Kumulative Habilitationsschrift
zur Erlangung der der Lehrbefähigung
im Fachgebiet Photogrammetrie und Robotik
eingereicht an der Landwirtschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

Towards LiDAR-based Spatio-temporal Scene Understanding for Autonomous Vehicles

von

Jens Behley

aus

Cottbus, Germany

Bonn 2026

Datum des Habilitationskolloquiums: 07.06.2023

Summary

Self-driving cars are expected to reduce the number of casualties caused by traffic accidents, since a machine is always attentive, it can exploit various input modalities, and it always obeys the traffic rules. Liberating people from driving a vehicle will also enable them to do more pleasant activities while getting from one place to another. A fleet of self-driving cars could also lead to less parked cars in the city as cars could efficiently shared and be available on-demand. All these prospects of self-driving cars led to an increasing activity in this area of research and many large automotive companies invested substantially in the research and development of self-driving cars.

A central aspect of self-driving cars is perception to make sense of the different sensory inputs available. Most self-driving car prototypes rely on a combination of different sensors, such as cameras and 3D LiDAR sensors. In particular, 3D LiDAR sensors provide accurate and dense depth measurements of the environment. Since the advent of fast 3D LiDAR sensors that can produce millions of measurements of the 360° field-of-view, research on 3D LiDAR-based perception attracted increasing attention in the recent years.

In this habilitation thesis, we present our contributions in the area of 3D LiDAR-based perception. We cover our work on 3D LiDAR-based spatial perception to enable an autonomous system to localize itself in the environment. We present our approaches for Simultaneous Localization and Mapping (SLAM) for building maps on-the-fly, localization using existing maps, mapping to generate detailed maps, and map compression to efficiently transfer mapping data.

Furthermore, we cover our approaches for semantic interpretation of a single 3D LiDAR scan. All the presented work in semantic perception is based on our dataset, SemanticKITTI, that provides the data needed to train machine learning approaches for semantic interpretation. Furthermore, we present our work on semantic segmentation and panoptic segmentation. Additionally, we present our approach to reduce the need for labeled data.

Lastly, we cover also our work on unifying spatial and semantic interpretation in the area of spatio-temporal interpretation. In this part, we present our approach for moving object segmentation using a sequence of 3D LiDAR scans. We present our approach for semantic SLAM that use semantic information to improve pose estimation. Lastly, we present our work on panoptic segmentation on a sequence of 3D LiDAR scans that provides spatio-temporal interpretation.

Acknowledgements

The presented body of work is the product of a fruitful research collaboration and would not have been possible without the many PhD students of the Photogrammetry & Robotics lab, collaborators of the computer science department of the University of Bonn and Technical University of Munich that I had the pleasure to work with.

In particular, I thank Prof. Dr. Cyrill Stachniss who gave me the opportunity to work in his lab, supported all my research endeavors, and also allowed me to jointly co-supervise many of his aforementioned PhD students of the Photogrammetry & Robotics lab. Cyrill opened many doors and I profited heavily from the infrastructure, advice on my career, and could certainly learn many things on how to lead and build a successful research group.

Contents

Summary **i**

Contents **v**

Publication List of the Cumulative Habilitation **vii**

1 Introduction **1**
1.1 Contributions 3
1.2 Organization of the Habilitation Thesis 5

2 Localization and Mapping using 3D LiDAR Sequences **9**
2.1 Efficient Surfel-based SLAM 10
2.2 Adaptive Robust Kernels 12
2.3 Mesh-Based Mapping 13
2.4 Map Compression 15
2.5 Overlap-Based Place Recognition 16
2.6 Overlap-Based Localization 18
2.7 Mesh-Based Localization 19

3 Semantic Interpretation of 3D LiDAR Scans **21**
3.1 A Large-scale Dataset for Semantic Interpretation 22
3.2 3D Semantic Segmentation 24
3.3 3D Panoptic Segmentation 25
3.4 Beyond Supervised Learning 26

4 Spatio-temporal Scene Interpretation of 3D LiDAR Sequences **29**
4.1 Moving Object Segmentation 30
4.2 Semantic SLAM using 3D LiDAR Sequences 32
4.3 Panoptic Segmentation of 3D LiDAR Sequences 33

5 Conclusion **37**

Publications **48**

Publication List of the Cumulative Habilitation

Peer-reviewed journal articles

[A] R. Marcuzzi, L. Nunes, L. Wiesmann, I. Vizzo, J. Behley, and C. Stachniss, "Contrastive Instance Association for 4D Panoptic Segmentation using Sequences of 3D LiDAR Scans," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 1550–1557, 2022. DOI: 10.1109/LRA.2022.3140439.

[B] L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss, "SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segement Discrimination," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 2116–2123, 2022. DOI: 10.1109/LRA.2022.3142440.

[C] J. Behley, M. Garbade, A. Milioto, *et al.*, "Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset," *Intl. Journal of Robotics Research (IJRR)*, vol. 40, no. 8-9, pp. 959–967, 2021. DOI: 10.1177/02783649211006735.

[D] N. Chebrolu, T. Läbe, O. Vysotska, J. Behley, and C. Stachniss, "Adaptive Robust Kernels for Non-Linear Least Squares Problems," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2240–2247, 2021. DOI: 10.1109/LRA.2021.3061331.

[E] X. Chen, T. Läbe, A. Milioto, T. Röhling, J. Behley, and C. Stachniss, "OverlapNet: A Siamese Network for Computing LiDAR Scan Similarity with Applications to Loop Closing and Localization," *Autonomous Robots (AURO)*, vol. 46, pp. 61–81, 2021. DOI: 10.1007/s10514-021-09999-0.

[F] X. Chen, S. Li, B. Mersch, *et al.*, "Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 4, pp. 6529–6536, 2021. DOI: 10.1109/LRA.2021.3093567.

[G] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss, and J. Behley, "Deep Compression for Dense Point Cloud Maps," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2060–2067, 2021. DOI: 10.1109/LRA.2021.3059633.

Peer-reviewed conference papers

[H] M. Aygun, A. Osep, M. Weber, *et al.*, "4D Panoptic LiDAR Segmentation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. DOI: 10.1109/CVPR46437.2021.00548.

[I] J. Behley, A. Milioto, and C. Stachniss, "A Benchmark for LiDAR-based Panoptic Segmentation based on KITTI," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021. DOI: 10.1109/ICRA48506.2021.9561476.

[J] X. Chen, I. Vizzo, T. Läbe, J. Behley, and C. Stachniss, "Range Image-based LiDAR Localization for Autonomous Vehicles," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021. DOI: 10.1109/ICRA48506.2021.9561335.

[K] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, "Poisson Surface Reconstruction for LiDAR Odometry and Mapping," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021. DOI: 10.1109/ICRA48506.2021.9562069.

[L] X. Chen, T. Läbe, A. Milioto, *et al.*, "OverlapNet: Loop Closing for LiDAR-based SLAM," in *Proc. of Robotics: Science and Systems (RSS)*, 2020. DOI: 10.15607/RSS.2020.XVI.009.

[M] X. Chen, T. Läbe, L. Nardi, J. Behley, and C. Stachniss, "Learning an Overlap-based Observation Model for 3D LiDAR Localization," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020. DOI: 10.1109/IROS45743.2020.9340769.

[N] A. Milioto, J. Behley, C. McCool, and C. Stachniss, "LiDAR Panoptic Segmentation for Autonomous Driving," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020. DOI: 10.1109/IROS45743.2020.9340837.

[O] J. Behley, M. Garbarde, A. Milioto, *et al.*, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019. DOI: 10.1109/ICCV.2019.00939.

[P] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based Semantic SLAM," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019. DOI: 10.1109/IROS40897.2019.8967704.

[Q] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019. DOI: 10.1109/IROS40897.2019.8967762.

[R] J. Behley and C. Stachniss, "Efficient Surfel-based SLAM using 3D Laser Range Data in Urban Environments," in *Proc. of Robotics: Science and Systems (RSS)*, 2018. DOI: 10.15607/RSS.2018.XIV.016.

Chapter 1

Introduction

Driving cars to reach a desired destination is still one of the main modes of transportation and accidents involving cars cause monetary damages or even casualties every day. Most traffic accidents are caused by human error and could be potentially avoided by letting computers take over the steering wheel, since a machine is always attentive, can leverage various input modalities, and it always obeys the traffic rules. Liberating people from driving a vehicle would also enable them to do more pleasant activities while getting from one place to another. Furthermore, self-driving cars do not need to be parked somewhere and therefore could always drive themselves to the place where they are needed, which could lead to environmental benefits as less cars are required to enable individual mobility.

Since fully autonomous cars crossed the finishing line at the DARPA Urban Challenge in 2007, where participants were tasked to drive autonomously in a semi-static urban-like environment, the aforementioned prospects of cars that drive themselves in everyday urban traffic seemed in close reach. Many startups were founded pursuing the promise of fully autonomous vehicles and large established automotive cooperations invest heavily to make this dream reality. However, the gap between the DARPA Urban Challenge and real urban traffic turned out to be larger than expected and ambitious announcements of the arrival of fully autonomous cars could not be held so far.

The road to vehicle autonomy is characterized by levels in the SAE J 3016-2021 ranging from Level 0, which means no driving automation, to Level 5, which means full driving automation, where the vehicle takes all responsibilities and a human driver is not needed anymore. While autonomous taxis operating at Level 4 autonomy in geofenced locations are practically available, see Figure 1.1, and certain driving capabilities can be taken over by the car nowadays, such as driving in a traffic jam, i.e., Level 3 autonomy, these cars are still far from operating in everyday traffic in every situation without any interaction by a human driver, i.e., Level 5 autonomy.

While control, planning, collision avoidance and navigation is already matured that much that it can be reliably executed [1], [2] and we can design algorithmic solutions via programming such components, the crucial ingredient of perception is still not handled at a level that is needed to reliably interpret and understand arbitrary traffic situa-



(a) (b)

Figure 1.1: Academic prototypes at the DARPA Urban Challenge (a), here CMU's entry "Boss" that crossed the finishing line, paved the way for an emerging self-driving car industry, where prototypes like the shown Waymo's One (b) already operate in designated geo-fenced areas as a driverless taxi service. However, these self-driving taxis still need sometimes interventions by human operators when unforeseen situations arise and the car cannot handle these situations. (Picture of "Boss" courtesy of Urmson et al. [2], Waymo picture provided by courtesy of Waymo.)

tions [3]. This is mainly caused by the variety of situations and the conditions that a vehicle might encounter in unconstrained, natural environments. While certain objects occur relatively often in every day traffic situations, certain classes, such as emergency vehicles, or unforeseen objects appear only rarely. However, a robust perception system must recognize or handle such objects even though there might only be a few training examples in the dataset to train the machine learning approach responsible for performing interpretation of the sensor data.

Most prototypes of self-driving cars feature a range of sensors with complementing characteristics for perceiving the surrounding vicinity of the car, but also the internal state of the car. Cameras enable an autonomous vehicle to perceive the world using high-resolution color images, but are severely affected by lighting conditions. Light Detection and Ranging (LiDAR) sensors provide a sparse 3D point cloud of distance measurements and are unaffected by lighting conditions, but can be impaired by weather conditions or reflectance properties of the objects. RADARs provide locations and velocities of objects using the Doppler effect, but measurements are very sparse and potentially affected by noise. Odometer and inertial measurement units (IMU) provide information about the motion and location of the vehicle, but can drift over time. Generally, GPS can provide the location of a vehicle, but can get inaccurate or unreliable if satellites are occluded.

A key component of the perception stack of most prototypes of autonomous cars is the LiDAR sensor that actively and accurately measures distances to objects in the surrounding complementing cameras that can perceive high-resolution images of the surrounding with fine-grained visual details. Most of the prototypes for autonomous driving are equipped with both sensors that seem essential to achieve reliable perception for autonomous driving.

While visual perception has advanced significantly in the recent years with the advent of deep learning [4], [5], the field of 3D LiDAR-based perception is relatively

young and there are still several open questions, like how to best represent and process this kind of data that shows a distance-dependent sparsity. Furthermore, acquiring data for training 3D LiDAR-based perception systems is comparably hard due to the price of the sensors and therefore large datasets are relatively rare compared to the abundance of images that can be crawled from the internet.

