Improving Map Re-localization with Deep ‘Movable’ Objects Segmentation on 3D LiDAR Point Clouds

Victor Vaquero[1]* Kai Fischer[2]* Francesc Moreno-Noguer[1] Alberto Sanfeliu[1] Stefan Milz[2]

***Abstract*— Localization and Mapping is an essential component to enable Autonomous Vehicles navigation, and requires an accuracy exceeding that of commercial GPS-based systems. Current odometry and mapping algorithms are nowadays able to provide this accurate information. However, the lack of robustness of these algorithms against dynamic obstacles and environmental changes, even for sort time periods, force the generation of new maps on every session without taking advantage of previously obtained ones. In this paper we propose the use of deep learning network to segment *movable* objects from 3D LiDAR point clouds in order to obtain longer-lasting 3D maps. This will in turn allow for better, faster and more accurate re-localization and trajectoy estimation on subsequent days. We show the effectiveness of our approach in a very dynamic and cluttered scenario, i.e. a supermarket parking lot. For that, we record several sequences on different days and compare localization errors with and without our *movable* objects segmentation method. Results show that we are able to accurately re-locate over a filtered map consistently reducing trajectory errors between an average of 35.1% with respect to a not filtered map version and of 47.9% with respect to a standalone map created on the current session.**

I. INTRODUCTION

Accurate localization is an essential component of autonomous vehicles and intelligent transportation systems, as it enables the accomplishment of further tasks such as path planing, safety navigation or obstacle avoidance. Moreover, estimating a precise position in a map will also allow for obtaining further environmental information such as traffic state, accidents, road closures/works, etc, which would in turn facilitate the eventual completion of the predefined mission. The same idea holds in the opposite direction, in which a correctly located vehicle may augment map information with its current observations of a scene.

Nowadays, vehicle position can be easily obtained by different Global Navigation Satellite Systems (GNSS) such as GPS, Galileo, GLONASS, etc. Although these systems can provide good results, they have limited precision in urban scenarios with buildings and other elements that may block the satellite signals. Other accurate approaches like beacon-based methods exist, but require prior installation of external infrastructures and thus are not ready for general usage.

For autonomous vehicles, it is preferable to include localization systems based on their own perceptive sensors, such as cameras or LiDAR. Although cameras can provide very rich information and colored textures, they are very sensitive to changes of illumination and lack of light. Contrary, LiDAR sensors provide robust and accurate 3D range measurements independently of the illumination conditions and even at night, which is why in this article we focus only on the use of these sensors.

Simultaneous Localization and Mapping (SLAM) has gained utmost attention within in-vehicle localization algorithms. However, in very dynamic and cluttered urban environments where vehicles and other elements are constantly moving or can potentially do, SLAM algorithms encounter difficulties on finding static and stable features that would allow to re-use the generated map on subsequent days. Thus, in many applications where the goal is just to travel predefined routes in a known area, SLAM systems may introduce unnecessary and redundant computations creating a new map each time instead of just providing localization. Our motivation in this way is to find potential moving obstacles in the source point cloud data, excluding them for the mapping pipeline so that obtaining longer standing representation while providing better accuracy.

With the advent of deep learning (DL), in-vehicle perception systems have strengthen their capacities on vital tasks for autonomous driving. Moreover, new developments are recently expanding the applicability of these techniques beyond optical cameras and postulating them also as powerful

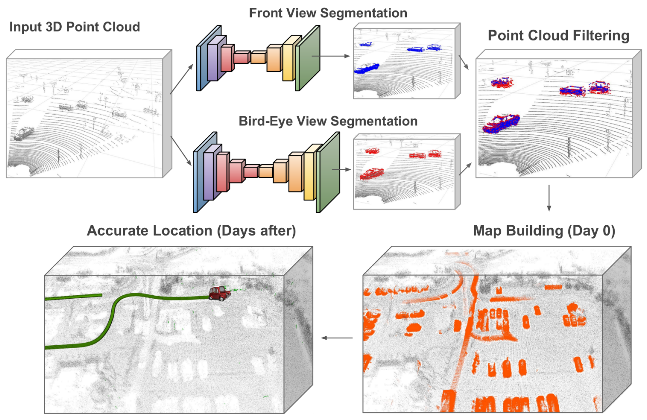


Fig. 1: We propose to segment *movable* objects from 3D LiDAR point clouds to build longer lasting maps that can assist on trajectory estimation for subsequent days. For that we employ two deep networks processing respectively a front and a bird’s eye view projection of the LiDAR input frames. By retaining mostly static elements on the scene, we are able to accurately estimate our position and trajectory on subsequent days by additionally re-localizing on the map.

*Equal Contribution

[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC Llorens i Artigas 4-6, 08028 Barcelona, Spain `vvaquero@iri.upc.edu`

[2]Valeo Schalter und Sensoren GmbH, Hummendorfer Str. 74, 96317 Kronach, Germany. `kai.fischer@valeo.com`

tools for working with 3D LiDAR point clouds [1], [2], [3], [4], [5]. This fact also motivates us to leverage DL approaches for segmenting potential *movable* objects on 3D LiDAR pointclouds.

