2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Dynamic Object Aware LiDAR SLAM based on Automatic Generation of Training Data

Patrick Pfreundschuh[1,2], Hubertus F.C. Hendrikx[2], Victor Reijgwart[1], Renaud Dubé[2]
Roland Siegwart[1], Andrei Cramariuc[1]

[1]Autonomous Systems Lab, ETH Zürich, {firstname.lastname}@mavt.ethz.ch
[2]Sevensense Robotics AG, {firstname.lastname}@sevensense.ch

***Abstract*— Highly dynamic environments, with moving objects such as cars or humans, can pose a performance challenge for LiDAR SLAM systems that assume largely static scenes. To overcome this challenge and support the deployment of robots in real world scenarios, we propose a complete solution for a dynamic object aware LiDAR SLAM algorithm. This is achieved by leveraging a real-time capable neural network that can detect dynamic objects, thus allowing our system to deal with them explicitly. To efficiently generate the necessary training data which is key to our approach, we present a novel end-to-end occupancy grid based pipeline that can automatically label a wide variety of arbitrary dynamic objects. Our solution can thus generalize to different environments without the need for expensive manual labeling and at the same time avoids assumptions about the presence of a predefined set of known objects in the scene. Using this technique, we automatically label over 12000 LiDAR scans collected in an urban environment with a large amount of pedestrians and use this data to train a neural network, achieving an average segmentation IoU of 0.82. We show that explicitly dealing with dynamic objects can improve the LiDAR SLAM odometry performance by 39.6% while yielding maps which better represent the environments. A supplementary video[1] as well as our test data[2] are available online.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is an important capability of autonomous robots [1]. Feature-based LiDAR SLAM algorithms often use matches between currently and previously extracted 3D features to estimate the trajectory of a robot [2]–[4]. Such approaches typically assume that the majority of features are fixed in space. However, in many cases robots are exposed to dynamic environments where the presence of moving objects is unavoidable, e.g. urban delivery robots operating among pedestrians. Geometric features extracted from such dynamic objects add uncertainty and thus can lead to increased inaccuracy in the pose estimation [5]. Dynamic objects can also end up in 3D reconstructions created by SLAM algorithms, which is not desired since these features are likely to no longer exist when the same places are revisited.

Only a handful of LiDAR-based SLAM systems deal with dynamic objects explicitly. The most common approaches [5], [7], [8] assume that dynamic objects in the scene are restricted to a distinct set of classes (e.g. cars, pedestrians, bicyclists), that can be detected using recent advances in deep learned semantic segmentation [6], [9], [10]. These supervised learning methods require expensive and difficult to obtain training data, and therefore the capability to detect new dynamic objects remains constrained. Other not data driven approaches use heuristics such as ignoring all objects below a certain size because they could potentially move [11]. However, such assumptions can lead to the removal of features that are valuable for matching or 3D reconstruction.

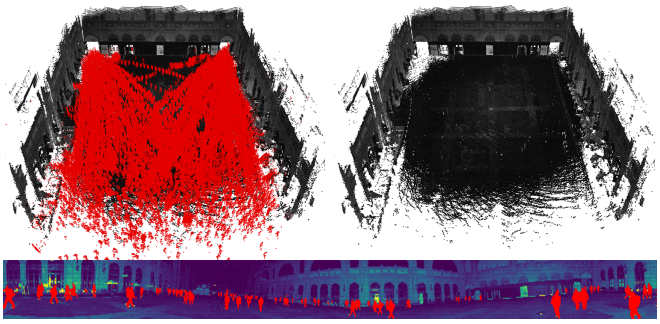

Fig. 1. *Top:* Point cloud map built using the dynamic object aware LOAM algorithm. Static points included in the map are shown in grayscale, while points colored in red originate from dynamic objects that were discarded. *Bottom:* Example output of dynamic object detections in a single 3D LiDAR scan of the same scene. Dynamic points are colored red in the intensity image resulting from the point cloud. Dynamic object detection is performed using 3D-MiniNet [6] trained on our automatically labeled dataset.

To the best of our knowledge, we present the first unsupervised approach to a generic dynamic object aware LiDAR SLAM, using deep learning. Even though we also use a neural network to detect dynamic objects and train it in a supervised manner, the labels and training data are generated completely automatically, thus making the overall system unsupervised.