Before we dive into the contributions of this habilitation thesis, we first have to be aware of the terminology to distinguish *interpretation* and *understanding* of a traffic scene as used in this habilitation thesis. While we refer to interpretation as the process of identifying objects in the scene, understanding refers to the process that additionally captures the relations, intent, and goals of the identified objects. In context of autonomous driving, interpretation identifies the objects and their locations at the current moment, while understanding additionally involves reasoning about the intent of the other actors, but also the relation of the objects in the scene. Scene understanding facilitates prediction that enables to accurately account for future locations and intents of all traffic participants. Thus, scene interpretation is necessary to attain scene understanding. But scene understanding will ultimately enable self-driving cars to reach Level 5 autonomy as it involves also the capability to asses a unforeseen situation and the ability to handle such situations reasonably.

In this habilitation thesis, we will summarize our contribution towards semantic scene understanding using 3D LiDAR sensors. Starting with the aspect of spatial interpretation, i.e., determining the vehicles own pose in relation to the environment. We then focus on the aspect of semantic interpretation of a single 3D LiDAR scan, where we are interested in the semantics of a given point cloud. Finally, we integrate both aspects and provide an overview of our contributions for spatio-temporal interpretation in the last part.

1.1 Contributions

In this thesis, we present contributions towards the goal of the spatio-temporal semantic scene understanding of 3D LiDAR sequences. The presented research contributed to the advancement of the field of 3D LiDAR-baser perception in multiple aspects ranging from spatial interpretation to semantic interpretation to spatio-temporal interpretation. The accumulated work presented in this thesis made important steps in the direction of spatio-temporal scene understanding and the proposed datasets and novel task definitions pushed the boundaries of the field. In more detail, our contributions to the field of 3D LiDAR-based perception can be summarized as follows:

LiDAR-Based Simultaneous Mapping and Localization (SLAM). We present a real-time capable, surfel-based SLAM system (SuMa) [R] that enables to determine the globally consistent location of a vehicle equipped with a rotating 3D LiDAR sensor. It allows to estimate the relative pose using 3D point clouds in respect to a map represented by surfels and it furthermore is able to determine if places have be revisited to initiate a pose graph optimization to reduce accumulated errors by integrating loop closures. Consequently, our SLAM system provides globally consistent maps of

an unseen urban environment.

Adaptive Robust Kernels for Odometry estimation. We propose an approach [D] for adaptively determining a robust kernels for odometry estimation that enables to reduce outliers in the correspondences for odometry estimation. In situations with other moving vehicles, we are able to more robustly and accurately estimate relative poses in frame-to-frame odometry estimation.

LiDAR-Based Mapping. We propose an approach [K] based on Poisson Surface reconstruction that leads to more detailed maps represented as a triangle mesh with the potential to replace other more memory-consuming representations, such as a truncated signed distance function (TSDF) [6] or surfels [7].

Point cloud compression. We propose a deep-learning approach for compressing point cloud maps that could be used to transfer point clouds at a fraction of the size of the original points.

LiDAR-Based Loop Closure Detection. We develop an deep learning-based approach [E], [L], called OverlapNet, that is able to determine if places have been revisited. This so-called loop closures can be then employed to provide a globally consistent map by a pose graph optimization.

LiDAR-Based Localization. We develop systems [J], [M] based on the concept of overlap between scans for more effective place recognition for finding loop closures that ensures that correct loop closures are found for pose graph optimization.

Dataset and Benchmarks. With SemanticKITTI [O], we provide the first large-scale dataset with point-wise annotations for point cloud sequences produced by an automotive LiDAR sensor. We later extended the dataset with instance annotations for traffic participants [I]. Several competitions evaluate the progress of scene interpretation of a single point cloud, like semantic segmentation and panoptic segmentation, but also spatio-temporal scene interpretation using multiple point clouds, such as 4D panoptic segmentation [H] and moving object segmentation [F].

LiDAR-Based Semantic Segmentation. Based on our SemanticKITTI dataset, we propose one of the first semantic segmentation approaches, called RangeNet++ [Q] that provides a dense semantic interpretation of the environment including semantic classes for drivable space, but also traffic participants.

LiDAR-Based Panoptic Segmentation. Based on the semantic and instance annotation of SemanticKITTI [I], we propose the first approach for LiDAR-based panoptic segmentation [N] not only providing dense semantic class predictions, but also separating different instances of the same object class of traffic participants.

Self-Supervised Representation Learning. In our work on self-supervised representation learning [B], we propose a segment-based contrastive learning approach that enables us to learn with less labeled data. This allows us to considerably reduce the labeling time needed to train supervised approaches.

Moving Object Segmentation. Our approach for moving object segmentation [F] proposes a novel benchmark for segmentation of moving vs. static objects. Our method for LiDAR-based motion segmentation can be applied for mapping of the just the static environment and leads to superior pose accuracy in SLAM approaches

Section	Topic	Publications
2.1	SLAM	[R]
2.2	Adaptive Robust Kernels	[D]
2.3	Mapping	[K]
2.4	Map Compression	[G]
2.5	Place Recognition	[L]
2.6	Localization	[M], [E]
2.7	Localization	[J]
3.1	Datasets	[C], [I], [O]
3.2	Semantic Segmentation	[Q]
3.3	Panoptic Segmentation	[N]
3.4	Self-supervised Representation Learning	[B]
4.1	Moving Object Segmentation	[F]
4.2	Semantic SLAM	[P]
4.3	4D Panoptic Segmentation	[H], [A]

Table 1.1: Topics covered in the thesis and associated papers.

by filtering correspondences of moving objects that might have adversarial effects.

Semantic SLAM. We propose one of the first approaches for semantic SLAM, called SuMa++ [P], that exploits dense semantic segmentation [Q] that leads to more accurate pose estimation results than the geometry-only SuMa [R].

4D Panoptic Segmentation. We formalized the novel task of 4D LiDAR panoptic segmentation [H] providing temporally consistent instance labels identifying the same object class instance over time. We proposed also a contrastive approach for learning a network that provides features for clustering temporally consistent instances [A]

Overall, we contributed with the work presented in this thesis to the substantial progress in 3D LiDAR-based semantic scene interpretation, but also opened the door for spatio-temporal scene interpretation by developing novel tasks and pushing the research boundaries in LiDAR-based perception in context of autonomous driving.

1.2 Organization of the Habilitation Thesis

Before we dive into details, we want to provide an overview of the covered topics and link the sections to respective published papers that are the basis for this cumulative habilitation thesis. In Table 1.1, we provide an overview of the covered topics in the upcoming sections but also references to the respective papers.

In the first part, Chapter 2, we cover the spatial aspect of the spatial-temporal scene interpretation, where we cover how a self-driving car is able to determine the relative poses over time while driving and the global localization to localize the car in a given map. The spatial relation between scans is an important stepping stone for the subsequent sections, where we exploit the pose information to build a dataset for

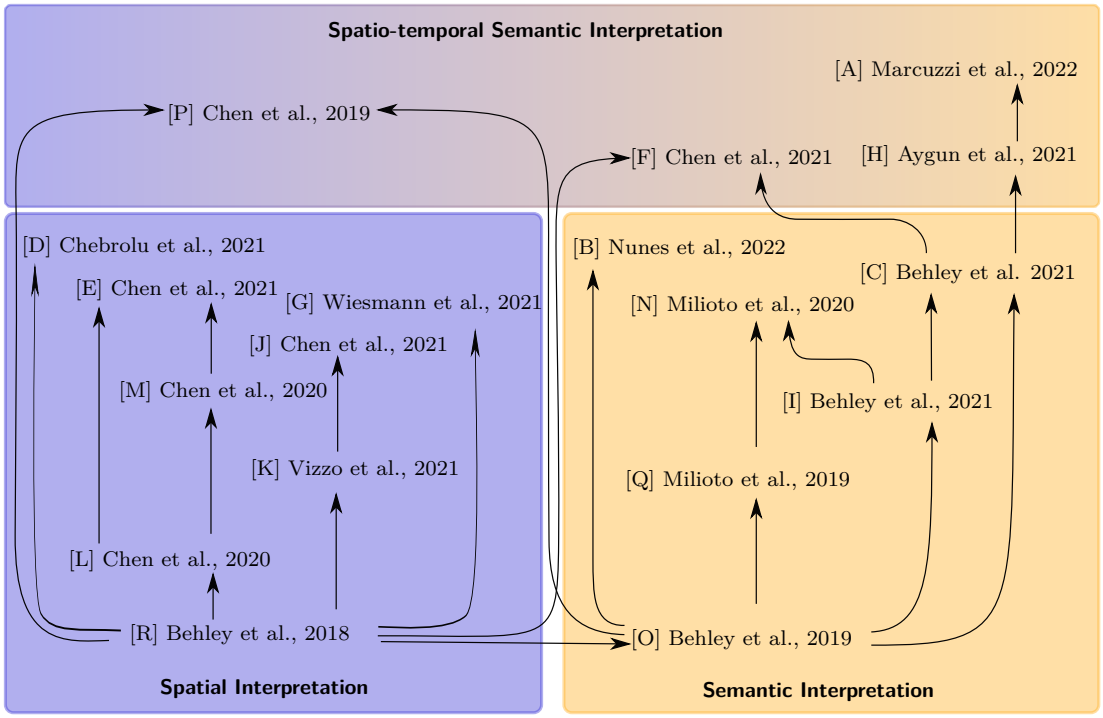


Figure 1.2: Dependencies and relations between the papers and contributions that are part of the habilitation thesis.

learning models to perform scan-wise scene interpretation, but also use this information for spatio-temporal scene interpretation.

In the next part, Chapter 3, we cover our contributions regarding the data needed to learn a model that is able to provide a scan-wise scene interpretation of the scene and the actual models for scene interpretation. We furthermore cover aspects of self-supervised representation learning that enables to reduce the amount of manually annotated data. The scan-wise scene interpretation is the second component of the spatio-temporal scene interpretation.

In the last part, Chapter 4, we finally describe our contributions regarding spatio-temporal scene interpretation that will cover semantic mapping and the sequence-wise scene interpretation. Both contributions unify the spatial and semantic scene interpretation that is needed to provide a self-driving car not only situational understanding but also temporal understanding of the surrounding scene.

In the end of the thesis, we conclude with an outlook what is missing to attain scene understanding and discuss aspects of reducing the need for accurately labeled data to learn the before discussed models that rely on supervised learning. While supervised learning enabled much of the progress in perception for self-driving cars, it cannot cover all situations that a self-driving car will encounter and there is a need for going beyond supervised learning. In particular the aspect of long-tail classes, i.e., object classes that only appear rarely, or even unknown classes, i.e., object classes that do not appear in the training set, is an active research area and handling such cases is still an unsolved problem for perception for autonomous vehicles.

In Figure 1.2, we show the dependencies between the respective parts, but also the relation between the publications of this thesis. As can be seen, the spatial relation between scans provided by the relative pose estimation [R] is the foundation of many different research directions covered in the thesis, i.e., localization [E], [L], [M] and map building [K]. Moreover, it is also the key building block to provide a dataset [O] for learning semantic models of the world. The dataset [C], [O] and its extension [I] are the basis for what follows in respect to semantic interpretation using single LiDAR scans [B], [N], [Q], but also sequences of LiDAR scans [A], [F], [P]. Building on the foundation of a large-scale dataset of sequential 3D LiDAR scans providing point-wise annotations, most of the investigated research direction could be explored and novel contribution in respect to semantic interpretation of 3D LiDAR data could be explored. We furthermore contributed to the progress of the field by providing multiple benchmarks, ranging from semantic segmentation, panoptic segmentation, semantic scene completion to moving object segmentation and 4D panoptic segmentation using 3D LiDAR sequences.

Chapter 2

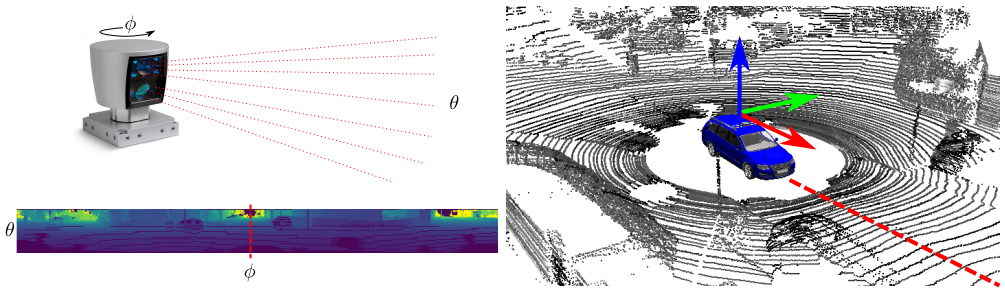
Localization and Mapping using 3D LiDAR Sequences

In this chapter, we will summarize our contributions on building maps and localizing in these maps using a sequence of 3D LiDAR scans. Using either SLAM or localization algorithms, an autonomous system determines it's pose in three-dimensional space over time. Given these pose estimates, it can then relate observations of the environments to each other and we will later use this information to aggregate LiDAR scans to gain spatial understanding of the environment. Consequently, reliable pose estimation is an essential stepping stone towards spatio-temporal scene understanding.