To summarize, in this paper we propose the creation of longer lasting 3D LiDAR maps with state of the art localization and mapping algorithms by eliminating from the source the possible dynamic components of the scene. For that, we take advantage of deep learning techniques and introduce the use of two different convolutional neural networks which, having respectively as input a front and a bird's eye view projections of the 3D LiDAR point cloud, segment the *movable* objects from the scene. To demonstrate our approach, we build 3D maps of a very dynamic environment such as a parking lot with and without our segmentation applied and use it to assist for locating ourselves at different times and days, showing a consistent reduction of the average trajectory estimation error. Moreover, we show how our approach can also be employed for building a full map of an area covered in several days. Our main contributions are:

- We present a deep convolutional dual-view architecture that having as input 3D LiDAR point clouds is able to segment possible moving elements from a driving scenario, such as vehicles, cyclists or pedestrians.
- We introduce a simple yet effective re-localization approach for odometry and mapping algorithms based on feature matching against a previously generated map, extending for longer time the life of those maps.
- We show that by eliminating the possible dynamic elements in a scene, localization for subsequent days improve, demonstrating that most of the strong features extracted by current lidar odometry and mapping algorithms may lay on moving elements such as vehicles.
- We perform real life experiments recording a parking lot scenario for several days and different trajectories, obtaining consistent quantitative results that support our approach.Additionally, we show the application of our method for building maps in a multi-agent manner or through different days.

II. RELATED WORK

In this section we review the state of the art of Deep Learning techniques and localization and mapping algorithms, both applied over 3D LiDAR point clouds.

Deep learning applied on LiDAR point clouds. Although Convolutional Neural Networks (CNNs) have been successfully applied on image-based tasks such as object classification, detection or semantic segmentation [6], [7], [8] between others, its potential has not been yet extensively deployed to analyze 3D LiDAR point clouds. However, recent approaches are demonstrating the high capacities of deep neural networks to process LiDAR information on problems such as vehicle detection [1], [2], [3], [5] or motion segmentation [4]. In addition, the availability of large-scale real world datasets [9] as well as synthetic data [10] are allowing the training of data driven deep models more easily.

Initial straightforward applications over 3D LiDAR data made use of 3D convolutions [11], more efficient sparse convolutions [12], or just subdivided the input point cloud into voxels [3].Some other very recent approaches directly use the raw point cloud. In this way, PointNet [13] applies a set of transformations and multi-layer perceptrons to generate global point cloud features which are then used for classification and segmentation tasks. PointNet++ [14] proposes to recursively apply PointNet on nested areas of the input point cloud, learning additionally local features with increasing contextual scales. FrustrumPointNet [2], explores larger areas and extracts 3D frustums from bounding boxes given by a 2D CNN detector over RGB images. However, all these methods required of high computational power and include ad-hoc steps that can greatly affect its performance and stability.

However, nowadays the most embraced procedure is to project the 3D LiDAR point cloud to create 2D representations from which to apply standard image-based 2D convolutions. In this way, [1] uses a front view projection encoding the range distance and height of each 3D point and train a deep network to extract vehicle 3D bounding boxes. Similarly, [15] creates a front view projection of the polar LiDAR coordinates along with the reflectivity of each point to segment vehicles by predicting the *vehicleness* confidence of each point. On the other hand, BirdNet[5], TopNet [16] or RT3D [17] make use of a bird's eye view projection of the point cloud, encoding different features on each cell.

In this paper we devise a dual-branch LiDAR convolutional architecture for filtering all the possibly (*movable*) objects in the scene. One branch processes a front view of the 3D Lidar input whereas the other a bird's eye view projection, and each of them will predict the probability of the 3D projected points (*pixels*) to belong to a *movable* class (e.g. car, bicycle, pedestrian, etc.) or otherwise a *static* point.

Map building and localization. Localization and Mapping with LiDAR is a very active research topic in the robotics and automotive community. Early approaches used 2D laser data and ICP methods to correct the ego-motion distortion. However, when employing more complex 3D LiDARs with further amounts of information more sophisticated approaches need to be considered, like including other sensor's information such as IMU, wheel encoders or GPS/INS using for example extended Kalman filters [18], [19].

Some early approaches take motivation from visual slam methods [20] and create intensity images using the laser returns from which to extract and match distinctive features between frames to infer the ego motion [21], [22]. However, in these methods based in matching visual or geometric features, most commonly the localization and trajectory is recovered by a batch optimization post-process, which make them unsuitable for real time localization.

The introduction of LOAM algorithm [23] supposed a great advance in terms of accuracy –achieving top position on the Kitti Odometry benchmark– as well as real-time performance. It proposes to divide the complex SLAM problem in two algorithms, where one estimates odometry at a high

