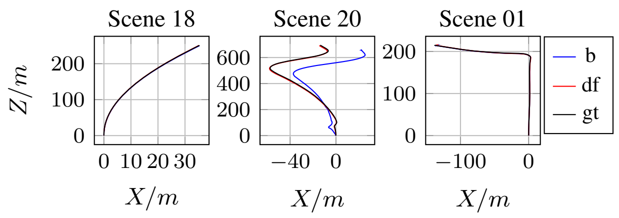

Figure 5. Bird's-eye view (X/Z plot) of Scene 18, 20 and 01 from the Virtual-KITTI [11] dataset. Dense traffic in sequence 20 induces drift in the baseline VSLAM (*b*). The VSLAM with applied dynamics factor (*df*) produces less drift and aligns closer to the ground truth (*gt*).

		$trans.err.$ /%	$rot.err.$ $/\frac{deg}{100m}$	$ATE\ rmse$ $/m$
Scene18	b	0.60	0.13	1.19
	df	**0.22**	0.15	0.37
	m	**0.22**	**0.12**	**0.34**
Scene20	b	6.95	0.61	33.69
	df	1.66	**0.20**	8.63
	m	**1.44**	0.23	**6.92**
Scene01	b	**0.34**	**0.15**	**0.88**
	df	0.40	0.16	0.96
	m	0.54	0.21	1.29

Table 2. Relative translational, rotational and absolute trajectory error on the primarily dynamic scenes 18, 20 and the primarily static scene 01. Results are average of 6 runs.

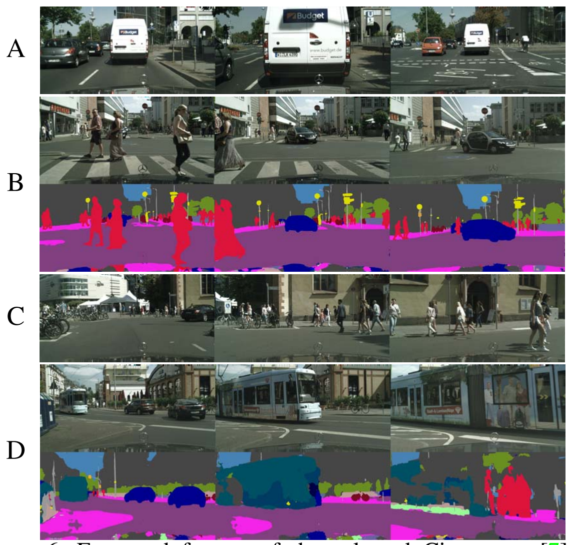

Figure 6. Extracted frames of the selected Cityscapes [7] sequences with exemplary semantic labeling result.

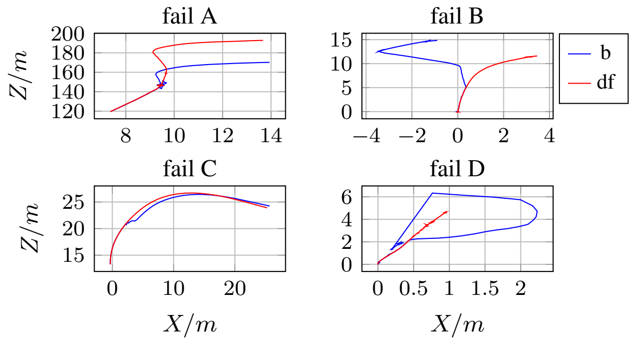

Figure 7. Extreme VSLAM failure cases from Cityscapes [7]. Bird's-eye view of the VSLAM pose estimation. The baseline (*b*) estimates an implausible trajectory. The VSLAM with applied dynamics factor (*df*) produces a more probable trajectory.

crosswalk. While the pedestrians do not significantly influence the pose estimation, the oncoming left-turning car becomes the dominant scene element and the VSLAM erroneously estimates a left-motion. In sequence C (frame 55050-55400), the car makes a right turn in front of a group of walking pedestrians which falsely becomes part of the map and consequently cause a slight trajectory shift. In sequence D (frame 76000-76300), a tram is crossing in front of the car and completely corrupts the pose estimation.