With the advent of fast 3D LiDAR sensors that can provide complete 360° scans of the environments, the research interest in 3D LiDAR-based SLAM increased drastically. In contrast to cameras, LiDAR sensors provide accurate range measurements of the environment. A 3D LiDAR sensor covers the environment with range measurements, which can be translated to point clouds representing the vicinity of the autonomous system. Such point clouds, shown in Figure 2.1 are the basis for the solutions to SLAM, localization, and mapping in this chapter.

A key challenge for using point clouds for SLAM, localization and mapping is to represent the (generated) map in a memory-efficient way such that new measurements can be associated with the known map to determine the current pose. Therefore, the map representation must be space efficient, but also show a high fidelity, i.e., sufficiently represent details of the environment, at the same time. An appropriate map representation can be easily queried to compare current observations with the map to determine the current pose.

There are many different approaches to fulfill these requirements and we will cover in the first chapter our contributions to efficiently perform SLAM, mapping, and localization using a sequences LiDAR scans. In Section 2.1, we will summarize our contributions regarding an efficient LiDAR-based SLAM system using so-called surfels that provides globally-consistent poses using a sequences of LiDAR scans without any external information. In Section 2.2, we will summarize our solution to adaptively select an appropriate robust kernel to handle outliers in the data association for scan registra-



(a) (b)

Figure 2.1: Generation of a point cloud. (a) A rotating 3D LiDAR scanner, such as the shown Velodyne HDL-64 E, has multiple laser emitter and detectors that measure distances by time-of-flight, i.e., the time it takes from emitting a laser beam until reflected photons of the laser beam are detected again. For each beam, the LiDAR records the range and intensity of the laser return in a range image, where it also records the turn angle ϕ of the LiDAR sensor. (b) Given the rotation angle ϕ and the emitter angle θ, it is possible to convert the range image representation into a 3D point cloud.

tion. In Section 2.3, we will summarize our approach for generating a high-fidelity map using triangle meshes. In Section 2.4, we will present our approach for compressing an aggregated point cloud, such that it can be efficiently transmitted, but also then reconstructed. In Section 2.6, we will summarize our contribution for localization using a deep neural network that utilizes the concept of overlap. Lastly, in Section 2.7, we will describe our approach for exploiting triangle meshes for localization using particle filters.

2.1 Efficient Surfel-based SLAM

Simultaneous Localization and Mapping (SLAM) is classical topic in robotics [10], [9], [8] and an essential building block of most autonomous systems, since it provides the capability to localize an autonomous system in an unknown environment. Given a sequences of observations, a SLAM approach builds a map of the environment that allows it to determine the location of the sensor in respect to this map. Thus, it needs to use the sensor information to determine the position of current observations in relation to prior observations. What makes SLAM a challenging problem is that the algorithm needs to determine a consistent map integrating past observations and it needs to localize in the map to determine the current position of the sensor at the same time.

A SLAM front-end is usually composed of two main parts: (i) odometry estimation that provides a relative pose estimate to prior observations using the map and (ii) loop closure handling that allows to correct drift that has been accumulated over time due to inaccuracies of the relative pose estimates over time. A loop closure, i.e., identifying revisited places, can be used to compute a globally consistent map and correct prior pose estimates and consequently the generated map using a SLAM backend, often based on optimization techniques.

With the advent of fast 3D LiDAR sensors that can produce millions of accurate

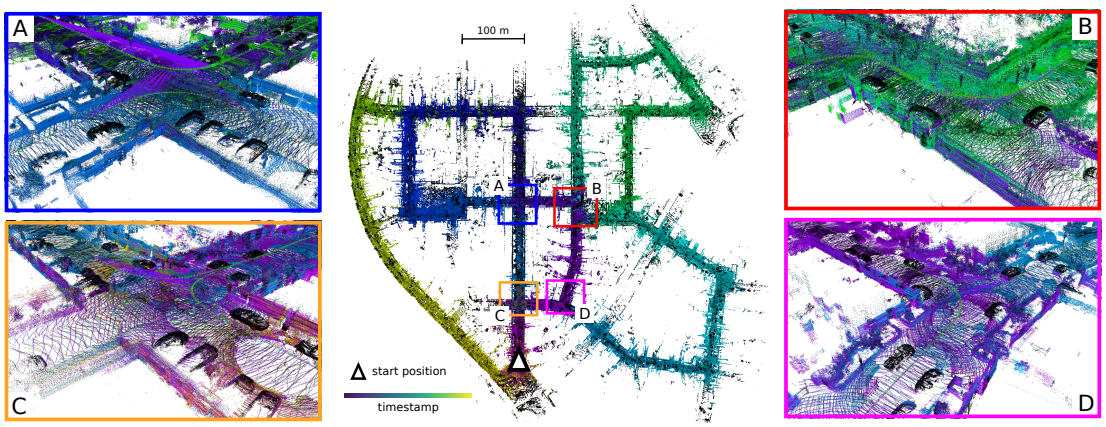


Figure 2.2: Aggregated point cloud of sequence 00 of the KITTI odometry benchmark. In the middle, the complete aggregated point cloud using the estimated poses is shown. The colored rectangles show the areas that are shown in more detail. Notable is the high consistency of the aggregated point clouds visualized in the detailed views at loop closure locations.

distance measurements, the robotics community developed many approaches for SLAM. In our approach [R], we develop an approach that can estimate globally consistent poses of a 3D LiDAR sensor that is commonly equipped on autonomous vehicles. The key contributions of our work can be summarized as follows:

- We present a real-time capable approach for SLAM that generated globally consistent pose estimates and globally consistent maps.
- We propose a projective approach for data association that exploits that we can efficiently render our map representation using surfels, which are small circular disks approximating the environment via small surface patches.
- We propose a loop closure detection approach that uses a map-based strategy to determine loop closures even in situation with little overlap between scans.

More concretely, our SLAM system is comprised of two parts: (1) odometry estimation and (2) loop closure detection and integration. For odometry, we propose an approach that uses the iterative closes point (ICP) algorithm [11] to determine the pose of a 3D LiDAR scan in relation to a surfel-based map representation [12]. The surfel-based map representation aggregates 3D LiDAR scans in a dense representation that can be used to efficiently render views of the scene that can be used to projectively associate 3D LiDAR measurements with the map.

For finding loop closures, we propose furthermore an approach that uses the map representation to determine via ICP if a place has be revisited. By a consistency check that pre-integrates LiDAR scans into a temporary map using the pose estimate in relation to the old map, we are able to identify loop closures in an online fashion. Using the loop closure constraints between current observations and prior observations aggregated in the surfel map, we can perform a pose graph optimization [13] to determine globally consistent poses. As surfels are always associated to a pose estimate that can

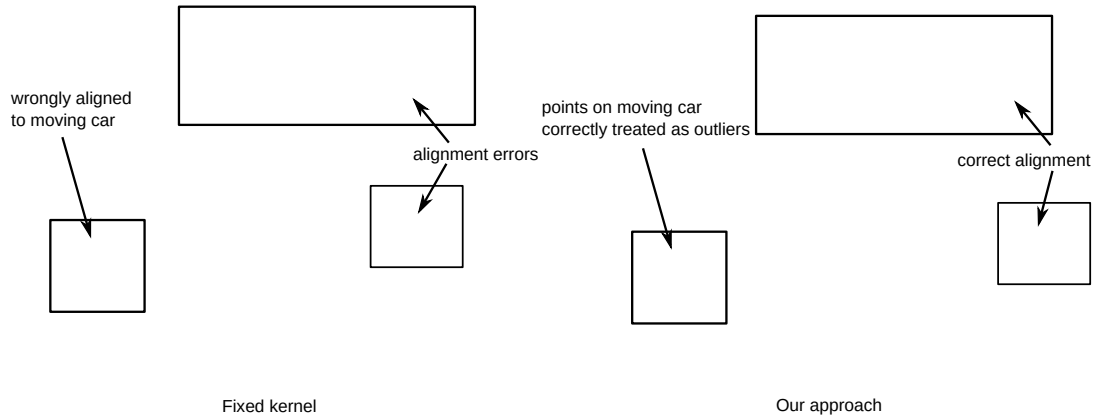


Figure 2.3: Comparison between fixed and adaptive kernels on a challenging situation on a highway, where only few of structural geometric features can be used to find correct correspondences. In this situation, the car overtaking leads to wrong associations. With out proposed solution, we can adaptively select the right kernel leading to more consistent scan matching results.

be updated, we are able to efficiently refine the map and determine globally consistent surfel maps.

Experiments on the KITTI odometry dataset [15], [14] showed that our proposed approach is able to estimate globally consistent poses at the frame rate of the employed 3D LiDAR sensor, i.e., above 10 Hz. Compared to other approaches for 3D LiDAR-based SLAM, it achieved superior results on the training set. Figure 2.2 shows an example scene from the KITTI dataset of the aggregated point clouds with multiple loop closures, where the aggregated point clouds shows the consistency of the scan alignment achieved by our proposed method.

2.2 Adaptive Robust Kernels

The described SLAM system in the previous section relies on finding correct correspondences between the map and the current observation to determine the pose of the 3D LiDAR sensor. However, determining correct correspondences is affected by errors due to dynamics in the scene and occlusions caused by objects that can lead to wrong pose estimates and subsequently lead to wrong maps generated by the SLAM approach.

To reduce the effect of wrong data associations, a common solution is to use robust kernels that reduce the influence of large residuals in the non-linear least squares solution. In our surfel-based mapping approach presented in Section 2.1, we used Huber weighting [16] that reduces the influence of large residuals by a linear weighting of large residuals. Other variants, such as Cauchy [17], Geman-McClure [18] or Welsch [19] kernels have even stronger effects on the weighting of large residuals. While such robust kernels can make the solution more stable and reduce the influence of severe outliers on the solution, it can also lead to less accurate solutions as correct data associations are also affected by the kernels. Thus, using an appropriate kernel is essential to provide accurate pose estimates.

For handling such situations adaptively by using the right kernel depending on the outlier distribution, we propose an approach [D] extending the work of Barron [20] to select the right kernel automatically. Our contributions are as follows:

- We extend the approach of Barron [20] to also consider additional kernels, which enables to use also kernels that have a stronger downweighting of large residuals.
- We show quantitatively and qualitatively in our experiments that frame-to-frame odometry in situations with dynamics can be effectively handled by our extended adaptive kernel selection leading to more robust pose estimation.

In our approach [D], we use an adaptive robust kernel $\rho(r; \alpha, c)$ proposed by Baron [20] that unifies different commonly used robust kernels:

$$\rho(r; \alpha, c) = \frac{|\alpha - 2|}{\alpha} \left(\left(\frac{(r/c)^2}{|\alpha - 2|} + 1 \right)^{\alpha/2} - 1 \right), \tag{2.1}$$

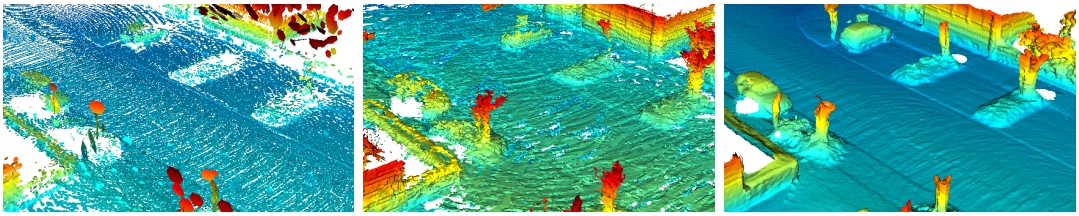
where α controls the shape of the kernel function and $c > 0$ determines the size of the quadric loss region around $r = 0$. While Baron proposed a method to compute the best α using an adaptive loss function based on $\rho(r; \alpha, c)$, it can only account for cases where $\alpha \geq 0$. We proposed a solution that also accounts for cases with $\alpha < 0$ and thus allowed us to incorporate kernels that ignores correspondences outside the region defined by c.