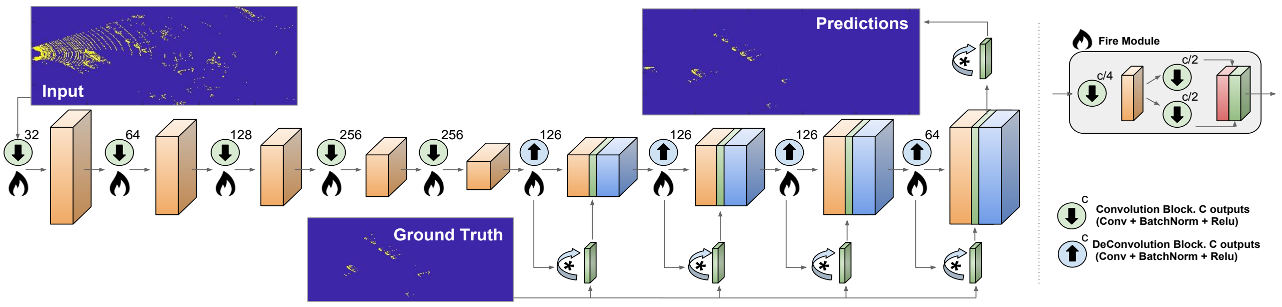

Input
Predictions
Ground Truth
32
64
128
256
256
126
126
126
64
Fire Module
c/4
c/2
c/2
C
Convolution Block. C outputs (Conv + BatchNorm + Relu)
C
DeConvolution Block. C outputs (Conv + BatchNorm + Relu)

Fig. 2: We model $\mathcal{F}_{BE}$ as a refined encoder-decoder architecture. As the inputs are quite big (500×600) we apply five contractive and five expansive levels. To get richer features at each level, we insert customized fire modules that capture local and context information from the previous feature maps. These modules first reduce the number of feature channels and apply two parallel sets of convolutional filters on them to finally concatenate the results, obtaining local and context aware features. Intermediate losses are computed in this network, merging predictions at different resolutions.

frequency but low fidelity whereas the other runs at lower frequencies for fine matching and registration of the point cloud. From this approach, incremental improvements have been done. LeGO-LOAM [24], proposed a lightweight, real-time pose estimation method that has into account the presence of a ground plane in its segmentation and optimization steps to obtain distinctive planar and edge features for the next optimization method. We base this approach to build our maps as it provides a real-time and accurate representation as well as is more robust using real raw LiDAR data.

Other contemporary approaches propose to infer the vehicle position by matching segments from the point cloud belonging to partial or full objects as well as sections of larger structures [25]. In this way they obtain a good balance between local descriptors, which may suffer from ambiguity without context information, and global features, which are viewpoint dependent. The method extract several features from the segmented clusters and try to match those segments over the ones existing in a map. Further incremental approaches employ a data driven feature encoder to extract compact and discriminative features from the segments [26]. These features can be used for creating a compact map representation due to its high reconstruction capacities and for accurate location over an existing map, as long as it does not contain much movable elements.

In this way, the ability to create longer standing maps overcoming challenging issues such as map scalability and updatability or management of dynamic elements, is still a pending task for SLAM algorithms [27], [28] In the presented work, we directly segment the input information with a deep learning pipeline prior to build any map. Thus, we avoid from the beginning the inclusion of possible outliers and known-dynamic elements, so therefore do not give chances to further extract any feature from them. By doing so, we demonstrate in Section IV that more accurate localization is possible on a constantly changing environment over a map created days before.

III. APPROACH

The main objective of this paper is to demonstrate how, by pre-filtering possibly moving objects from the scene, we are able to accurately locate during longer periods of time in standard 3D maps such as the ones generated with common SLAM or LiDAR odometry and mapping (LOAM) algorithms. Our approach has two main modules, which are detailed in this section: A) deep-learning based segmentation of movable objects in 3D LiDAR point clouds; B) accurate re-localization at different days over a map built from a cluttered dynamic scenario using standard LeGO-LOAM algorithm. This can also be employed to build a full map over several days or in a multi agent way.

A. Movable Objects Segmentation

For this task, our only input is a 3D LiDAR point cloud $\mathcal{P} = \{q_1, \cdots, q_Q\}$, where each point $q_k \in \mathbb{R}^4$ is represented by its Euclidean coordinates and the returned reflectivity. Our objective is to classify each of these 3D points as belonging to a $\{$*movable*, or *non-movable*$\}$ class, where we consider as *movable* all the Kitti [9] annotated classes, i.e. 'Vehicle', 'Van', 'Truck','Cyclist','Pedestrian', 'Tram' and 'Misc'.

We formulate the task as a binary semantic segmentation one, in which we perform a per-point classification prediction. To solve it, we take advantage of the recent success of deep convolutional neural networks and propose to model two segmentation functions $\mathcal{F}_{FR}$ and $\mathcal{F}_{BE}$ (see Fig. 2), each one respectively committed over a front view $\mathbf{I}_{FR}$ and a bird's-eye view $\mathbf{I}_{BE}$ 2D projections of the 3D LiDAR data. Therefore we want to learn the $\mathcal{F}$ mappings such that:

$$\begin{aligned} \mathcal{F}_{FR} &: (\mathbf{I}_{FR}, \mathbf{Y}_{FR}) \rightarrow \hat{\mathbf{Y}}_{FR} \\ \mathcal{F}_{BE} &: (\mathbf{I}_{BE}, \mathbf{Y}_{BE}) \rightarrow \hat{\mathbf{Y}}_{BE} \end{aligned} \tag{1}$$

where $\mathbf{Y}_{FR}$ and $\mathbf{Y}_{BE}$ are two ground-truth masks that indicate whether or not each 3D projected point belongs to a movable object and $\hat{\mathbf{Y}}_{FR}$ and $\hat{\mathbf{Y}}_{BE}$ will be the predicted probability maps on each projection plane. Next, we present the main components of the stated deep segmentation approach, such as the projections and ground truth created from the 3D point cloud, the convolutional architectures involved and the training process to model our objective functions.