Currently available datasets with annotated point clouds are all manually labeled. The most extensive ones include Waymo Open [12], nuScenes [13] and SemanticKITTI [14]. These datasets were all recorded for autonomous driving applications and therefore methods trained using this data will under-perform in drastically different environments, such as indoors. To alleviate the necessity for time-consuming manual labeling of data we present a novel dynamic object detection approach to automatically annotate point clouds. Our approach extends the idea that dynamic objects can be detected by

[1]https://youtu.be/LcDxd97r1Gc
[2]https://projects.asl.ethz.ch/datasets/doals

observing spatio-temporal changes in occupancy grids [15] with a two stage clustering and a ratio based validation check to filter outliers. Occupancy grid based detection requires an accurate pose estimate for each point cloud observation, but at the same time LiDAR odometry can be inaccurate in the highly dynamic environments where this approach would be used. Even though this problem renders an occupancy based approach not suitable for online filtering of dynamic objects during LiDAR SLAM, it allows us to label point clouds in an unsupervised manner. In contrast to the aforementioned semantic datasets, our labeling approach does not make any assumption on the type of dynamic objects in the scene.

We automatically label over 12000 LiDAR pointclouds in an environment with large amounts of pedestrians and use the resulting dataset to train 3D-MiniNet [6]. We leverage the trained network to predict dynamic objects in point clouds in real-time and integrate it into an existing LiDAR SLAM system [2] to create our dynamic object aware SLAM system. We show that by filtering points from dynamic objects online, we improve the relative translational odometry error by 39.6% and generate 3D reconstructions that contain drastically less non-static points as illustrated in Figure 1.

The main contributions of this work are:

- We present a full solution to creating a dynamic object aware LiDAR SLAM system, based on a deep neural network that performs online dynamic object detection.
- To solve the issue of generating training data we present a novel occupancy grid based approach, that includes a two stage clustering and validation step, to automatically label arbitrary dynamic objects in LiDAR point clouds.
- In real world experiments in highly dynamic urban environments we show a clear improvement to odometry and 3D reconstruction quality.

II. RELATED WORK

Various approaches exist to detect dynamic objects in known environments, *i.e.* places for which previously built maps already exist [16]–[19]. However, since these approaches assume prior knowledge of the environment, they are not suitable for online SLAM.

Approaches to dynamic object detection in unknown environments include the one proposed by Eppenberger *et al.* [20] that combines semantic detections with occupancy changes from an RGBD sensor to avoid moving obstacles. However, this approach is not directly applicable to 3D LiDAR point clouds, that lack visual information. Approaches such as [21], [22] successfully build static maps purely from LiDAR point clouds that contain dynamic objects. However, they use point clouds recorded at poses that are both temporally and spatially separated, which facilitates detection since dynamic objects move a lot in between point clouds. Our approach works on scan sequences, where dynamic objects move only slightly (or not at all) between subsequent point clouds. The ray-tracing approach by Yoon *et al.* [23] includes a clustering step, but the detections are often incomplete or contain static areas. Our proposed approach mitigates this, by instead including a two stage clustering designed to detect more complete objects and a validation step which removes clusters containing static areas. Dewan et al. [24] propose using rigid scene flow to detect dynamic objects. For slowly moving objects it is hard to distinguish scene flow from noise, while our approach is not dependent on the object velocity.

In contrast to the aforementioned approaches that assume poses as given, the approaches presented in [25], [26] perform localization and object detection jointly. They assume that objects can be tracked in subsequent scans [25] or rely on a minimum velocity assumption [26] which is often not suitable for crowds of pedestrians.

While the previous approaches are based on detecting actual motion, dynamic objects can also be detected based on their appearance by leveraging recent advances in deep learning. In urban scenarios it can be assumed that the most common dynamic objects are typically pedestrians, bicyclists and cars. These object classes can then be detected in point clouds using deep learned semantic segmentation methods [6], [9], [10], [27]–[29]. However, these approaches rely on manually labeled training data, and are limited to the subset of object types that exist in available datasets [12]–[14], [30]. Automatically generated labels are used in [31] to learn to detect dynamic cells, but the approach operates on 2D occupancy grids with static sensors and is thus not applicable to 3D point clouds from a moving LiDAR. Our approach makes it possible to create annotated 3D LiDAR datasets of arbitrary moving objects automatically, which extends the field of possible applications.

Systems which incorporate dynamic object awareness into LiDAR SLAM include the work of Ruchti and Burgard [32], which uses a neural network to predict dynamic objects but only excludes them from mapping. Other approaches use point-wise semantic information [5], [7], [8] to treat point matches differently depending on their class. As our approach classifies static and dynamic objects, we are able to remove dynamic features from the whole SLAM pipeline.