Our experiments on two tasks, odometry estimation for LiDAR-based SLAM and bundle adjustment for monocular images, convincingly shows that the proposed extension provides more accurate and reliable pose estimates then using a single robust kernel, but also that the extension provides us with the capability to handle gross outlier events caused by dynamics in the scene. Figure 2.3 shows some qualitative results for a challenging situation on a highway where only very few geometric features can be used for correct alignment. However, in this situation the fixed kernel solution cannot correctly filter the wrong correspondences on a moving car.

2.3 Mesh-Based Mapping

The map representation is a critical component of every SLAM system, since it aggregates prior observations to allow a SLAM algorithm to determine the relative pose in respect to the map. Thus, the map representation in the context of autonomous driving must be accurate, i.e., capturing enough details to provide good correspondences for pose estimation, but also lightweight at the same time to allow us to represent large outdoor scenes. Finding a good tradeoff between the ability to represent details of the environment, but also the memory consumption is an active research area in 3D LiDAR mapping.

In Section 2.1, we represented the map via surfels [7] that are small circular patches approximating the surfaces in the environments. While this representation is easy to maintain and modify, it is also quite memory inefficient and sometimes only a poor



(a) Surfels (b) TSDF (c) Mesh

Figure 2.4: Qualitative results of the different map representations. (a) surfels, (b) truncated signed distance function (TSDF), and (c) mesh-based map representation. The proposed mesh representation is much denser, smoother, and shows less artifacts that potentially affects the pose estimation or localization methods.

approximation of object surfaces. In particular, thin surfaces or cylindrical objects, such as poles, can only be accurately represented, when very small surfels are used. However, poles or tree trunks are in particular a good landmark in the environment [21] that can be effectively used for registration and therefore would be beneficial to be represented accurately.

Another map representation that gained popularity is a truncated signed distance function (TSDF) [22], [6], which is a voxel-based representation, where each voxel stores the signed distance to the nearest surface. Approaches, such as KinectFusion [22] for RGB-D mapping, showed that this can be used for mapping and efficiently updated leading to compelling results. However, representing the environment by a dense voxel grid does not scale well to larger environments and therefore more sparse storage of the signed distance field where investigated [24], [25], [23].

In our mapping approach [K], we represent the map by a triangle mesh that allows us to represent the surfaces far more efficiently in terms of memory consumption, but also more faithfully as a triangle mesh representation can represent arbitrary surface topologies. Our contributions can be summarized as follows:

- We present a mesh-based map representation based on Poisson surface reconstruction [26], [27] that produces geometrically accurate surfaces and is more memory efficient than other map representations at the same time.
- We show that a frame-to-mesh-based registration leads to a more accurate pose estimation compared to a point cloud-based map representation.

For our mapping approach [K], we use estimated poses to build an aggregated point cloud that is then reconstructed using Poisson surface reconstruction (PSR) [26], [27] that allows to determine a water-tight mesh from a given point cloud. For pose estimation, we use a buffer of N = 30 scans from which a triangle mesh is reconstructed using the PSR and data association is performed via ray casting to find point correspondences by the intersection of the rays with the triangle mesh. Using these correspondences of intersections $\mathbf{q}_i \in \mathbb{R}^3$ and the corresponding points $\mathbf{p}_i \in \mathbb{R}^3$, we perform point-to-plane ICP [28]. Using the local mesh reconstruction, we also produce a global reconstruction by incrementally updating the global triangle mesh. The globally reconstructed mesh

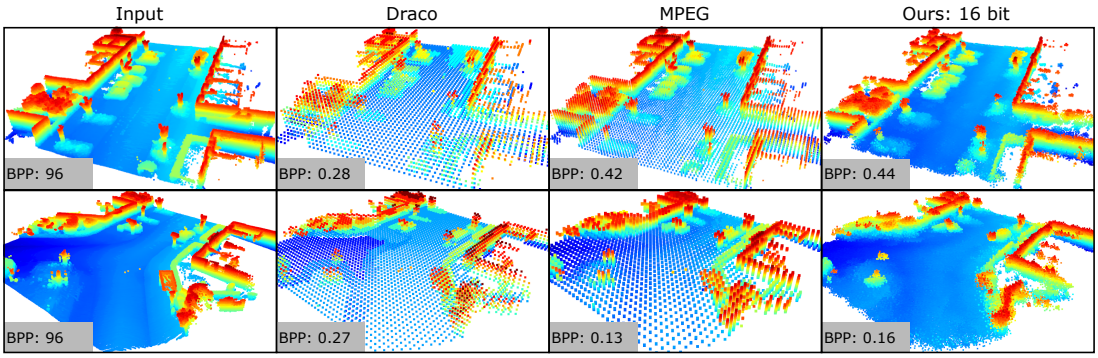


Figure 2.5: Qualitative results of the map compression approach in comparison to different baselines. The bits per point (BPP) is determined based on the size of the compressed point cloud and the number of the original points. We show the reconstructed point clouds from the compressed representation. Points are colored according to the height above ground. Clearly visible is a much denser reconstruction of our approach, which retains details of the original input point clouds.

can then be used for visualization purposes, but also as a representation for mesh-based localization as presented in our follow-up work [J], where we use the mesh to render range images used for localization.

Experiments on synthetic and real-world data shows that the provided reconstruction quality is more accurate and more memory efficient at the same time in comparison to the aforementioned map representations, such as surfels or a TSDF. Figure 2.4 illustrates the qualitative differences between different map representations including surfels and a truncated signed distance function [22]. As can be seen from the qualitative images, the level of detail and fidelity of the mesh-based representation provides more geometrically accurate and dense maps.

2.4 Map Compression

As a fleet of autonomous cars is navigating through a city environment, it will rely to some extend on external maps. Either topological maps that contain only the structure of the road network as used by navigation systems to perform global planning or so-called high definition maps, also called HD maps, that furthermore also contain additional more fine-grained information, such as location of traffic lights, lane markings, etc. that are used to perform local planning.

All map representation must be continuously updated, since changes such as a construction site or temporary changes such as changes in the available parking areas might appear without much time beforehand. Therefore, some information from a specific car of the fleet must be transferred to update the aforementioned map representations. Even with the promised capabilities of 5G networks, this information must be transferred in an efficient manner.

A kind of information that could be exploited is an aggregated point cloud gathered a specific location, which provides a detailed view of the encountered situation. Transferring such raw data requires a considerable amount of bandwidth. Also, storing such

data requires a huge amount of storage space that might prevent to record an extended period of time.

While traditional compression algorithms could reduce the amount of data that needs to be transferred or stored, we proposed a far more effective approach [G] that can exploit the structure of point clouds. In summary, we make the following contributions:

- We develop a novel deep learning-based approach for compression 3D LiDAR point cloud data in a lossy fashion for large-scale outdoor environments.
- We furthermore propose a novel deconvolution for point clouds with feature propagation and integration that can be used to generate upsampled point clouds from a given set of point with features.

In our work on map compression [G], we propose an auto-encoder that produces a latent representation of the point cloud consisting of a reduced set of three-dimensional points with features that can be used to reconstruct the original point cloud. Using the encoder, the network can now encode the latent representation, which is considerably smaller than the original point clouds. Only this reduced set of points with features is needed to be transferred or stored. The corresponding decoder can then produce from the latent representation a reconstructed point cloud that faithfully recovers the original point cloud map.

We evaluated our approach on aggregated point clouds of the KITTI Vision Benchmark [15], [14] and showed superior performance in terms of compression performance and reconstruction quality. Figure 2.5 shows qualitatively the uncompressed point clouds for various approaches.

As the compressed representation captures the spatial context of the point cloud, which makes it possible to reconstruct the original point cloud, we employed the compressed representation also in context of place recognition [29] and registration [30]. Interestingly, the learned representation is also quite generalizable, which made it possible to use the learned representation on different dataset recorded in different locations.

2.5 Overlap-Based Place Recognition

As described in Section 2.1, SLAM systems rely on finding loops, i.e., determining when an autonomous system revisits a previously recorded place. Such loop closures can be used to correct accumulated drift leading to globally consistent maps. Therefore, finding loops is a crucial aspect of a SLAM system and finding wrong loop closures can significantly degrade the performance of a SLAM system.

Finding loop closures can be achieved by place recognition methods that determine if a given observation matches to a previously recorded observation of the same place. LiDAR-based place recognition is a challenging problem as it has to account for viewpoint changes, but also structural changes in the environment, such as parked cars or the change of the season that can lead to different appearances of the same scene.

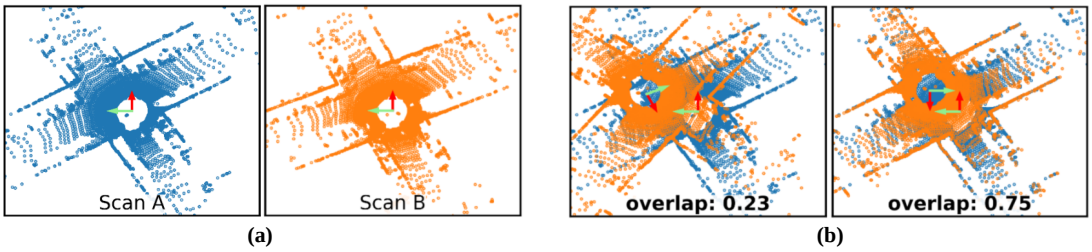


Figure 2.6: Overlap between (a) two point clouds that have been recorded at an intersection from two different view points. Depending on the pose between the two scans, the overlap between the scans changes and better alignment corresponds to better alignment. Our approach learns to predicts the overlap between two input point clouds without a relative pose between the scans.

Most approaches for LiDAR-based place recognition rely on extracting a descriptor [31], [33], [32], [34] that can be used for matching of locations. However, recently also deep learning-based approaches attracted considerable interest in the robotics community.

In our work on LiDAR-based place recognition [E], [L], we developed an approach which determines the similarity of two different point clouds by using the concept of overlap. The overlap between point clouds is defined as the amount of points that have a correspondence between different point clouds in the global coordinate frame, as shown in Figure 2.6. Thus, higher overlap corresponds to larger similarity between scenes. In summary, we make the following contributions:

- We propose an approach for LiDAR-based place recognition that uses a convolutional neural network to predict the overlap and the relative yaw angle between two point clouds.
- We combine odometry estimates and the overlap estimation to perform loop closure detection yielding more globally consistent maps in a SLAM approach.
- We show superior performance in terms of loop closure detection compared to other geometric and deep learning-based approaches.

In our work [E], [L], we propose a neural network, the so-called OverlapNet, that can learn to predict overlap between point clouds without the relative pose between these point clouds. Therefore, we can identify point clouds with high overlap corresponding to loop closures. To this end, we propose a siamese architecture with two branches that compute feature representations of each point cloud based on range images. The features are then used together with a novel aggregation module to compute the overlap and the relative yaw angle between the point clouds. To perform loop closure detection, we exploit that we have an odometry estimate of the global pose that can be used together with the uncertainty represented as a covariance to restrict the computational load to the relevant parts of the environment.

We evaluated our method for overlap prediction on the KITTI Vision Benchmark [15], [14] and showed superior performance to other LiDAR place recognition approaches. In comparison to the approach proposed in Section 2.1, we saw an increase in correct loop closures and therefore more reliable pose correction results.