Front View Projection. The input Front view, $\mathbf{I}_{FR}$, is obtained similarly than in [15]. We arrange the 3D point cloud $\mathcal{P}$ according to the Velodyne HDL-64 geometry into a

2D array such that $\mathbf{I}_{FR} \in \mathbb{R}^{H \times W \times C}$. The Euclidean points (x, y, z) are transformed to spherical coordinates, where the elevation angle θ represent the $H = 64$ Velodyne horizontal lasers. We filter the point cloud in the corresponding camera field of view that contains the Kitti annotations ($\phi \in [-40.5, 40.5]$) and discretize it according to the sensor manufacturer using an azimuth step of $\Delta\phi = 0.18$ degrees, which map to a width of $W = 448$ pixels. In the third dimension of our front view map we store the corresponding range values ρ, and reflectivity r, so therefore $C = 2$ channels.

Bird's Eye View Projection. The bird's eye (zenithal) view $\mathbf{I}_{BE}$ is obtained over an area of 60×50 meters in front of the LiDAR sensor after carefully observing that roughly 95% of the annotated vehicles in Kitti are within these margins. Inspired by [5], [16], we generate a 2D grid with a resolution of 0.1 meters and project the cropped point cloud on it. We therefore obtain a bird's eye view $\mathbf{I}_{BE} \in \mathbb{R}^{H' \times W' \times C'}$, where $H' = 600$, $W' = 500$, and $C' = 6$, accounting for six different features: 1) a binary occupancy term with zero value if no points are projected in the cell and one otherwise; 2) an absolute occupancy term, counting the total number of points in the cell; 3) the mean reflectivity value of the points on the cell; and 4, 5, and 6) the mean, minimum and maximum height values of the points projected on the cell.

Ground Truth Generation. The movable elements ground truth for the front $\mathbf{Y}_{FR}$ and bird's eye $\mathbf{Y}_{BE}$ projections is generated by using the 3D-oriented bounding boxes from the Kitti Tracking dataset. We transform these bounding boxes from the camera to the LiDAR frame and label the 3D points that fall inside each box of all the annotated movable classes.

Network Architectures. We propose for both projection domains contractive-expansive architectures which allows for a good embedding of features. Additionally, we include skip connections and concatenate feature maps between contractive and expansive parts to build stronger features that will help the learning process by back-propagating purer gradients from the upper parts to the lower layers.

1) Front view Architecture: We employ for the front view segmentation task the deconvolutional architecture proposed in [15]. As key design, it imposes a stride ratio of 1:2 in the first convolutional layer to obtain more tractable intermediate feature maps, reducing the input size imbalance of $\mathbf{I}_{FR}$ from (64 vs 448) to (64 vs 224). From there, it performs two more resolution decreases in the contractive part of the network, which are later recovered on the expansive sector. Additionally, in this network we also carefully design the initial filter sizes according to the observed shape of the most predominant movable objects in this view (vehicles), so imposing a filter of 7x15 with a big initial receptive field.

2) Bird's Eye View Architecture: Within this view we encounter a new challenge which is that, averaging within the training set, movable objects occupy less than the 8% of the grid-cells per frame with the chosen grid resolution of 0.1 meters. Segmenting these small areas is still a challenging problem for deep neural networks, and force us to design the specialized architecture of Fig. 2 to manage the trade-off between accuracy, number of parameters and speed.

To keep the number of network parameters small while still providing high accuracy, we employ in our architecture a customized version of the well-established convolutional 'fire modules' [29], [30], [31], which can be seen in the top area of Fig- 2. Within these modules, we initially use a convolution layer to reduce the number of feature maps and later on we apply in parallel two new convolutional layers with different filter sizes. Results are finally concatenated to obtain robust features with local and large context-aware information, while using a low number of parameters. In our architecture we include these 'fire modules' after each change of resolution of the feature maps.

Training the Networks. We train both networks to segment the front and bird's eye views in a supervised manner using a class weighted cross entropy loss function [15], defined as:

$$\mathcal{L}^{WCE}(\mathbf{I}^n, \mathbf{Y}^n) = -\sum_{h,w,l}^{H,W,L} \omega(Y^n_{h,w}) Id_{[\mathrm{Y}^n_{h,w}]} \log(\mathcal{F}(I^n, Y^n)_{h,w,l}), \tag{2}$$

where $\mathbf{I}^n$ is the n-th training projection sample and $\mathbf{Y}^n$ is the corresponding ground truth map. We compute a class imbalance weighting function ω as the inverse ratio between the vehicle and background classes from the training set samples. *Id* is an index function that selects the predicted probability associated to the expected ground truth class.

In order to guide the network to a correct result faster, we implement a multi-scale solution for the segmentation problem by introducing intermediate predictions and losses at different resolutions, which insert valuable gradients at middle levels. Hence, we compute the final loss $\mathcal{L}$ independently for the front and bird's eye networks as

$$\mathcal{L}(\mathbf{I}^n, \mathbf{Y}^n) = \sum_{m=1}^{M} \lambda_m \mathcal{L}^{WCE}(\mathbf{I}^n_m, \mathbf{Y}^n_m) \tag{3}$$

where λ_m are regularization weights for each resolution loss and m is the resolution step in which the loss functions is computed. We compute 3 partial losses for the front view projection network, and 5 for the bird's eye view one.