III. METHODOLOGY

We present a novel occupancy grid based pipeline to detect arbitrary dynamic objects in 3D LiDAR point clouds. This pipeline is used to generate an automatically labeled datasets of dynamic objects in an offline stage. We then train a 3D-MiniNet network on the resulting dataset to detect dynamic objects online. Using the deep learned online dynamic object detection we enable a dynamic object aware LiDAR SLAM pipeline, that by filtering out dynamic features achieves a more precise odometry and better 3D reconstructions of the environment. A diagram of the full pipeline is presented in Figure 2. Our contributions are focused on how we perform the occupancy grid based object detection and the proposed two stage clustering detailed in IV-C.

A. Occupancy Grid Based Dynamic Object Detection

Our proposed occupancy grid based dynamic object detection integrates successive point clouds into a global voxel grid and detects occupancy changes in this grid. In a first pass over the sequence, we acquire knowledge about all

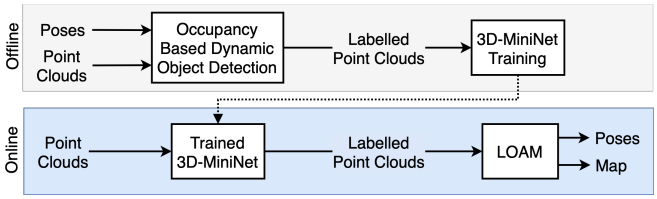

Fig. 2. Full pipeline for generic dynamic object aware LiDAR SLAM.

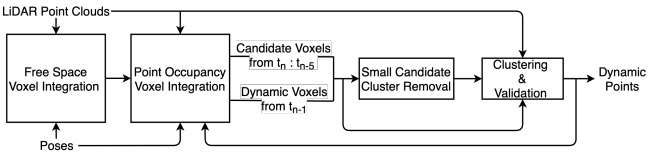

Fig. 3. Occupancy grid based dynamic object detection pipeline.

areas that have been free during the recording. We explicitly detect areas in which ray-tracing can lead to incorrect free voxels and avoid these. In the second pass, we integrate points into the previously acquired free space grid to observe occupancy changes. If the occupancy of a voxel changes from *free* to *occupied*, this indicates that an object might have moved into this voxel. We will refer to such voxels as candidate voxels. Purely classifying candidate voxels as dynamic points can lead to false positives and might not include all points belonging to dynamic objects. We propose a two stage clustering and validation step to filter out noise and better detect entire objects. A schematic of the occupancy based pipeline is given in Figure 3. The voxel integration is implemented using Voxblox [33] and a voxel size of $0.3\,\mathrm{m}$.

1) Voxel Integration:

a) Input: The input to the voxel integration contains the points recorded during one revolution of the LiDAR. We undistort the point cloud from the egomotion of the LiDAR by reprojecting each point using linear interpolation between the closest input poses at the pointwise timestamps.

b) Ray-Tracing: The initial state of all voxels is *unobserved*. To detect free space, we perform ray-tracing from the LiDAR origin to the observed points. If multiple points are observed in the same voxel, ray-tracing is only performed for the closest point to the LiDAR, to reduce the amount of redundant voxel traversals. If a voxel that is *occupied* by a currently observed point is encountered during ray-tracing, the traversal is stopped for this ray, since the area behind it is occluded. We detect areas in which naive ray-tracing can lead to incorrect *free* voxels. Such incorrect *free* voxels emerge if a ray appears along a surface that is occluded in the current scan. In this case, the voxels containing the surface would be falsely set to *free*, even though they contain objects in the real world. This results from ray-tracing through voxels that are partially occluded due to the limited grid resolution, as outlined in Figure 4. Schauer and Nüchter [22] use point normals to prevent ray-tracing through partially occluded voxels, which can be unreliable for sparse point clouds and is computationally expensive. In contrast, we explicitly detect voxels that are partially occluded using the range image. Partially occluded voxels arise if two points, p_1 and p_2, that are neighboring in the range image vary in range by more than $voxelsize$. In this case, the surface of the point closer to the LiDAR p_1, partially occludes the voxels between p_1 and p_2. Thus, we define a discontinuity as $r_2 - r_1 > voxelsize$, where r_n denotes the range of p_n with $r_2 > r_1$ by definition. In case of a discontinuity, all voxels further than r_1 along the ray of p_2 are set to *blocked*. If a *blocked* voxel is traversed, the state of it remains unchanged. We also block all voxels that are neighboring to voxels in which a point is currently observed to add robustness to noise in the point cloud or pose. All other traversed voxels are set to *free*.

c) Free Space Pass: If the integration would be performed by iterating over all point clouds only once in temporal order, objects that are moving into areas that are unobserved at recording time would remain undetected (e.g objects moving away in direction of the laser ray), since no assumption about the previous state could be made. Thus, we iterate over all point clouds twice. In the first pass, we start with an empty voxel grid and only allocate free space voxels by ray-tracing to acquire knowledge about all areas that have been free during the sequence.