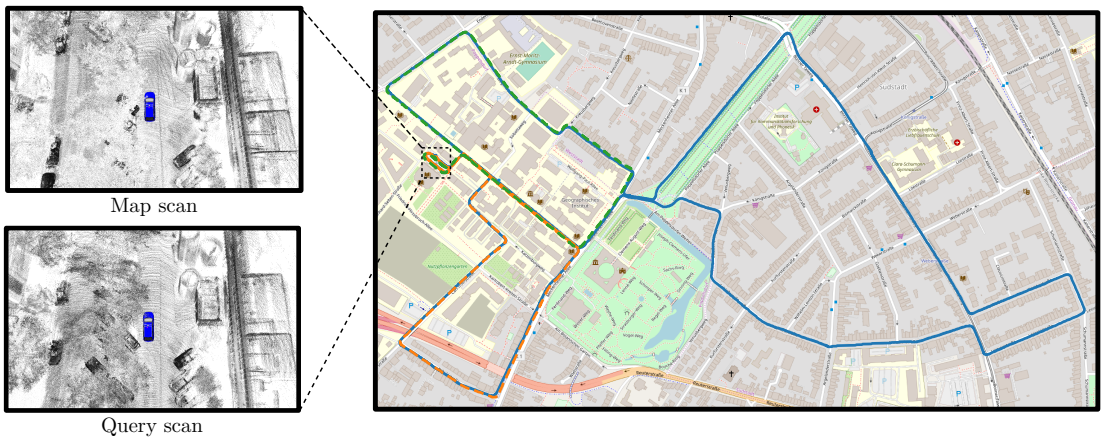


Figure 2.7: A localization approach needs to handle situations such as the shown difference in the appearance between the scans recorded in the mapping phase and the scan observed at a different point in time. As shown here, the scans have been recorded in different seasons. Furthermore, moveable objects, such as the parked cars, appear at different locations and therefore identifying the relevant parts of the environment is crucial for successful localization.

2.6 Overlap-Based Localization

While SLAM aims a localizing and building a map at the same time, localization aims at localizing in a *given* map. Separating mapping and localization has the advantage that the mapping can use more accurate, but also more time-consuming methods to provide a well estimated map representation, but also enables to manually correct errors by providing manual loop closure annotations.

Localization is closely related to place recognition, since a global localization algorithm needs also to identify the previously visited places based on the current observation. However, most localization algorithms exploit the temporal coherence between subsequent observations and account for the fact that we have a sequence of observations.

Probabilistic methods based on Bayes Filters are commonly employed for localization [8]. In particular, the Monte-Carlo localization (MCL) approach by Dellaert et al. [35] using a particle filter, where each particle represents a state estimate of the current location in respect to a given map, is an effective framework for localization. A crucial component of a MCL approach is the observation model that is used to weight the particles to ensure tha the particles are converging to the true location.

In our work [E], [M], we used the previously introduced OverlapNet (see Section 2.5) to define an effective observation model for the MCL to weight the particles according to the overlap between current observations and pre-computed virtual scan locations. In summary, we make the following contributions:

- We propose a MCL-based approach for localization that uses an observation model based on OverlapNet [L] for determining the overlap between the current observation and locations in the given map.
- Our approach can cope with situations where the appearance changes between

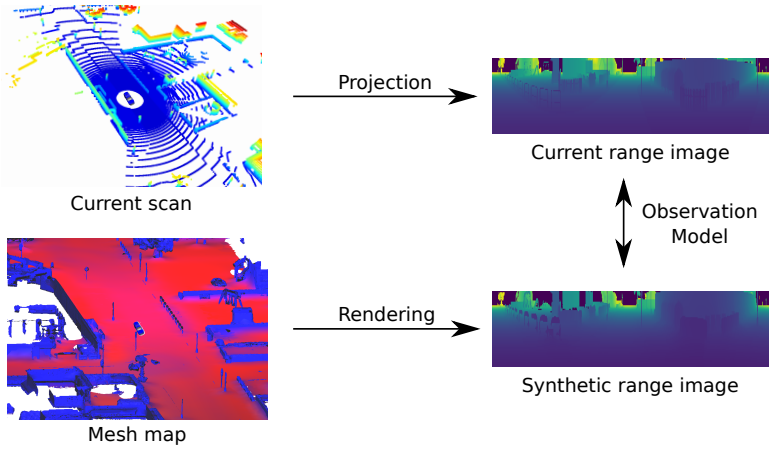


Figure 2.8: Given a map represented as a mesh, we can render at arbitrary poses to generate a view that can be compared by the observation model with the current range view of the observation for a particle filter-based localization.

the mapped and the observed environment caused by movable objects, but also seasonal changes.

In our work [M], we exploit that OverlapNet [L] uses two feature representations of two scans to estimate the overlap. Therefore, we can pre-compute the feature representations of virtual scans of the map to reduce the amount of memory considerably compared to storing the raw point clouds (features: 0.1 Gb/km vs. raw scans: 1.7 GB/km). The observation model for the weighting of the particles, where larger weights corresponds to more likely positions in the given map, has two components. First, we use directly the estimated overlap as an term in the observation model $P(\mathbf{z}_t|\mathbf{x}_t)$, where z_t corresponds to the current observation and $\mathbf{x}_t$ to the state. Second, we also exploit that OverlapNet provides an estimate of the yaw angle between the given point clouds – therefore, we can determine the difference between the estimated yaw angle and the yaw angle of an particle filter to derive a second term in the observation model. Both together, the overlap estimate and the difference in yaw angles is then the probability of making the observation z_t at the given state $\mathbf{x}_t$.

We compared our overlap-based observation model with commonly employed observation models on the KITTI Vision Benchmark [15], [14] and in-house sequences recorded with a different sensor setup. The experimental results show that the overlap-based observation model leads to superior accuracy in terms of pose accuracy and faster convergence in global localization, where the initial pose is not available. At the same time, by exploiting the spatial distribution of the particles, we can ensure that only relative few evaluations of the OverlapNet must be performed in practice, which leads to a highly efficient re-weighting of particles.

2.7 Mesh-Based Localization

In Section 2.3, we summarized our approach for mesh-based mapping, where we could generate a very detailed and dense map representation using a triangle mesh. As

pointed out, such meshes can faithfully represent details and are very memory efficient at the same time. Furthermore, triangle meshes are a backbone of object representation in computer graphics and therefore modern graphics processing units (GPUs) are optimized to render triangle meshes very efficiently.

Thus, using such representation for localization has multiple advantages, such as reduced memory consumption for storing the map in memory, but also the prospect to exploit the rendering capabilities of modern graphics hardware to render a triangle mesh. In our approach [J], we leverage the aforementioned properties to localize an autonomous system equipped with a rotating 3D LiDAR sensor in a mesh map by using a MCL-based approach as shown in Figure 2.8. In summary, we make the following contributions:

- We propose a MCL-based approach for localization that uses rendering of a triangle mesh to determine the compatibility of an observation with the map.
- We formulate an observation model directly operating on the dense observations and rendered map views as range images.
- An efficient batch rendering using OpenGL enables real-time capable localization for pose tracking above the frame rate of the employed LiDAR sensor.

In our approach [J], we first generate from an aggregated point cloud of an environment a mesh using the approach presented in Section 2.3 resulting a triangle mesh of the whole environment. Given that triangle mesh, we can then use an OpenGL shader to render a range image at arbitrary positions that can be used in our observation model. More specifically, we propose a novel observation model that determines the difference between the current observation and a rendered view at the given particle location by using range images. A range image of a point cloud is a spherical projection and we implemented a rendering method that can generate such range images from a given mesh map. For global localization, we use a rather large number of particles, i.e., 1000 particles, leading to iterations times above the sensor frame rate, but after convergence the system is able to perform the pose tracking with a reduced number of particles, i.e., 100 particles, enabling real-time capable localization.

Experiments on a variety of datasets, the KITTI Vision Benchmark [15], [14], the MulRan [36], our in-house dataset, and the Apollo (Columbia Park) [37] show efficient and accurate localizations capabilities of our proposed approach that often surpasses the performance of OverlapNet, but at a relatively low runtime reaching 21.8 Hz after convergence.

Chapter 3

Semantic Interpretation of 3D LiDAR Scans

In this chapter, we will summarize our contributions in respect to the semantic interpretation of a single 3D LiDAR scans, which is needed to make sense of the surrounding environment by identifying drivable and non-drivable space, but also identify the other traffic participants. Scene interpretation is the basis for many downstream tasks, such as planning and navigation. We will focus on approaches for semantic segmentation, but also panoptic segmentation, since these perception tasks provide use with a fine-grained scene interpretation of the environment that can be used for other downstream tasks, such as planning, collision avoidance, but also tracking. Figure 3.1 shows the kind of point-wise labels generated by the different semantic interpretation techniques.

In semantic segmentation, we aim at determining a semantic class for each point of a 3D LiDAR scan, which enables us to not only reason about objects, e.g., pedestrians, cars, bicyclists, but also surfaces, such as the road surface, parking areas, sidewalks, and buildings. Knowing the location of other traffic participants, but also the drivable and non-drivable surface is essential for an autonomous car to perform safe and reliable navigation.

In panoptic segmentation, we extend the notion of point-wise classification to also provide a distinction between individual object instances, where we distinguish between different instances of the same object class. This additional information is essential to reason about the behavior of other traffic participants and can be exploited for tasks, such as tracking or behavior estimation. Panoptic segmentation combines semantic and instance segmentation into a single task.

Before we discuss solutions for semantic and panoptic segmentation, we will first present our contributions regarding the development of a dataset for learning models that can perform these tasks. In Section 3.1, we will summarize our contributions towards a large-scale dataset with point-wise annotations, which not only provides dense semantic classes for points of a 3D LiDAR scan, but also temporally consistent instance annotations. In Section 3.2, we will then summarize our contributions towards an efficient semantic segmentation approach that learns from the data a model to predict

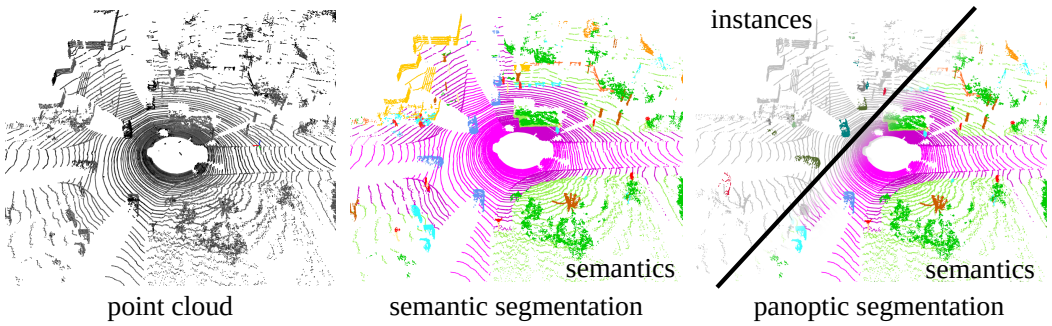


Figure 3.1: Difference between semantic segmentation and panoptic segmentation in terms of labels. While semantic segmentation provides point-wise labels, panoptic segmentation additionally provides instance labels for so-called thing classes, such as cars. Colors in the case of semantics corresponds to different classes, such as road, sidewalk, vegetation, terrain, car, building. For instances, each color correspond to a distinct instances.

for each point of an unseen 3D LiDAR scan a semantic class. We then will summarize in Section 3.3 our contributions towards an approach for panoptic segmentation that extends our work introduced in the previous section to also provide instance predictions. In the last chapter, Section 3.4, we discuss an approach that has the potential to reduce the amount of labeled data considerably by using self-supervised representation learning.

3.1 A Large-scale Dataset for Semantic Interpretation

Recent progress in perception using images is driven by advances in deep learning enabling end-to-end trainable perception systems without the need to hand-craft features [4], [5]. Training deep neural networks with millions of parameters was mainly enabled by two essential developments: (1) the possibility to repurpose graphic processing units (GPUs) from producing pixels on a computer screen to compute matrix products in a highly parallel fashion, and (2) the availability of large-scale labeled datasets, such as ImageNet [38], that enabled training networks without over-fitting to the training data.

While there has been large datasets for fine-grained image interpretation available, such as Microsoft's Common Objects in Context (COCO) [39] using images crawled from the internet or Cityscapes [40] using driving scenes recorded in everyday traffic, there was no comparable dataset for LiDAR point clouds available with a similar scale or quality of annotations. Especially, temporally consistent annotations of whole sequences recorded with an automotive 3D LiDAR sensor was not readily available. To fill the gap and enable much of the work that follows in the upcoming chapters, we developed a way to annotate a large dataset with temporally consistent point-wise annotations.

In our work [O], we presented our dataset and associated benchmark for studying semantic segmentation and semantic scene completion. We later extended the dataset with instance annotations [I], such that also investigation of panoptic segmentation is

possible. A summary of the whole dataset is given in our article [C]. In summary, we make the following contributions:

- We provide a dataset based on the KITTI odometry benchmark [14] that is annotated with dense point-wise class labels.
- We furthermore establish several benchmark tasks for semantic segmentation and semantic scene completion that can be used to evaluate the performance of approaches. In particular, we hide the test set labels and use a server-based evaluations script to compare the predictions with the hidden ground truth data.
- We later extended the point-wise semantic annotations with instance annotations of traffic participants that enables to also study the problem of panoptic segmentation. An associated benchmark tasks enables reproducible and unbiased results on a hidden test set.