As demonstrated in Figures 7 and 8, these situations can be successfully handled by our proposed approach. Unfortunately, the Cityscapes dataset provides no accurate 6 DOF ground truth to show quantitative results with standard trajectory metrics. Leveraging the fact that the dataset provides ground truth data for speed, we derive the speed of the VSLAM trajectory and compute the direction by considering movements towards the optical axis as forward. Figure 8 shows the speed comparison with reference data where the

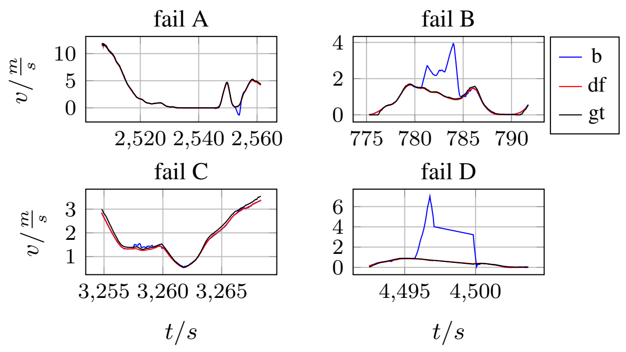

Figure 8. Failure cases from Figure 7 of the baseline VSLAM (*b*) with comparison to ground truth speed. Our applied dynamics factor (*df*) aligns closely to the ground-truth (*gt*).

Cityscapes	$v_{err}/\frac{m}{s}$	$rmse$	std
b		0.448	0.420
df		**0.176**	**0.125**
m		0.188	0.134

Table 3. RMSE and standard deviation of the velocity error v_{err} ($N = 99971$) on the Frankfurt sequence of the Cityscapes dataset.

erroneous pose estimation from the baseline VSLAM b is clearly visible.

In Figure 10 we compare the VSLAM trajectory of the complete sequence by using the speed ground truth. Because of timing and synchronization issues within the dataset, we split the sequence into 3 sub-sequences, removed error values with invalid time differences (time difference of consecutive frames less than the camera frame rate) and computed the rolling average of 6 frames. In case of a complete pose estimation failure (VSLAM lost tracking), we re-initialized the VSLAM. We display the comparison as histogram similar to the method of [2]. The vertical axis of the speed error histogram in Figure 10 is normalized and uses a logarithmic scale to emphasize rare erroneous measurements. In Table 3, which shows statistical properties of the speed errors, it can be seen that the baseline VSLAM (*b*) has more cases of large errors than our method (*df*). Additionally we computed the speed error with the semantic mask-out approach (m). As expected, the error is larger than with our dynamics-factor approach. This confirms our assumption, that parked cars are an important source for 3D-points. During execution, only our method *df* was able to provide continuous pose estimation without re-initialization. The main reasons which caused re-initialization of baseline VSLAM b were scenes such as shown in Figure 7. The mask-out approach m lost track on texture-less scenes with parked cars as the main content of the scene. This again highlights the importance and contribution of our method.

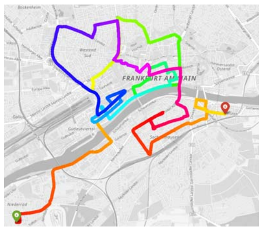

Figure 9. GPS trajectory of Cityscapes Frankfurt. Sequence starts on the green marker. Color represents continuity. Map: © OpenStreetMap contributors

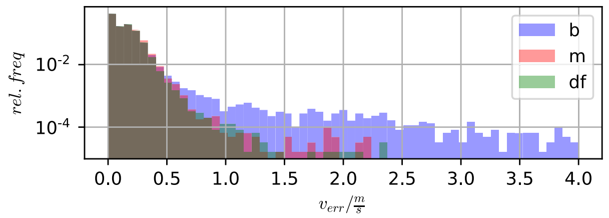

Figure 10. Speed error histogram. Our method *df* has fewer errors compared to the baseline b and mask-out only approach m.