d) Point Occupancy Pass: Starting with the previously created free space voxel grid, we subsequently integrate point clouds in the second pass with a better prior knowledge on free space, thus better detecting dynamic objects. During the point occupancy pass, the same ray-tracing as in the free space pass is performed, but voxels in which points are observed are set to *occupied*. A voxel is then added to the list of candidate voxels, if its state changes from *free* to *occupied*. Illustrations of examples are given in Figure 4. The area occupied by a dynamic object can overlap in subsequent scans if an object moves slowly or temporarily stops moving. In such cases, not all voxels occupied by the dynamic object would be detected as candidate voxels in two subsequent point clouds. Thus, for the n*th* LiDAR scan, at time t_n we do not only consider the candidate voxels obtained based on the previous point cloud, but also include candidate voxels from timesteps t_{n-5} to t_n. If points have already been detected as dynamic in t_{n-1}, points observed in t_n inside the respective voxels are considered candidate points if the voxel has been *free* at least once before. This requirement prevents the growth of false positive clusters over time.

2) Two Stage Clustering:

a) Ground Removal: Ground and ceiling points are removed from the candidate points by extending the approach proposed in [34]. It uses the range image to calculate an elevation angle between neighboring points in each column and assumes that each ground point segment is connected to a ground point pixel in the bottom row. If objects are close to the LiDAR as in our datasets, this assumption is violated. We overcome this issue by using a RANSAC plane fitting on all points with an elevation angle smaller than 30 degree. Restricting the plane fitting to this subset of points allows the use of a large distance threshold of $0.25\,\mathrm{m}$, which enables also detecting ground and ceiling points on slightly tilted or uneven surfaces. The inlier points are then used as seed points for the ground removal clustering proposed in [34]. This assumes that the LiDAR remains approximately parallel

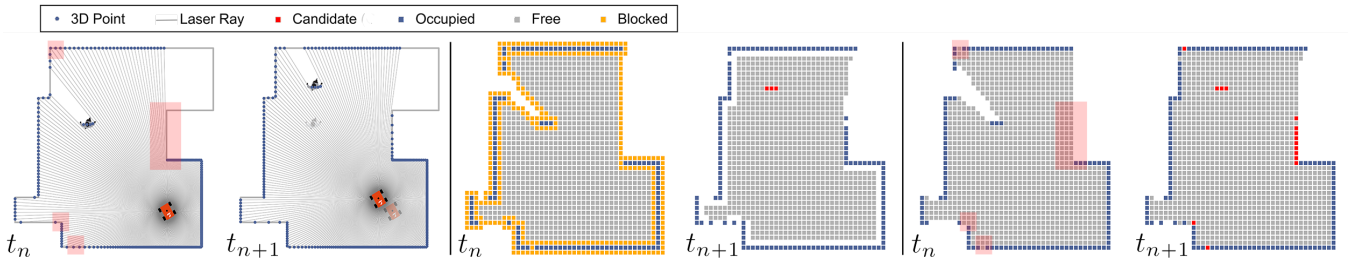

Fig. 4. *Left:* A simplified point cloud is projected on top of a scene with walls and a pedestrian. Two different timesteps are shown. Note that the wall in the large red shaded area is occluded in the first point cloud, which results in partially occluded voxels leading to false candidate voxels for a naive ray-tracing. *Middle:* Occupancy grid for our proposed approach. Voxels of the moved pedestrian are detected as candidates. Blocked voxels are not shown for the second timestep for clearness. *Right:* Occupancy grid for a naive ray-tracing, where multiple false candidate voxels can be observed in red shaded areas.

to the ground plane.

b) Clustering: Identified candidate points can contain false positive points from thin objects like thin poles or branches of trees, which are not reliably observed in each scan due to the sparsity of the point cloud. In a first stage a euclidean distance based clustering using a radius of $2 * voxelsize$ is performed on the candidate points. Only points in clusters with a diameter larger than $d = 0.2\,\mathrm{m}$ are added to the seed points for the next stage. The remaining seed points are used to find their respective clusters in the full point cloud. We use the range image based clustering approach proposed in [34]. Points that are detected as ground or ceiling can be added at the boundary of a cluster, but they are not used to continue region growing. This enables more complete object boarders close to the ground such as feet of pedestrians that are often detected as ground.

c) Candidate Cluster Validation: If a cluster is dynamic, the majority of its points should be detected as candidate points. Otherwise, candidate points inside the cluster might result from noise and not from a dynamic object. For each resulting cluster, we calculate the ratio of candidate points: $R_c = \frac{\#Candidate\ Points\ in\ Cluster}{\#Points\ in\ Cluster}$ and reject a cluster if R_c is lower than 0.6 or if the cluster contains less than 5 points. The parameters are hand-tuned for our sensor and scenario.