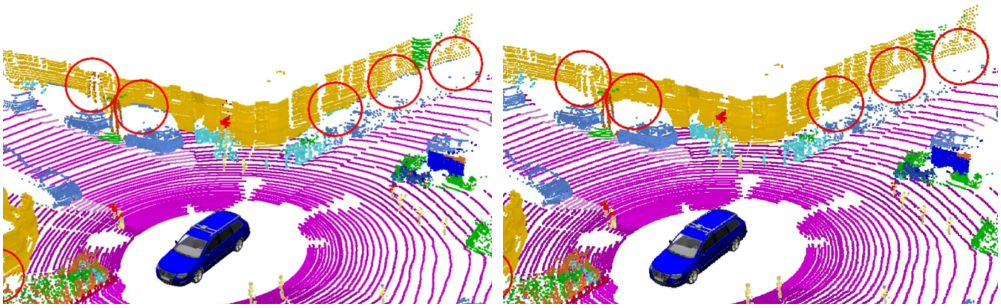
We started from the well-established odometry dataset from the KITTI Vision Benchmark [15], [14], since it was the largest dataset available that provided sequences of LiDAR point clouds for a different driving scenarios at that point in time. The whole KITTI odometry benchmark comprises of 22 sequences with over 43,000 scans. For efficiently labeling such an amount of data, we exploited that we can get accurate poses of the sequences using our method for LiDAR-based SLAM presented in Section 2.1, which are globally consistent and loop closed. This enabled us to develop an annotation tool that uses aggregated point clouds to speed up the process of annotation, since we do not have to label each scan individually. Thus, labeling of a a part of the aggregated scans provides labels for multiple individual scans. Besides the time saved by using the aggregated scans, we also achieve a much more consistent label assignment of the same spatial locations seen by many viewpoints.

Overall, a team of labelers provided initial labels resulting in over 1 400 hours of work, which was then validated and refined with additional 300 hours of labeling time. We separated the aggregated point cloud into tiles of 100 m×100 m, where a labeler needs between 1.5–4.5 hours depending on the complexity of the scene. Additionally, we invested roughly 70 hours to additionally annotate individual instances resulting in the first panoptic segmentation benchmark for point clouds [I].

As the provided annotations, also the benchmark tasks were incrementally extended and we introduced novel tasks over time. We extended the benchmark with so-called 4D panoptic segmentation [H], see also Section 4.3, where one not only needs to perform scan-wise panoptic segmentation, but also provide temporally consistent instances. We, furthermore, introduced a benchmark task that concentrates on distinguishing moving and non-moving objects [F], see also Section 4.1.

Since the release of the dataset and associated benchmarks, we saw great adoption of the dataset by the robotics and computer vision community. The associated benchmark for semantic segmentation hosted on CodaLab [1], has over 600 registered accounts, which overall produced over 1 800 submissions to the benchmark servers. However, the

[1] see `https://competitions.codalab.org/competitions/20331`



(a) (b)

Figure 3.2: Results before (a) and after (b) the k-NN-based consensus filtering. Colors in the case of semantics corresponds to different classes, such as road, sidewalk, vegetation, terrain, car, building.

dataset also open the door for investigation of different tasks, where semantics can be exploited or are the foundation for higher-level tasks, such as semantic mapping [P], simulation [41] or domain transfer [42].

The dataset also drove much of our research on approaches for semantic interpretation, presented in Section 3.2, Section 3.3, and Section 3.4, but also enabled contributions for spatio-temporal scene interpretation, presented in Section 4.1, Section 4.2 and Section 4.3.

3.2 3D Semantic Segmentation

Fine-grained semantic scene interpretation is an integral building block of autonomous vehicles, since it allows to distinguish drivable and non-drivable space, such as road and sidewalks, but also allows to gain information about the traffic participants in the vicinity of the vehicle. On the basis of the large-scale dataset presented in the last section Section 3.1, it was now possible to also investigate large capacity models that proved to be successful in image-based semantic segmentation – deep neural networks. As the size of these models usually requires a large training set, work in this domain [43] was restricted to repurposing detection datasets, such as KITTI vision benchmark [15], [14], which only provided a few classes. The SemanticKITTI dataset [O] provides a much more diverse set of 28 classes, which can be used to learn models for semantic scene interpretation of point clouds produced by an automotive LiDAR sensor.

In our follow-up work [Q], we extended an approach which proved to be quite competitive in the baseline experiments, which we conducted for our dataset [O]. In summary, we make the following contributions:

- We propose an efficient approach for 3D LiDAR-based semantic segmentation that provides point-wise classes for a complete 3D LiDAR scan using range images.
- As the back-projection of range image labels leads to artifacts, we propose an efficient filtering method based on the spatial neighborhood of a point to refine the labels.

- Experiments on our dataset, SemanticKITTI, shows more consistent prediction results that can be produced in real-time, i.e., at 12 Hz on a desktop GPU.

In our approach [Q], we used a spherical projection of the point cloud into a range image that allowed us to apply 2D convolutions and leveraging progress in the image domain for the task of semantic segmentation. In comparison to other approaches [48], [47], [46], [44], [45] that directly operate on the point clouds, the projective approach could exploit a larger neural network, which led to improved performance. However, the backprojection of the inferred labels leads to artifacts, such as wrong labels in the background as shown in Figure 3.2. To resolve this issue, we proposed an efficient k-nearest neighbor (k-NN)-based consensus of labels that accounts for the 3D spatial relation between points.

Experiments on SemanticKITTI [C], [O] showed that the proposed post-processing leads to superior performance in terms of semantic segmentation quality, which provides real-time capable semantic segmentation, i.e., segmentation results with a higher frequency than the sensor turn rate. In Section 4.2, we will exploit this real-time capable semantic segmentation approach to extend our SLAM pipeline introduced in Section 2.1 to also account for the semantic classes while mapping leading to improved performance in scenes with dynamics.

3.3 3D Panoptic Segmentation

While semantic segmentation already provides valuable information in terms of the drivable and non-drivable space, it does not provide instance information, i.e., it is not possible to distinguish between different objects of the same class. To extract individual objects, such as location of cars, panoptic segmentation [49] combines commonly separately tackled semantic and instance segmentation. In this task, one distinguishes between stuff and thing classes. While stuff classes do not show a distinct object boundary, thing classes correspond to objects which can be clearly separated. In autonomous driving, thing classes are mainly other traffic participants, such as other vehicles, pedestrians, or bicyclists, that are mainly relevant for autonomous driving.

The novel task of panoptic segmentation was quickly adopted by image-based dataset such as Cityscapes [40] and Mapillary Vistas [50] after Kirillov et al. [49] proposed the task and a corresponding metric. In line with this developments, we extended SemanticKITTI with instance annotations [I] and provided the first 3D LiDAR dataset for studying panoptic segmentation using automotive LiDAR sensors.

In our work [N], we proposed the first approach to perform panoptic segmentation using a single network which reduces the processing time significantly as only a single forward pass through a single network is needed to compute labels and instance embedding vectors, which then can be used to produce instance annotations. In summary, we make the following contributions:

- We propose a CNN-based approach for panoptic segmentation of 3D LiDAR point

clouds that can be used in real-time settings, where a predictions must be produced below the frame rate of the sensor.

- Along with the instance embedding vectors that are used to cluster instances, we also learn to predict point-wise uncertainty radii that can be used to produce more coherent instance segments.

Based on our semantic segmentation approach, see Section 3.2, we proposed in our follow-up work [N] the first real-time capable panoptic segmentation approach for automotive 3D LiDAR data. To this end, we extended the network with a decoder that also provides embedding vectors that can be used in a post-processing step to cluster instances. As center predictions of instances and corresponding points of the instance can be far away, we furthermore predict an uncertainty-based radius to allow more reliable clustering. Our approach follows therefore the commonly employed bottom-up processing [51], where first point-wise embedding vectors are predicted and then finally via a clustering algorithm combined into instance clusters. The embedding vectors are learned such that points belonging to the same instance get vectors that are close together, while far away from other embedding vectors of different instances at the same.

Experiments on SemanticKITTI [C], [O] showed superior performance in terms of runtime in comparison to baseline approaches that combine semantic segmentation and object detection to extract instances. Furthermore, our approach allows to determine semantics and instances with a single forward pass, while the other approaches need to employ two different networks to attain panoptic segmentation.

3.4 Beyond Supervised Learning

Deep learning approaches using supervised learning achieved remarkable performance for semantic and panoptic segmentation. Researchers improved the performance in semantic segmentation considerably since we released our dataset. Modern approaches [52] reach over 70% in terms of mean intersection-over-union (mIoU), which is a huge improvement from the 49% mIoU of our best baseline using a projective approach, which was the basis for our contribution presented in Section 3.2.

While there has been considerable progress in respect to the methodology, it is also clear that it is not viable to always first invest a huge amount of annotation effort to achieve such decent performance by labeling thousands of LiDAR point clouds. In particular, models trained on a particular 3D LiDAR sensor configuration usually show weak performance on data recorded with a different sensor, i.e., sensors with a different number of beams or just a different mounting position [42]. Therefore, so-called unsupervised domain adaption approaches [54], [53], [42] were investigated to adapt an already trained model to a new sensor configuration using unlabeled data from the target sensor configuration. However, domain adaptation approaches only enable to adapt an existing trained model and these models still need to have a large labeled source dataset to attain a decent segmentation performance.

A different research direction uses pre-training to reduce the amount of labeled data. In particular, deep neural networks pre-trained on the ImageNet dataset [38] show faster convergence and better performance [55], when trained on different target datasets [56]. In this setting, we refer to this approach as supervised pre-training as a labeled dataset is still needed. Supervised training of a pre-trained network to the not necessarily same target task using a different set of labeled data is then called fine-tuning.

To alleviate the reliance on labeled data for pre-training, self-supervised representation learning using contrastive learning [57], [59], [58] showed great promise in the domain of image-based classification. By using the idea of training a network to produce first good representation by data augmentation, the amount of labeled data for fine-tuning is often drastically reduced to reach similar or even better performance as a fully supervised approach. Recently, these approaches showed also great performance with Vision Transformers [60] allowing to pre-train Transformer-based [61] architectures without the need for large labeled images for supervised pre-training.

Building upon this idea of self-supervised representation learning, we proposed an approach [B] to produce well pre-trained representations that can be used to reduce the need for labeled data. In our work, we provide the following contributions:

- We propose an approach for self-supervised representation learning using a momentum encoder [58] that uses segments leading to pre-trained networks that show superior performance for semantic segmentation with only few labeled points.
- We show that our self-supervised approach can be applied on different dataset and show superior performance compared to supervised pre-training.

We proposed an approach that uses a class-agnostic segmentation that can be used to learn good feature representation of the segments using momentum contrast [58]. Important is that we not only pre-train an encoder, but can also pre-train the decoder for semantic segmentation. Both together leads to pre-trained networks that can be fine-tuned with only very few labels that can reach the same performance as a network trained with all labels available. Thus, our approach leads to less labeling effort that has to be invested and can be used to learn strong semantic segmentation approaches.

Experiments on SemanticKITTI [C], [O] shows that the proposed pre-training strategy enables to attain similar performance as training with all data with just 50% of the data and superior performance with all the data. Furthermore, experiments with a pre-trained model on SemanticKITTI shows superior performance on a different dataset [62]. These experiments validates that self-supervised representation learning could enable to reduce the need for labeled data considerably.

Chapter 4

Spatio-temporal Scene Interpretation of 3D LiDAR Sequences

In Chapter 2, we summarized our contributions towards a more accurate spatial understanding using a sequence of LiDAR scans. These solution allow us to determine the relative poses between consecutive scans, which enables the capability to spatially correlate observations made over time.

The last chapter, Chapter 3, provided solutions for scan-wise semantic interpretation of 3D LiDAR scans, which can be used to determine the location of other traffic participants and extent of drivable space needed for safe navigation. However, we ignore the temporal aspect of the observations provided by the sensor that continuously produces 3D LiDAR scans of the environment.

An autonomous vehicle is operating in a highly dynamic environment with many moving actors that must be reliably identified to anticipate future locations of moving objects. In this chapter, we provide an overview of our contributions towards unifying the spatial aspect with the semantic interpretation by presenting our work towards spatial-temporal understanding of a scene. These approaches aim at providing a temporally consistent interpretation of the scene, where objects are identified consistently over time allowing an understanding of the dynamic and static parts of the environment. Such understanding facilitates prediction of future states of the environment.