We train our networks using the Kitti Tracking dataset from which we obtained the ground truth of movable objects. To preserve the geometry properties of the driving scene we augment the dataset with horizontal flips in the front view and vertical ones in the bird's eye view with a 50% chance. For the training procedure, we initialize the architectures with He's method [32] and use Adam optimization with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We train each network independently on a single Nvidia 1080Ti GPU using a batch size of 10 during $400,000$ iterations. We start with a learning rate of 10^{-3}, which is halved every $50,000$ iterations after the first $150,000$. Attending to the ratio between classes on each domain, we set the regularizator ω to 25 in the front view and to 1000 in the bird's eye view network. The multi-resolution loss regularizers λ_r are set to 1, assigning equal importance to each resolution.

Filtering Movable Objects. To filter out the points that belong to movable objects in the 3D input LiDAR point cloud, we first obtain the segmentation prediction from the two deep models and fuse the obtained predictions back in the 3D Euclidean space. Next we cluster points from both predictions and validate them according to a minimum size of at least 50 points per cluster. We also discard clusters according to its mean probability predicted, where we weight the contribution of the bird's eye samples to be twice as the front view ones (0.2 vs 0.1 respectively) as this last have showed noisier results, and set a threshold of 0.13 to approve resulting clusters. Once we have the final clusters of the predictions, we filter the original input LiDAR data by eliminating all points in a radio of 10 cms from a predicted one, so that restricting the effect of possible projection errors.

B. Vehicle Localization

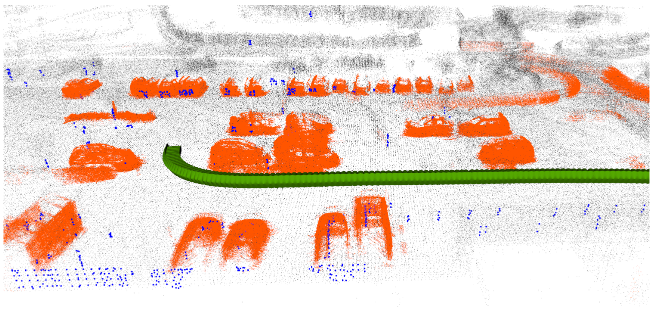
To locate inside an already built map our system uses just Velodyne LIDAR data and, if available, information from low-cost car GPS. Although the latter is very imprecise and has a very low refresh rate, it is a standard equipment nowadays in most vehicles, so we can use it to estimate a coarse initial position which is afterwards refined by using our re-localization algorithm. Next, we first show the creation of the ground truth map at day zero against which we aim to locate on the subsequent days, and then we detail our localization approach, which consist on obtaining an coarse initial guess followed by the final accurate localization.

Ground Truth Map Building. For building the ground truth map from which to validate our approach, we employ DGPS-synchronized LiDAR scans that are feed to the state of the art LeGO-LOAM algorithm, in charge of composing the final map representation. LeGO-LOAM also extracts edge and surface features from the generated map by analyzing the local surface properties of certain areas in the point cloud. Edge features are extracted from rough local regions, whereas surface ones from smooth surfaces.

In order to obtain more distinctive features from the map, the LeGO-LOAM algorithm does not account for features within a minimum distance from some of them considered as strong. This fact can have a big counterpart. As movable objects like vehicles have a very prominent surface structure, there are therefore more likely to be chosen as strong features over other relevant static features of the environment. In this way, by pre-filtering the point cloud we enforce the selection of distinctive strong features just from static elements instead of from movable objects, thus allowing better inter-day re-localization. A comparison of the extracted features from the full and the filtered point cloud respectively can be observed in Fig. 3 Additionally, we also remove the ground-floor features determined by LeGO-LOAM to obtain a more compact and distinctive representation of the features map.

Finally, our resulting ground truth map (GT-Map) consists of LeGO-LOAM features extracted from each filtered frame along with the own frame transformation from the DGPS.

Initial Pose Estimation. To initially accelerate the current localization process over a previously existing map, we can use any rough prior information. In our experimentation, we use a commercial GPS available nowadays as standard on-vehicle equipment. Notice that we use it only to speed up the initial localization, but it is not strictly necessary.


(a) Features extracted from full point cloud

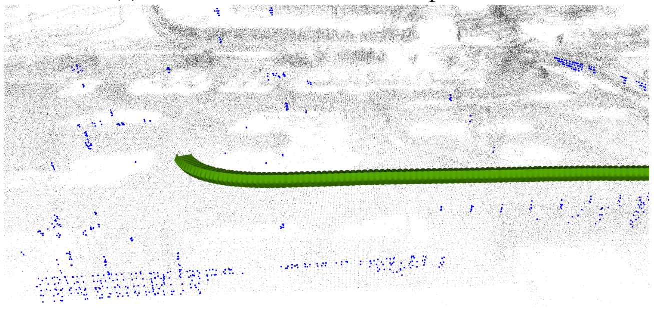

(b) Features extracted from filtered point cloud

Fig. 3: Comparison of the extracted features (blue) from the full point cloud and the one with removed movable objects respectively. By providing an unfiltered point cloud the feature extraction mechanism selects a vast amount of points from dynamic objects which can be compensated by applying our proposed movable object detection algorithm.