B. Dynamic Object Aware Lidar Odometry and Mapping

LOAM [2] is based on matching edge and plane features extracted by calculating the smoothness of the local surface in each point cloud. LOAM performs two separate scan matching steps. Scan-to-scan matching is performed between corresponding features in subsequent scans ($10\,\mathrm{Hz}$), while scan-to-map matching ($1\,\mathrm{Hz}$) is performed between features of the current scan and features of the environment map that is gradually built. Scan-to-map matching is more accurate, but also computationally more expensive. The detailed algorithm is found in [2]. We use a custom implementation of their work, that we refer to as *standard LOAM*. LOAM is based on the assumption, that most of the features used for matching are fixed in space, which is violated by dynamic objects.

We use the trained network described in III-C to estimate dynamic object points. Simply removing all dynamic points prior to feature detection leads to artificial edges, that are then detected as features. Instead, we detect all features and classify them as static or dynamic. A feature is considered static, if all points that contribute to it were classified as static. To maintain the assumption of a static environment, features classified as dynamic are neither used for feature matching nor added to the map. We refer to this approach as *dynamic object aware LOAM*.

C. 3D-MiniNet Training

3D-MiniNet was proposed by Alonso et al. and performs state of the art semantic segmentation for 3D LiDAR point clouds. It consists of two main modules: The projection learning module learns a 2D representation from the x, y, z, intensity and range value of each point, that is then fed into the fully convolutional MiniNet [35] backbone that predicts a semantic label for each point. With a runtime of $36\,\mathrm{fps}$ on an Nvidia RTX 2080 Ti GPU, it operates faster than the recording frequency of the sensor, which is crucial for real-time operation. We use label smoothing [36] to make training more robust to false annotations that are present in our dataset. We horizontally flip the images with a probability of 0.5 and adapt the x and y values accordingly to augment the training data. We train the network using Adam optimizer [37] for 35 epochs with a learning rate of 0.0003 and batch size 3.

IV. Evaluation

We evaluate each component of our proposed pipeline as well as the entire end-to-end process. In a simulated environment with ground truth annotations we show that the occupancy grid based dynamic object detection is applicable to a multitude of different types of dynamic objects. We annotated a subset of our real world dataset that we present in IV-B, to evaluate the segmentation performance of the occupancy grid based detection methods as well as of the 3D-MiniNet trained on it. Finally, we compare the performance of a standard LOAM pipeline with our dynamic object aware LOAM. We evaluate all 4 locations of the dataset using the same settings. At each location, 2 sequences are evaluated. Trajectory lengths vary between $100 - 400\,\mathrm{m}$ and sequences last between $100 - 200\,\mathrm{s}$. To evaluate 3D-MiniNet at one location, the network is trained on the data of the 3 remaining locations, e.g. we train on data from *Hauptgebaeude*, *Shopville* and *Station* to evaluate at *Niederdorf*. Experiments were run on an Intel I7-8559U CPU and using an Nvidia Geforce RTX 2080 Ti GPU.

Fig. 5. Occupancy grid based detection examples shown on the intensity image generated from the point clouds. Detected dynamic objects are colored in red. *Top* to *Bottom*: *Niederdorf, Hauptgebaeude, Shopville, Station.*

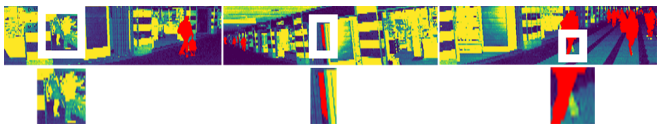

Fig. 6. Examples of erroneous annotations in the dataset: *Left:* Missed detection from under-segmentation: Humans are close to a shelf and end up in a cluster with it. *Middle:* False positive resulting from reflective surfaces. *Right:* The object is detected, but the lower leg is missing.

Fig. 7. 3D-MiniNet prediction examples at the four different locations. For each location, the training was performed on the automatically labeled data of the 3 remaining locations.

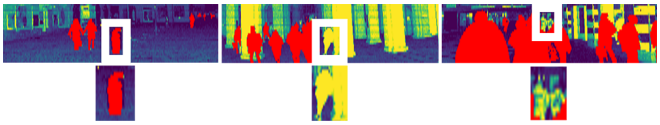

Fig. 8. Erroneous 3D-MiniNet Prediction Results: *Left:* False positive detection on a trash bin. *Middle:* Missed detection on a person leaning onto a pillar with similar intensity values. *Right:* False negative prediction on two pedestrians in a cluttered group.