In particular, we investigated the usage of a sequence of LiDAR scans that are produced by a rotating automotive 3D LiDAR sensor. As the measurements are three-dimensional points, these observations can be aggregated in to global observation of the environment, which is usually not possible by just using images due to the unknown scale and distance of objects projected into the 2D image. Thus, knowing where the LiDAR sensor is located in the environment and having poses for the whole sensor trajectory available is crucial for getting a coherent and consistent representation of the environment.

In context of self-driving cars, the understanding of moving objects is essential to

account for the trajectories of other vehicles in the environment. However, determining whether a vehicle moves is only possible to a certain extent using a single observation from a LiDAR scan, since LiDARs measure only distances in contrast to RADARs which can also determine the velocity of objects. Therefore, we need a sequence of LiDAR scans to determine the motion state of objects and this will be covered in Section 4.1.

As shown in Section 2, maps allow to localize a robot in environment and these maps must be geometrically detailed, but also lightweight at the same time to represent a large environment. While such geometric association between scans is already providing good pose estimates, it usually needs several heuristics and a carefully designed map aggregation mechanism to ignore non-static objects that can lead to wrong associations. Therefore, considering also the semantics of a scan can provide an mapping algorithm with additional information to better associate multiple observations. In Section 4.2, we summarize our contributions towards semantic SLAM, where we exploit semantic information provided by the approach described in Section 3.2 to determine more reliable associations between LiDAR scans.

In the last section, Section 4.3, we summarize our contributions towards spatial-temporal semantic interpretation, where we not only aim at distinguishing moving and non-moving objects, but also at providing a complete semantic interpretation of a sequence of 3D LiDAR scans. Similar to Section 3.3, we also want to provide a panoptic segmentation, but additionally want to determine temporally-consistent instances for traffic participants.

4.1 Moving Object Segmentation

Determining which objects in the vicinity of an autonomous vehicle are moving is essential to enable safe and reliable navigation. Moving objects must be considered in the trajectory planning, but also future positions of moving objects must be anticipated. Therefore, determining which points of a LiDAR scan are dynamic and which points correspond to static objects in the environment is a crucial aspect in LiDAR-based perception.

In contrast to RADARs, which can exploit the Doppler effect to determine besides the location of an objects also its velocity, LiDAR sensors can only determine the distance with a single measurement. However, RADARs only provide a very coarse and noisy representation of the environment, which makes it necessary to employ robust tracking or classification algorithms to extract sensible measurements. Therefore, using the complementary information of a high-resolution and accurate 3D LiDAR sensor is a promising research direction to get a more detailed picture about the moving and static parts of the environment.

For using multiple LiDAR scans to determine which objects moved in the environment, we first need to determine how the sensor moved while observing the scene. This can be achieved by the methods described in Chapter 2, which are also the basis for the moving object segmentation described in our work [F]. In this approach, we aim at

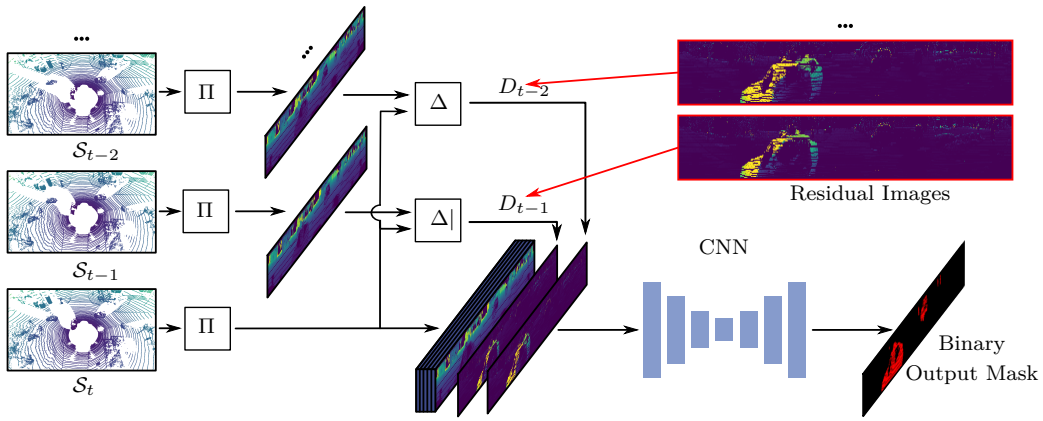


Figure 4.1: Overview of the proposed approach for moving object segmentation. We use the range projection-based representation of LiDAR scans to achieve online moving object segmentation. Given the current scan $\mathcal{S}_t$, we generate residual images from previous scans $\{\mathcal{S}_{t-i}\}_{i=1}^{N-1}$ to capture the sequential information. This is by transforming them to the current viewpoint estimated by a SLAM / odometry approach, projecting them to the range image with a mapping Π and subtracting them from the current scan's range image. The residual images D_j are then concatenated with the current range image R_t and used as input to a fully convolutional neural network (CNN). Trained with the binary labels, the proposed method can separate moving and static objects.

providing a label {moving, static} for each point of a LiDAR scan to determine if this LiDAR point corresponds to a moving or a static object.

In particular, our article [F] makes the following contributions:

- We present a real-time capable approach that is able to provide for each point of a LiDAR scan if it is moving or not.
- We propose a novel benchmark task for moving object segmentation that builds on SemanticKITTI [O].
- We demonstrate superior performance of the proposed method in comparison to several baselines and show that such binary semantic segmentation can be effectively used to attain more accurate SLAM performance and generate clean maps of an environment only containing the static environment of the scene.

In a nutshell, using the locally-consistent poses estimated by an odometry algorithm [R], we aggregate a sequence of LiDAR scans $\mathcal{S}_{t-N-1}, \mathcal{S}_{t-N-2}, \ldots, \mathcal{S}_{t-1}, \mathcal{S}_t$ represented as range images to build a spatio-temporal input representation. To this end, we transform all point clouds $\mathcal{S}_j$ into the local coordinate system of the current scan $\mathcal{S}_t$ using the pose estimates. Each point cloud is then projected to a range image R_j and a residual image D_j as pixel-wise difference between the current range image R_t and the previous range image R_j is generated. Based on the current range image R_t and the residual images $D_{t-N-1}, \ldots, D_{t-1}$, we then train a convolutional neural network (CNN) to classify each pixel as moving or static. Figure 4.1 provides an overview of our proposed method.

Experiments on the moving object segmentation benchmark shows that the proposed approach is able to more reliably predict, which points are moving than several

baseline approaches. In the paper, we also provide two direct applications of the prediction results: filtering of moving objects for SLAM and moving object removal for mapping using the approach presented in Section 2.7. For the application in the context of SLAM, we show that the pose error is considerably reduced (1.39 vs. 0.99 translational error on the KITTI odometry testset) as correspondences on non-static parts of the observations can be removed. In the mapping scenario, we show qualitative results showing that the moving objects can be reliably removed leading to cleaner maps without artifacts caused by moving objects.

4.2 Semantic SLAM using 3D LiDAR Sequences

As motivated earlier, reliable and accurate pose estimation is a key capability needed in the context of autonomous driving. As discussed in Chapter 2, there are generally two different paradigms of determining the location of a autonomous vehicle in respect to a map: SLAM, where the map is build on-the-fly and localization, where the map is already given. In both cases, semantics can be used to simplify the task. For SLAM, the semantic classes can be used to remove invalid correspondences, where the semantic classes disagree. It furthermore can be used to consider that potentially moving objects, such as cars, cannot be used for reliable registration. In case of global localization, the semantics can be used to improve place recognition [64], [63].

In our work on semantic SLAM [P], we leveraged semantics to improve the selection of correspondences needed to estimate the pose via iterative closest point (ICP) [11]. The key contributions presented in this paper are:

- We propose a surfel-based SLAM approach that uses semantics to improve the correspondences used for pose estimation leading to more accurate pose estimation results.
- We are able to build a semantic map representation of the static environment without artifacts caused by moving objects.
- We show superior performance of the approach using semantics compared to an approach relying only on geometric constraints for determining correspondences.

Our approach for semantic SLAM builds on the SLAM system presented in Section 2.1 and extends it using semantics. To this end, we used the semantic segmentation approach summarized in Section 3.2 to get semantic labels for every point of the point cloud. Our proposed approach uses semantic information for weighting of the correspondences, which lead to more consistent correspondences as moving objects will be filtered by considering the mismatch in the semantic classes. By augmenting the surfel-based map representation with semantic classes, we are able to generate a semantic map as shown in Figure 4.2.

We evaluated our approach on the KITTI odometry benchmark and demonstrate that the semantics improve the pose estimation performance considerably, i.e., using semantics we achieve a translational error of just 1.06% compared to a translational

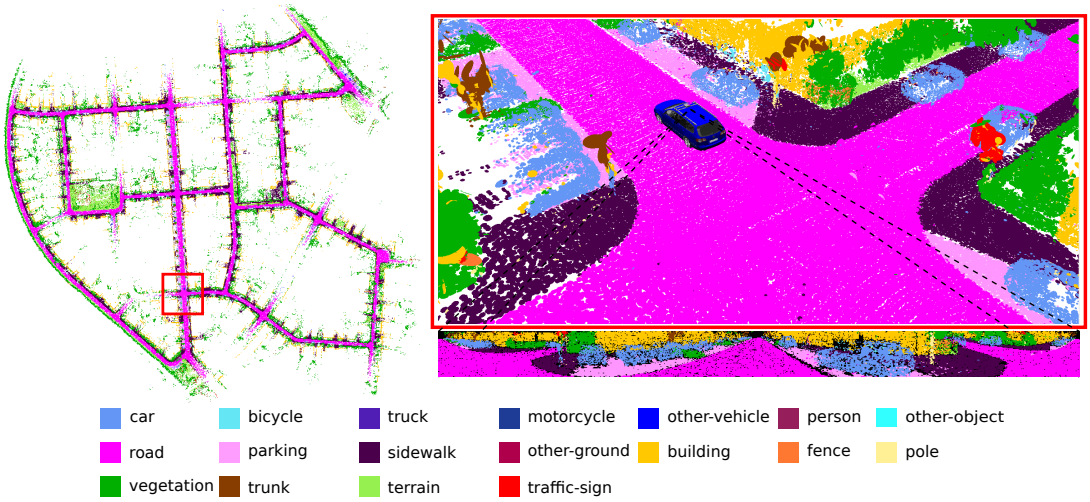


Figure 4.2: Semantic maps generated by our approach. On the left, we show a large overview of the map generated. On the right, we show a detailed view of the generated map and the range image representation used for registration at the location indicated by the red square.

error of 1.39% without using semantics on the testset of the dataset. We furthermore show that the consistency of semantics leads to more reliable pose estimation performance in cases with many moving cars in a traffic jam. Especially in such situations, where solely geometric approaches cannot distinguish between many moving objects and static background using semantic information can provide a better estimation by downweighting of probably moving objects, but also by discarding semantically wrong associations. As shown in Figure 4.2, we are additionally able to generate a semantic map that already includes much of the information that usually must be added manually, such as the road structure.

4.3 Panoptic Segmentation of 3D LiDAR Sequences

In Section 3.3, we discussed panoptic segmentation using a single 3D LiDAR scan, where one is interested in generating a semantic segmentation and determining distinct instances of objects at the same time. In this section, we discuss our solutions to extend panoptic segmentation to also provide temporally consistent panoptic segmentation, such that one can identify object instances over time.

In the context of autonomous vehicles, we have to consider not only the current interpretation of the scene at the current moment, but as outlined before also have to take into account how moving traffic participants evolve over time. Using the temporal evolution of objects, we can reason about the future positions of these objects in the scene. Knowing where objects have been in the past allows us to extrapolate where these objects might be probably in the future.