As GPS solely provides coarse information about the position, we additionally need to estimate the initial orientation. In order to do that, we extract a subset of our ground truth map around the GPS coordinates and perform feature matching from our current observed frame using ICP. Feature points used for the current frame are extracted and selected in the same way as described in the map building process. To optimize the ICP process and get an initial orientation estimation, we firstly perform a coarse matching prediction by applying ICP over different rotations of the current frame features on steps of 45 degrees. For each rotation, we get a fitting score, which describes the remaining sum of squared differences from the feature points of the current frame to their corresponding nearest neighbours in the ground truth map. Our initial orientation guess is selected as the one with a fitting score lower than 0.4.

Finally, the transformation to our initial pose estimation can be expressed as $T_{init} = T_{GPS} \cdot T_{ICP} \cdot T_{Rot}$, where T_{GPS} would be initial rough position estimate given by the commercial GPS, T_{Rot} is the best fitting initial rotation found for the ICP, and T_{ICP} refers to the final refined transformation obtained by the ICP algorithm that optimizes the feature matching best. Notice here that T_{GPS} and T_{Rot} are just used to speed up the matching process, and that any

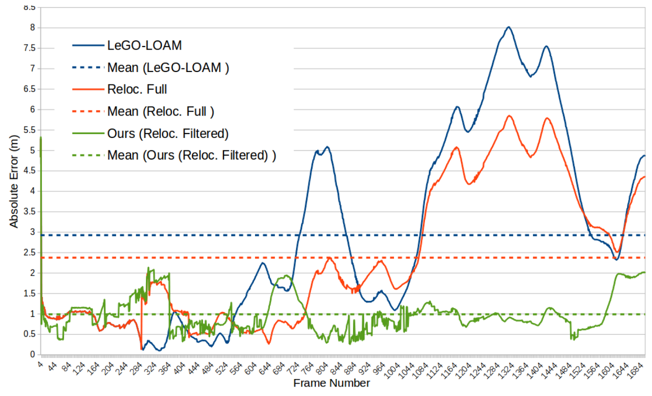
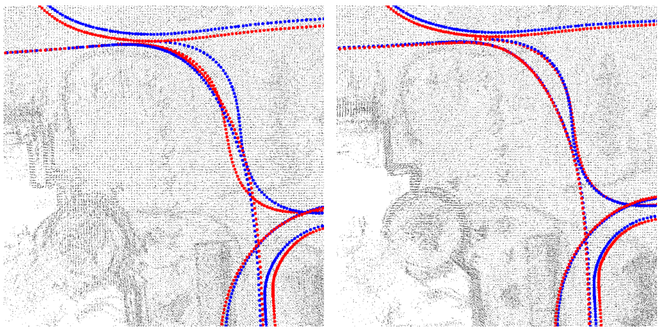


Fig. 4: Progression of the absolute error of re-locating Sequence 3 in Map 1 for the three regarded methods in comparison to the ground truth data including each corresponding Mean Absolute Errors (MAE) as dotted line.



(a) Drift using Reloc. Full (b) Better with Reloc. Filtered

Fig. 5: Qualitative results comparing the performances of ‘Reloc. Full’ and ‘Reloc. Filtered’ on the Map Extension experiment. In red we display the estimated trajectories along the days and in blue the ground truth. For ‘Reloc. Full’ the resulting map shows blurry and doubled map contents caused by drift from the ground truth due to incorrect feature matching which are clearly compensated by our approach.

other prior coarse pose estimation could be employed.

Continuous Re-Localization Once the initial pose estimation is performed, we again use LeGO-LOAM to continuously calculate further transformations based on our segmented LiDAR scans. At the same time, we perform relocalization, trying to match our current position against the pre-existing generated map (GT-Map). For this continuous re-localization steps, we follow a similar process than above and employ the extracted features from the current scan with ICP to correct possible drifts caused by the current trajectory obtained with the LeGO-LOAM algorithm.

In comparison to the initial pose estimation, these relocalization transformations are simpler to calculate, as they only depend on the current estimated position and the correction given by the ICP over the GT-Map. Therefore, $T_{reloc} = T_{ICP} \cdot T_c$, where T_c stands for the current pose estimated. The threshold for the ICP fitting score in the course of this re-localization step is lowered to 0.3 in order to estimate the transformation more robustly.

IV. EXPERIMENTS

To show the effectiveness of the proposed approach, we have recorded 7 different sequences of a cluttered and dynamic urban environment, i.e. a supermarket parking lot, on different days and at diverse hours. We first detail the data acquisition process for performing the experiments. Next, we show our re-localization capacities in this highly unsteady scenario using as GT-Map a recording from a different day, which would be useless if it were not for our approach, as it will not last more than the session for which it was created. Finally, we show additional application of our approach for building a map through different days, which can also extrapolated to a multi-agent map building task.

A. *Data acquisition*

The data used for evaluating our experiments was captured and recorded with a Test car by Valeo which is equipped with multiple sensors. The relevant hardware concerning our experiments is a Velodyne LiDAR HDL-64E S3, a differential GPS by IMAR and the serial production car GPS. The 7 sequences were recorded on a parking lot of a local shopping mall in Kronach at different days and times to ensure diverse constellation of the parking cars. Sequence 1 was recorded early in the morning aiming to obtain an almost empty parking lot. Sequence 2 and 3 were recorded on another day with the area being slightly crowded. Sequence 4 to 7 were recorded on different hours of another day with a very crowded parking lot. Our dataset therefore consists of 7 sequences recorded at three different days with diverse constitution of the parking lot.