A. Simulated Dataset

We evaluate the performance of the occupancy grid based pipeline on a wide variety of arbitrary objects in a simulated environment of a small town. The environment contains moving cars, planes, pedestrians, animals, cylinders, spheres and cubes at different sizes that are moving horizontally and vertically at different velocities and in different directions. The simulated sensor is moving in a closed trajectory through the environment. Ground truth annotations for all moving objects for each of the 1642 point clouds are available.

B. Real World Dataset

We recorded a total of more than 12000 scans in the main hall of ETH Zurich (*Hauptgebaeude*), at two different levels of the main train station in Zurich (*Station*, *Shopville*) and in a touristic pedestrian zone (*Niederdorf*). Examples of the collected point clouds are shown in Figure 5. A handheld Ouster OS1 64 LiDAR and a Alphasense Core multi-camera module sensor[3] were used for recording. Point clouds are recorded at 10 Hz with 2048 points per revolution. In addition a VI-SLAM pipeline is run on the Sevensense sensor data to obtain high frequency pose estimates.

We use the pipeline presented in III-A to automatically label dynamic objects in LiDAR point clouds. For evaluation purposes we manually annotated a subset of our dataset. Pedestrians and objects associated to them (e.g suitcases, bicycles, dogs) were annotated for 10 temporally separated point clouds for each of the 8 sequences. Pedestrians were annotated by appearance only, thus it was not considered if they are static or moving.

C. Occupancy Grid Based Dynamic Object Detection Results

The occupancy grid based dynamic object detection achieves an Intersection over Union (IoU) of 0.92 for moving objects averaged over all sequences of our simulated environment. This shows, that our approach is applicable to a wide variety of objects of different dimensions and shapes and is also not restricted to a certain type of motions. Missing detections are caused by thin or far away objects, because their resulting clusters fall below the minimum required amount of points, due to the point cloud sparsity.

On our real world dataset, an IoU of 0.88 is achieved averaged over all locations on the annotated test set. The error is partly due to ground truth annotation that are based on appearance. Some sequences contain pedestrians that do not move and thus are not detected by the approach. Detailed results are provided in Table I. As shown in Figure 5 the vast majority of pedestrians are detected, also if they are partly occluded or close to the LiDAR. Detection fails in some cases if a pedestrian is close to a static object, as points belonging to the pedestrian end up in a cluster with the static object and thus the cluster validation is not passed. In rare cases small parts like a leg of a pedestrian remain undetected due to over-segmentation. The detection performance decreases with distance from the sensor, due to the increasing point sparsity. A limitation of LiDAR sensors is that reflective surfaces can cause invalid distance measurements, that lead to invalid voxel states and erroneous annotations. Examples of erroneous annotations are given in Figure 6. The detection takes 1.2 s on average per point cloud.

D. 3D-MiniNet Based Dynamic Object Detection Results

We achieve an IoU of 0.82 for the pedestrian class, averaged over all locations on the manually obtained ground truth annotations for the real world dataset. Results for the

TABLE I

PEDESTRIAN IOU ON TEST SET AT DIFFERENT LOCATIONS

Method \ Location	Station	Shopville	Hauptgebaeude	Niederdorf
Occupancy Grid	0.91	0.85	0.88	0.87
3D-MiniNet	0.84	0.82	0.82	0.8

[3] https://github.com/sevensense-robotics/alphasense_core_manual

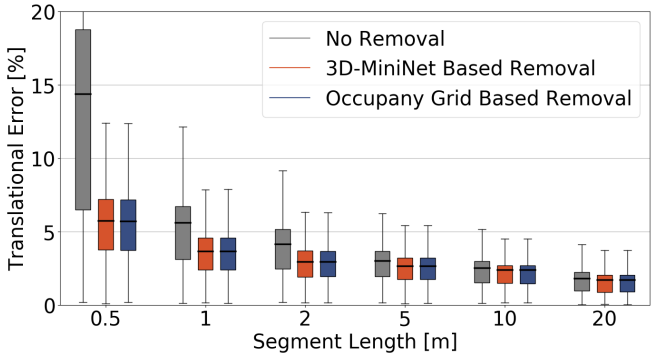

Fig. 9. Relative trajectory errors for different segment lengths. Mean values are indicated by black bars.