4.3. Comparison with the State of the Art

We compare our method with DynaSLAM [3], which is most relevant to our work, on the TUM-RGBD as well as the KITTI dataset.

The comparison on TUM (Table 4) shows that DynaSLAM and our approach perform similarly (with a small advantage for DynaSLAM) in terms of the absolute translational error. Both methods significantly outperform the baseline (ORB2). However, as can be seen on the bottom of Table 4, our approach is significantly faster than DynaSLAM, which makes it suitable for real-world applications. Unfortunately, the TUM-RGBD-dataset does not feature scenes to showcase our main contribution, the distinction between dynamic and potential-dynamic 3D-points.

The evaluations on KITTI (Table 5) show that our approach, as well as DynaSLAM, perform similar to the baseline method. This is not surprising since the sequences do not contain many dynamic scenes, which our algorithm has been specifically designed to cope with. These experiments demonstrate that our method does not overfit on a small set of test cases and does not critically decrease the performance in static environments.

5. Conclusion

In this paper we tackled the challenge of dominant dynamic environments for feature-based VSLAM. We proposed to integrate semantic information as well as observation consistency to estimate the dynamics of a scene point which enables explicit handling of dynamic areas without the need of additional motion detection. In tests on synthetic and real-world data, (i) our method is able to cope with challenging scenes where traditional VSLAM methods fail, and (ii) it maintains performance on static scenes. A limitation of our method may arise when static assumed objects are actually in motion such as a building painted on a moving car. As a future work, this could be solved by including further geometric plausibility checks.

TUM	ORB2 [20]	Dyna [3]	Ours	MR [28]	MS [32]	DE [17]	DS [33]
	$ATE\ rmse/m$						
w_hlf	0.351	**0.025**	0.027	0.125	0.055	0.049	0.030
w_xyz	0.459	**0.015**	0.016	0.093	0.040	0.060	0.025
w_rpy	0.662	**0.035**	0.043	0.133	0.076	0.179	0.444
w_sta	0.090	**0.006**	0.008	0.066	0.024	0.026	0.008
s_hlf	0.020	0.017	**0.016**	0.047	-	0.043	-
s_xyz	**0.009**	0.015	0.012	0.048	-	0.040	-
	$Tracking/ms$						
med	28	53	34	-	-	-	-
mean	30	1446	35	-	-	-	-

Table 4. ATE on dyn. sequences of the TUM RGBD dataset [26]. Mean (N=10) or from respective paper. Tracking time refers to the tracking thread per frame (excl. semantic labeling).

KITTI	$trans.err./\%$			$rot.err./deg/100m$		
Seq.	ORB2	Dyna	Ours	ORB2	Dyna	Ours
0	**0.70**	0.74	0.71	**0.25**	0.26	**0.25**
1	**1.39**	1.57	1.54	**0.21**	0.22	0.23
2	**0.76**	0.80	0.78	**0.23**	0.24	0.24
3	0.71	**0.69**	0.75	**0.18**	**0.18**	0.20
4	0.48	**0.45**	0.49	0.13	**0.09**	0.17
5	**0.40**	**0.40**	**0.40**	**0.16**	**0.16**	**0.16**
6	0.51	**0.50**	**0.50**	**0.15**	0.17	**0.15**
7	**0.50**	0.52	0.55	**0.28**	0.29	0.30
8	**1.05**	**1.05**	**1.05**	**0.32**	**0.32**	**0.32**
9	0.87	0.93	**0.86**	0.27	0.29	**0.25**
10	**0.60**	0.67	0.62	**0.27**	0.32	0.29

Table 5. KITTI [12] relative errors of ORB2 [20], DynaSLAM [3] and ours (mean, N=10). Dataset consists primarily of static env., hence minor differences between algorithms.