B. *Re-localization in dynamic environments*

To validate the proposed method we use sequences 1 and 3 for building GT-Maps as described in III-B. For each sequence, we build two maps a Full one including all objects and a filtered with removed movable objects. In this regard sequences 2 to 7 are used for re-localization in Map 1 and sequences 1 and 4 to 7 for Map 2 respectively, ensuring interday experiments with different environmental constitution.

In our experiments, we apply three different methods:

- **LeGO-Loam:** Processing of subsequent frames in an unfiltered map solely based on the pose estimation calculated by LeGO-LOAM.
- **Reloc. Full:** Parallel pose estimation by LeGO-LOAM and re-localization in an unfiltered map based on full unfiltered current frames.
- **Ours (Reloc. Filtered):** Parallel pose estimation by LeGO-LOAM and re-localization in a filtered map based on filtered current frames.

For each method, its performances in consideration of localization in a dynamic pre-build map are validated based on three different metrics:

- **First relocation** ($1st_r$): Frame number of the initial pose estimation.
- **Number of relocations** ($\#_r$): Number of frames at which the particular algorithm was able to relocate.

- **Mean Absolute Error** (MAE): in meters, the averaged absolute error of the estimated positions over the whole sequence compared to ground truth poses of the DGPS.

Table I shows the re-localization performances of the considered methods and sequences applied on the respective maps. Additionally, we show in Fig. 4 the absolute error obtained with the three algorithms for re-localization along sequence 3 using the GT-Map from sequence 1 (almost empty parking map) in comparison to the ground truth data.

At the lights of the results we can observe that filtering 'movable' objects from feature point clouds greatly improves the performance of re-localization in dynamic environments. Compared to the unfiltered re-localization (Reloc. Full, in Table I) we reduced the MAE value a 16.5% using Map 1 and up to 50.57% using Map 2 averaging over the respective sequences. Additionally our approach consistently scores highest in number of localized frames and mostly gets faster initial localization compared to the unfiltered approaches.

Since GPS data is used solely for localization until the initial pose estimation is obtained, there are bigger MAE values when this initial localization takes place late. The consequences of this effect can be observed in Table I) when re-locating at sequences 5 or 6 over both maps. Observe how over these sequences the first relocation occurs rather late, and therefore the MAE error is higher. This impact is mainly caused by partially nonexistent sequence and map overlap so therefore re-localization cannot be applied. The contrary effect can also be noticed in sequence 7 applied over Map 2, where unfiltered re-localization ranks higher in MAE than our approach. In this occasion, similar constellation of vehicles were present in the parking, so for the two baseline algorithms the first re-localization is performed fast.

Another factor affecting the shown results are partially erroneous re-localizations essentially happening in curves where a slight deviation in orientation estimation has a huge impact on the subsequent trajectory calculation. Assuming no further re-localizations occur after an erroneous re-localization there will be a huge drift in the ensuing poses which is reflected in the results of the unfiltered re-localization whose MAE is partially exceeding the ones of the standard LOAM approach. Comparing to the results of our approach this perturbation can be mostly eliminated, since in the filtered environment more distinctive, static features are selected to guarantee more robust pose estimations.

C. Multi day map extension

Apart from experiments on re-localization in dynamic environments we can prove the adaptability of the proposed algorithm to the application of a mapping process during several days. Here we are able to show that filtering movable objects from the processed data drastically improves the ability to build correspondences between maps from different days and consequently, the quality of the final map. In our experiments we choose sections of sequences from three different days which are partially overlapping to compose a final mapping of the entire parking lot.

Starting with a segment of sequence 1 we are building the map solely using the LeGO-LOAM algorithm with loop closure to accomplish a detailed mapping result. Next we are processing a section of sequence 3 and do re-localization in the previous built map based on the extracted features. In contrast to the previous experiments where we were doing a re-localization pose estimation separately for every frame here we are using the found correspondences to the previous map to do a graph optimization based on the inbuilt loop closure method of LeGO-LOAM in order to continuously update and refine the complete map. The previous step is repeated with a section of sequence 7 applied on the currently created map, so the final outcome is a merged map built upon sequences of three different days.

In this context we can prove the strength of our approach which, by filtering out movable objects, is able to establish more robust connections to previous days maps and more often, therefore obtaining a cleaner and more accurate final map. A comparison of the quality of the resulting maps, with and without including movable objects can be observed in figure Fig. 5. Looking at the unfiltered map more blurry regions and doubled elements can be observed which are on the other hand compensated in our filtered map.

To get a quantitative measurement of these experiments we are comparing the composed trajectories of the map extension to the corresponding ground truth data. Here we use ICP to align the individual trajectories to the ground truth one where we are using the resulting fitness score as validation metric. In this investigations it is more important to have correct transformations in between the respective partial sequences rather than achieving a minimum pose-to-pose distance at every time step from the start like in the previous experiment. Therefore we calculate the remaining sum of squared distances when aligning the composed trajectories to the ground truth. Here we can show that by filtering out movable objects we obtain an improvement of 50% compared to the unfiltered approach, lowering the trajectory fitting score from 0.26 to 0.13.