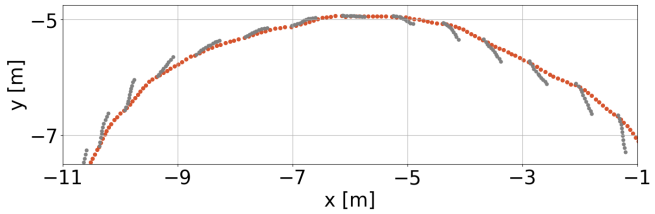

Fig. 10. Trajectory segment of sequence *Station-2*: The trajectory from our standard LOAM implementation(grey) contains several gaps, in contrast to the trajectory obtained from the dynamic object aware LOAM (orange).

individual locations are provided in Table I. Examples of the detections are given in Figure 7. The performance decreases with increasing point distance, as is to be expected, as more missing annotations are present at higher distance.

E. Odometry Results

We compare the trajectories resulting from dynamic object aware LOAM to standard LOAM. We also show the results that would be achieved by using dynamic object aware LOAM with the pointwise labels from the occupancy based dynamic object detection as a reference. This pipeline can not be used online, as the occupancy based detection relies on previously known pose estimates and free space knowledge acquired by future observation. We use the globally bundle adjusted poses of the VI-SLAM pipeline as a ground truth reference to evaluate the odometry.

We calculate the relative trajectory error as in [30], for overlapping segments of 0.5, 1, 2, 20 and $20\,\mathrm{m}$, which we deem representative for applications in the given environments. We average the respective errors over all sequences.

Estimating odometry using the dynamic object aware LOAM improves the relative translational error on all segments lengths compared to standard LOAM. Averaged over all segments, 39.6% less translational drift is achieved. The results are presented in more detail in Figure 9. The translational error decreases especially for shorter segments. This mainly results from drift in the standard LOAM approach during scan-to-scan steps. This drift is mainly compensated during scan-to-map steps and thus has a lower effect on longer segments. Looking qualitatively at the trajectories we observe that the bias in the drift correlates with the general direction of movement of nearby people in the dataset. This is also reflected in the trajectories estimated by

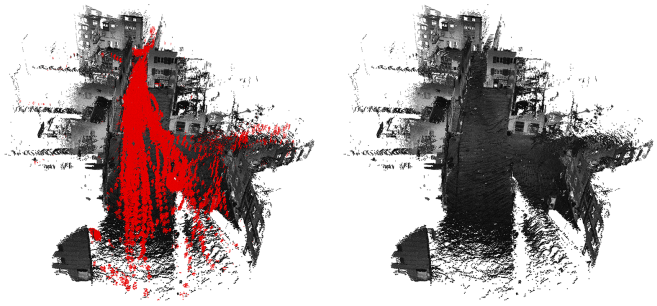

Fig. 11. Resulting map for sequence *Niederdorf-1*: *Left:* Points classified as *dynamic* are shown in red, *static* points are shown in grayscale. *Right:* The cleaned map without points classified as *dynamic* in grayscale.

both methods as shown in an example trajectory segment in Figure 10. It can be seen that the standard LOAM has very clear low frequency discontinuities in the odometry of $0.14\,\mathrm{m}$ on average, whenever the more accurate scan matching is performed. This non smooth odometry poses significant disadvantages when used for navigation or obstacle avoidance. In contrast, the trajectory of dynamic object aware LOAM is much smoother and reduces the low frequency jumps to $0.04\,\mathrm{m}$ on average, making it a much better candidate for use on a robotic platform. The relative rotational error remained approximately equal across approaches in our experiments.

F. Mapping

Maps presented in Figures 11 and 1 were built by aggregating point clouds using poses obtained from the scan-to-map matching steps and filtering out points that are further than $30\,\mathrm{m}$ from the LiDAR. The maps were subsampled using a $0.1\,\mathrm{m}$ voxel grid. Removing dynamic objects makes the static structure in the scene far more clearly observable.

V. Conclusions

We presented a complete solution for creating a dynamic object aware LiDAR SLAM pipeline, that is based on a deep learned dynamic object filtering step. To this end we proposed a novel occupancy grid based approach to automatically label arbitrary dynamic objects in point clouds offline, to efficiently create training data for any dynamic environment. We leveraged our method to automatically annotate a large amount of pedestrians and other dynamic objects at four distinct, highly dynamic, urban locations in more than 12000 real world LiDAR point clouds. The dataset is then used to train a 3D-MiniNet neural network to segment out dynamic objects in real-time and enhance the performance of LOAM by removing these objects from the point clouds before the matching and mapping process. This improves odometry by reducing drift on average by 39.6% and also significantly smoothing out the trajectory estimate. In addition we are able to create much better and accurate 3D scene reconstructions. Even though we showcase our proposed pipeline using 3D-MiniNet and LOAM as well as a dataset with a significant amount of pedestrians, our proposed methods are generic and equally applicable to a large variety of dynamic objects, segmentation methods and LiDAR SLAM algorithms.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard, "Simultaneous localization and mapping: Present, future, and the robust-perception age," *CoRR*, vol. abs/1606.05830, 2016. [Online]. Available: http://arxiv.org/abs/1606.05830

[2] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, no. 2, p. 401–416, February 2017.