Acknowledgements

This work received funding in the "IKT der Zukunft" programme by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT).

References

[1] P. F. Alcantarilla, J. J. Yebes, J. Almazán, and L. M. Bergasa. On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic en-

vironments. In *IEEE International Conference on Robotics and Automation*, pages 1290–1297, 2012. 2

[2] D. Barnes, W. Maddern, G. Pascoe, and I. Posner. Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments. In *IEEE International Conference on Robotics and Automation*, pages 1894–1900, 2018. 2, 3, 7

[3] B. Bescos, J. M. Fácil, J. Civera, and J. Neira. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, 2018. 2, 3, 8

[4] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 298–304, 2015. 2

[5] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016. 5

[6] 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. 2

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3, 5, 6, 7

[8] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018. 2

[9] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. *European Conference on Computer Vision*, pages 834–849, 2014. 2

[10] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017. 2

[11] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3, 5, 6

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 4, 6, 8

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, pages 2961–2969, 2017. 1, 5

[14] M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa. Mask-SLAM: Robust feature-based monocular SLAM by masking using semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 371–3718, 2018. 2, 3

[15] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007. 2

[16] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. Keyframe-based visual-inertial slam using nonlinear optimization. *Robotis Science and Systems*, 2013. 2

[17] S. Li and D. Lee. RGB-D SLAM in dynamic environments using static point weighting. *RAL*, 2(4):2263–2270, 2017. 2, 8

[18] K.-N. Lianos, J. L. Schonberger, M. Pollefeys, and T. Sattler. VSO: Visual Semantic Odometry. In *European Conference on Computer Vision*, pages 234–250, 2018. 2

[19] K. M. Judd, J. D. Gammell, and P. Newman. Multimotion visual odometry (MVO): Simultaneous estimation of camera and third-party motions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3949–3956, 2018. 2

[20] 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. 2, 3, 5, 8

[21] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1187–1200, 2014. 2

[22] T. Qin, S. Cao, J. Pan, and S. Shen. A general optimization-based framework for global pose estimation with multiple sensors. *arXiv preprint arXiv:1901.03642*, 2019. 2

[23] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018. 2

[24] D. M. Rosen, J. Mason, and J. J. Leonard. Towards lifelong feature-based mapping in semi-static environments. In *IEEE International Conference on Robotics and Automation*, pages 1063–1070, 2016. 2

[25] M. R. U. Saputra, A. Markham, and N. Trigoni. Visual SLAM and Structure from Motion in Dynamic Environments: A Survey. *ACM Computuing Surveys*, 51(2):37:1–37:36, 2018. 2

[26] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012. 5, 8

[27] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. 6

[28] Y. Sun, M. Liu, and M. Q. H. Meng. Improving RGB-D SLAM in dynamic environments: A motion removal approach. *Robotics and Autonomous Systems*, 89:110–122, 2017. 2, 8

[29] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-time dense monocular slam with learned depth prediction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6565–6574, 2017. 2

[30] C. Toft, E. Stenborg, L. Hammarstrand, L. Brynte, M. Pollefeys, T. Sattler, and F. Kahl. Semantic match consistency for long-term visual localization. In *European Conference on Computer Vision*, pages 383–399, 2018. 2

[31] J. Vertens, A. Valada, and W. Burgard. SMSnet: Semantic motion segmentation using deep convolutional neural networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, Canada, 2017. 2

[32] Y. Wang and S. Huang. Towards dense moving object segmentation based robust dense rgb-d slam in dynamic scenarios. In *IEEE International Conference on Control Automation Robotics and Vision*, pages 1841–1846, 2014. 2, 8

[33] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei. DS-SLAM: A semantic visual SLAM towards dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1168–1174, 2018. 2, 8

[34] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. Deep layer aggregation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2403–2412, 2018. 5

[35] D. Zou and P. Tan. CoSLAM: Collaborative visual SLAM in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):354–366, 2013. 2