V. CONCLUSION

In this work we proposed a robust LIDAR-based re-localization algorithm for autonomous driving tasks in highly dynamic environments. By filtering possible movable objects based on a convolutional dual-view architecture we can achieve a more robust, distinctive and static representation of the current environment, which can be used for further processing tasks such as path planning, map updating or re-localization. We proved that by filtering movable objects, the accuracy of re-localization inside a pre-built map can be increased by an average percentage of 35.1% compared to re-localization using the full point clouds and by 47.9% compared to a state-of-the-art lidar odometry and mapping algorithm. Furthermore we showed the adaptability of our approach by applying it to a multi-day map building task, where the accuracy of the final filtered map after applying our method exceeds the ones using full point clouds by 50%.

TABLE I: Re-Localization results comparing LeGO-LOAM, Reloc. Full and Ours (Reloc. Filtered)

GT-Map	Curr. Seq	LeGO-LOAM		Reloc. Full			Ours (Reloc. Filtered)			MAE Improvements (%)	
		$1st_r$	MAE (m)	$\#_r$	$1st_r$	MAE (m)	$\#_r$	$1st_r$	MAE (m)	To Reloc.Full	To LeGO-LOAM
1	2	3	1.73	7	3	1.29	288	8	0.84	34.42 %	51.07 %
	3	2	2.93	20	2	2.38	257	2	0.99	58.30 %	66.13 %
	4	197	2.09	0	275	6.38	108	8	1.08	83.05 %	48.23 %
	5	273	3.65	0	274	3.74	78	233	1.98	47.03 %	45.76 %
	6	567	5.94	0	279	4.14	5	281	3.91	5.56 %	34.25 %
	7	33	3.59	0	27	4.05	49	26	1.01	75.05 %	71.84 %
2	1	2	2.80	503	2	0.74	548	2	0.70	5.02 %	75.03 %
	4	67	2.26	14	66	1.88	101	56	1.73	7.91 %	23.40 %
	5	273	2.78	54	289	2.73	112	269	2.12	22.32 %	24.32 %
	6	168	4.73	9	168	4.61	26	165	2.39	48.29 %	49.51 %
	7	1	1.93	53	1	1.20	135	1	1.21	-0.79 %	37.37 %

References

[1] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," in *Robotics: Science and Systems (RSS)*, Robotics: Science and Systems Foundation, 2016.

[2] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 918–927, 2018.

[3] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4490–4499, 2018.

[4] V. Vaquero, A. Sanfeliu, and F. Moreno-Noguer, "Deep lidar cnn to understand the dynamics of moving vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[5] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. De La Escalera, "BirdNet: A 3D object detection framework from LiDAR information," in *IEEE Conference on Intelligent Transportation Systems (ITS)*, pp. 3517–3523, 2018.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems*, 2012.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 39, no. 6, pp. 1137–1149, 2017.

[8] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 39, no. 4, pp. 640–651, 2017.

[9] 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.

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[11] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 1513–1518, 2017.

[12] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *IEEE International Conference on Robotics and Automation*, pp. 1355–1361, 2017.

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Neural Information Processing Systems*, 2017.

[15] V. Vaquero, I. Del Pino, F. Moreno-Noguer, J. Sola, A. Sanfeliu, and J. Andrade-Cetto, "Deconvolutional networks for point-cloud vehicle detection and tracking in driving scenarios," in *European Conference on Mobile Robots (ECMR)*, 2017.

[16] S. Wirges, T. Fischer, C. Stiller, and J. B. Frias, "Object detection and classification in occupancy grid maps using deep convolutional networks," in *IEEE Conference on Intelligent Transportation Systems*, pp. 3530–3535, 2018.

[17] N. Sun, Y. Zeng, Y. Hu, Y. Han, J. Ye, X. Li, and S. Liu, "RT3D: Real-time 3-D vehicle detection in LiDAR point cloud for autonomous driving," *IEEE Robotics and Automation Letters*, pp. 3434–3440, 2018.

[18] F. Moosmann and C. Stiller, "Velodyne slam," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[19] D. Wang, H. Liang, T. Mei, H. Zhu, J. Fu, and X. Tao, "Lidar scan matching ekf-slam using the differential model of vehicle motion," in *IEEE Intelligent Vehicles Symposium (IV)*, 2013.

[20] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[21] H. Dong and T. D. Barfoot, "Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation," in *Field and Service Robotics*, pp. 327–342, Springer, 2014.

[22] S. Anderson and T. D. Barfoot, "Ransac for motion-distorted 3d visual sensors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2093–2099, 2013.

[23] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and Systems*, vol. 2, p. 9, 2014.

[24] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, IEEE, 2018.

[25] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3d point clouds," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5266–5272, IEEE, 2017.

[26] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: 3d segment mapping using data-driven descriptors," in *Robotics: Science and Systems (RSS)*, 2018.

[27] 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.

[28] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, pp. 194–220, 2017.

[29] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv:1602.07360*, 2016.

[30] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3d lidar point cloud," in *IEEE International Conference on Robotics and Automation*, pp. 1887–1893, 2018.

[31] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *IEEE International Conference on Robotics and Automation*, 2019.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE International Conference on Computer Vision*, 2015.