[3] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "An online multi-robot slam system for 3d lidars," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1004–1011.

[4] 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)*, 2018, pp. 4758–4765.

[5] S. Zhao, Z. Fang, H. Li, and S. Scherer, "A robust laser-inertial odometry and mapping method for large-scale highway environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1285–1292.

[6] I. Alonso, L. Riazuelo, L. Montesano, and A. C. Murillo, "3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[7] X. Chen, A. M. E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4530–4537, 2019.

[8] Z. Zhao, W. Zhang, J. Gu, J. Yang, and K. Huang, "Lidar mapping optimization based on lightweight semantic segmentation," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 353–362, 2019.

[9] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet ++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4213–4220.

[10] 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," *CoRR*, vol. abs/1809.08495, 2018.

[11] J. Deschaud, "IMLS-SLAM: scan-to-model matching based on 3d data," *CoRR*, vol. abs/1802.08633, 2018. [Online]. Available: http://arxiv.org/abs/1802.08633

[12] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," 2019.

[13] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[14] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[15] L. Wellhausen, R. Dubé, A. Gawel, R. Siegwart, and C. Cadena, "Reliable real-time change detection and mapping for 3d lidars," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2017, pp. 81–87.

[16] I. Jinno, Y. Sasaki, and H. Mizoguchi, "3d map update in human environment using change detection from lidar equipped mobile robot," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, 2019, pp. 330–335.

[17] X. Ding, Y. Wang, H. Yin, L. Tang, and R. Xiong, "Multi-session map construction in outdoor dynamic environment," in *IEEE International Conference on Real-time Computing and Robotics, RCAR 2018, Kandima, Maldives, August 1-5, 2018*. IEEE, 2018, pp. 384–389.

[18] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3d map maintenance in dynamic environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3712–3719.

[19] P. Drews, S. C. da Silva Filho, L. F. Marcolino, and P. Núñez, "Fast and adaptive 3d change detection algorithm for autonomous robots based on gaussian mixture models," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 4685–4690.

[20] T. Eppenberger, G. Cesari, M. Dymczyk, R. Siegwart, and R. Dubé, "Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking," IEEE, 2020.

[21] J. P. Underwood, D. Gillsjö, T. Bailey, and V. Vlaskine, "Explicit 3d change detection using ray-tracing in spherical coordinates," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 4735–4741.

[22] J. Schauer and A. Nüchter, "The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1679–1686, 2018.

[23] D. J. Yoon, T. Y. Tang, and T. D. Barfoot, "Mapless online detection of dynamic objects in 3d lidar," *2019 16th Conference on Computer and Robot Vision (CRV)*, pp. 113–120, 2019.

[24] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Rigid scene flow for 3d lidar scans," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1765–1770, 2016.

[25] F. Moosmann and C. Stiller, "Joint self-localization and tracking of generic objects in 3d range data," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 1138–1144.

[26] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Motion-based detection and tracking in 3d lidar scans," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4508–4513.

[27] 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1887–1893.

[28] P. Biasutti, V. Lepetit, J. Aujol, M. Brédif, and A. Bugeau, "Lu-net: An efficient network for 3d lidar point cloud semantic segmentation based on end-to-end-learned 3d features and u-net," in *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*. IEEE, 2019, pp. 942–950.

[29] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving," 2020.

[30] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[31] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2056–2063.

[32] P. Ruchti and W. Burgard, "Mapping with dynamic-object probabilities calculated from single 3d range scans," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6331–6336.

[33] H. Oleynikova, Z. Taylor, M. Fehr, J. I. Nieto, and R. Siegwart, "Voxblox: Building 3d signed distance fields for planning," *CoRR*, vol. abs/1611.03631, 2016.

[34] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3d laser scans for online operation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 163–169.

[35] I. Alonso, L. Riazuelo, and A. C. Murillo, "Mininet: An efficient semantic segmentation convnet for real-time robotic applications," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1340–1347, 2020.

[36] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" *CoRR*, vol. abs/1906.02629, 2019.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980