The analysis of the temporal evolution of the scene needs to take into account the own motion of the autonomous vehicle, which is achieved by the methods discussed in Chapter 2. These methods provide the relative poses between two observations given by a 3D LiDAR scan. Building on this poses, we developed approaches [A], [H] that

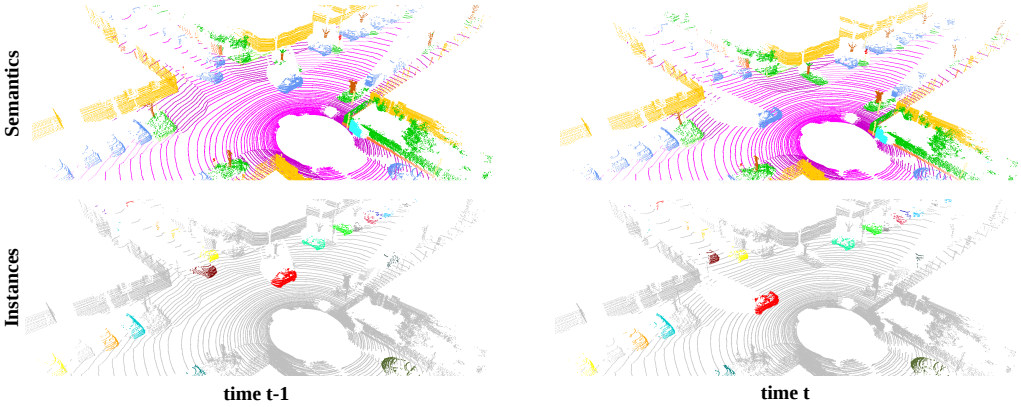


Figure 4.3: In 4d panoptic segmentation, we are interested in generating temporally-consistent semantic and instance segmentation, such that the scene gets consistent labels, but also instances can be identified over a sequence of LiDAR scans. Here, same colors of the instances indicate the same instance. Note that the static (parked cars on the side), but also moving cars (the red and brown car) get consistent instance ids as indicated by the same color.

provides a temporally consistent interpretation of a sequence of 3D LiDAR scans as shown in Figure 4.3. Besides the consistent semantic classes, we are also interested in assigning consistent instance ids, such that we can identify distinct objects over time.

In our articles on so-called 4D panoptic segmentation, we made the following contributions:

- We formalized in our paper [H], the task of 3D panoptic segmentation using 3D LiDAR sequences that produces temporally consistent panoptic segmentation providing object instances of thing classes. We, furthermore, proposed a metric to measure the performance of 4D panoptic segmentation approaches.

- In our follow-up work [A], we proposed a novel method for 4D Panoptic segmentation that provides more consistent instances over time using a contrastive learning approach building on single scan panoptic segmentation.

For providing a temporally consistent panoptic segmentation, which we termed 4D panoptic segmentation, we leveraged pose information generated by the SLAM approach described in Section 2.1 in our first paper [H]. The transformed scans are then sampled to generate a 4D point cloud, where each point not only contains the spatial coordinate, but also an index identifying the point in time when it was generated. Based on this point cloud, we train a network to predict the semantic class, embedding vectors and centers of instances that can be used in a post processing step to extract instances. Together with this approach for 4D panoptic segmentation, we also proposed a metric to measure the performance of such approaches that takes into account the consistency of semantic classes assigned to points, but also measures the consistency of instance associations over time.

In our follow-up work [A], we proposed a different paradigm to achieve temporally consistent instances. Here, we relied on strong single scan 3D panoptic segmentation and proposed an extension that takes features of the single scan panoptic network to

perform instance association over different points in time. In this work, we additionally exploited basic motion models to associate instances over time. For training of the association network, we employed a contrastive loss that ensures that the embedding vectors build from the instance features of the panoptic segmentation network are similar for the same instance and dissimilar for different instances. For the final instance association, we used the Hungarian method [65] to find a unique assignment that maximizes the similarity. Thus, the approach combines appearance-based and velocity-based matching commonly employed in tracking approaches. In contrast to tracking, we never explicitly represent designated tracks.

Experiments on a benchmark derived from our SemanticKITTI dataset [H] shows that the proposed approaches are able to provide more consistent instances over time then other baseline approaches. However, the performance of these approaches might be still suboptimal, since often instances are split or simply lost, which could lead to unexpected behavior in an autonomous driving context.

Chapter 5

Conclusion

This habilitation thesis summarizes our contributions towards LiDAR-based spatio-temporal scene understanding for autonomous vehicles that would enable autonomous cars to reason about the semantics and the objects in the vicinity of the car. It covers our work in the area of spatial interpretation using a sequence of LiDAR scans, the field of semantic interpretation using a single LiDAR scan, and finally our advances towards a spatio-temporal interpretation unifying spatial and semantic interpretation.

It covered our work in the areas of spatial interpretation provided by SLAM and localization that enables to determine the position of the car in respect to a map. Starting with our approach for SLAM using a surfel-based map, we developed approaches that can be used for localization using the concept of overlap. Furthermore, we proposed approaches to improve the mapping part by building more precise and detailed maps that can be ultimately used for more accurate localization, but also would enable to store such map representation more efficiently compared to other representations, such as surfel maps.

In the following chapter, we outlined our contributions for semantic interpretation of single LiDAR scans. By providing a large-scale dataset, the SemanticKITTI dataset, we were able to develop novel approaches for semantic, but also panoptic segmentation of LiDAR scans that pushed the boundaries of semantic perception using 3D LiDAR sensors. We also proposed an approach that could be effectively used to reduce the amount of labeled data by unsupervised pre-training.

In the last chapter, we covered our progress on unifying spatial and semantic interpretation, where we provided an overview of the contributions towards moving object segmentation, semantic SLAM, and finally 4D panoptic segmentation. Again, many of these approaches were driven by the availability of densely annotated, temporally-consistent data, but also precise relative poses. In particular, the 4D panoptic segmentation opens up a new research direction in the context of semantic interpretation of 3D LiDAR scans.

The contributions outlined in the habilitation thesis provide more accurate pose estimation using 3D LiDAR data and paved the way for research in the field of semantic scene interpretation by providing the data needed to investigate deep learning-based

approaches for semantic and panoptic segmentation. Pushing the boundaries by providing novel benchmark tasks and extending the dataset over time is still an ongoing effort to explore the possibilities of 3D LiDAR-based perception.

In particular, our contributions on 3D LiDAR-based SLAM and localization make it possible to determine the pose of an autonomous vehicle, which is needed to gain scene understanding. Our approaches for localization provide more accurate and more reliable localization results. Our contribution on adaptive robust kernels makes it possible to get more robust pose estimation results as wrong correspondences can be adaptively filtered. Lastly, our solution for map compression using an encoder-decoder architecture to compress large point clouds leads to an interesting research direction. Using the encoded representation capturing the structural content of the scene can be used for various applications, such as localization [29], but also point cloud registration [30].

Accurate and globally consistent poses generated by our SLAM approach, where critical for an efficient labeling of a large-scale dataset. The SemanticKITTI dataset and the benchmark for semantic segmentation jump started intensive research in the field of semantic scene interpretation. The dataset is also main driver for many of our follow-up works and we extended the dataset also with additional benchmarks, such as panoptic segmentation that recently got also more attention by the research community. Our approaches for semantic segmentation using range images and the kNN-based post-processing was adopted by several approaches.

By exploiting the sequential nature of the generated LiDAR scan sequences, we provide also first approaches for spatio-temporal scene interpretation. In particular, moving object segmentation leads to more accurate pose estimation as points on moving objects can be discarded and it furthermore leads to cleaner maps that only contain the static parts of the environment. The proposed semantic SLAM system can be used to generate semantic maps that can be exploited in localization, but also planning. Lastly, the novel task of 4D panoptic segmentation enables to not only gain semantics, but also temporally consistent instances over time that are needed to determine the trajectory of moving objects in the vicinity of the autonomous vehicle.

While we see considerable progress in the field of semantic perception using 3D LiDAR scans, where current approaches on semantic segmentation reach very convincing results on the SemanticKITTI dataset, there is still a significant amount of research necessary to reach the goal of full semantic understanding of traffic scenes that would enable to make reliable and safe driving possible. While our different benchmark tasks can be used to track the progress of semantic interpretation, it does not cover the diversity that would be needed to tackle autonomous driving in arbitrary scenarios. As SemanticKITTI is build on the KITTI benchmark, it covers only driving scenes in Germany at day time with relatively light traffic in the urban scenarios. It furthermore only covers a restricted set of traffic participants, such as cars, pedestrians, bicyclists, and motorcyclists. However, the advent of shared electric scooters is changing the situations that must be handled in everyday traffic. Furthermore, electric bikes that support the rider can drastically change the expected velocity and acceleration of bicyclists. Copying with that diversity of highly mobile traffic participants poses novel

challenges.

In particular, the problem of so-called long tail classes, which only occur rarely, is a problem that requires novel solutions for semantic perception. Here, a perception algorithm must deal with classes that only rarely or even never appear in the training data. Also in such cases, an perception algorithm should be able to provide some sensible recognition. This so-called open-world setting is highly relevant if we every want to reach Level 5 autonomous driving. Therefore, zero-shot learning approaches that can even with no fine-tuning can be adapted to novel tasks will be relevant in future applications.

Semantic scene understanding needs also the ability to predict future states of the environment. As understanding provides also the relation and intent of objects, it should be possible to provide a prediction based on such understanding. For prediction, it is also necessary to provide an estimate of the likelihood of a certain outcome. Therefore, we envision approaches that can provide multiple probable outcomes that can be integrated into a planning algorithms. Such probabilistic modelling requires a Bayesian estimation approach that can be used to update our belief of the scene. Unifying Bayesian reasoning with deep learning approaches for semantic perception is therefore another research direction that will become relevant in the next years.

While currently different tasks are solved in isolation, such as pose estimation and semantic interpretation, it seems needed to unify these different tasks into a single approach. While it makes sense to tackle these tasks in isolation to make progress, it seems wasteful in terms of compute to solve these tasks in isolation. In particular, pose estimation and semantic interpretation could profit from each other. However, it is still unclear how such unification could be achieved.

Bibliography

[1] K. Kirkpatrick, "Still waiting for self-driving cars," *Communications of the ACM*, vol. 65, no. 4, pp. 12–14,

[2] C. Urmson, J. Anhalt, D. Bagnell, *et al.*, "Autonomous Driving in Urban Environments: Boss and the Urban Challenge," *Journal of Field Robotics (JFR)*, vol. 25, no. 8, pp. 425–466, 2008.

[3] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art," vol. 12, no. 1–3, pp. 1–308,

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, 2015.

[5] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, ACM, 1996, pp. 303–312.

[7] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: surface elements as rendering primitives," in *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2000.

[8] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[9] C. Stachniss, J. Leonard, and S. Thrun, "Springer Handbook of Robotics, 2nd edition," in B. Siciliano and O. Khatib, Eds. Springer Verlag, 2016, ch. Chapt. 46: Simultaneous Localization and Mapping.

[10] C. Cadena, L. Carlone, H. Carrillo, *et al.*, "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age," *IEEE Trans. on Robotics (TRO)*, vol. 32, pp. 1309–1332, 6 2016.

[11] P. Besl and N. McKay, "A Method for Registration of 3D Shapes," *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, vol. 14, no. 2, pp. 239–256, 1992.

[12] M. Keller, D. Lefloch, M. Lambers, and S. Izadi, "Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion," in *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2013, pp. 1–8.

[13] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Trans. on Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, 4 2010.

[14] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *Intl. Journal of Robotics Research (IJRR)*, vol. 32, no. 11, 2013.

[16] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[17] M. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[18] S. Geman and D. McClure, "Bayesian image analysis: An application to single photon emission tomography," in *Proc. of the American Statistical Association*, 1985.

[19] E. D. Jr. and R. Welsch, "Techniques for non-linear least squares and robust regression," *Communications in Statistics-Simulation and Computation*, vol. 7, no. 4, pp. 345–359, 1978.

[20] J. T. Barron, "A General and Adaptive Robust Loss Function," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[21] D. Wilbers, C. Merfels, and C. Stachniss, "Localization with Sliding Window Factor Graphs on Third-Party Maps for Automated Driving," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[22] R. A. Newcombe, S. Izadi, O. Hilliges, *et al.*, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *Proc. of the Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[23] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D Reconstruction at Scale using Voxel Hashing," *Proc. of the SIGGRAPH Asia*, vol. 32, no. 6, 2013.

[24] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3D Mapping in Real-Time on a CPU," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2014.

[25] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large scale dense RGB-D SLAM with volumetric fusion," *Intl. Journal of Robotics Research (IJRR)*, vol. 34, no. 4-5, pp. 598–626, 2014.

[26] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 1–13, 2013.

[27] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[28] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. of Int. Conf. on 3-D Digital Imaging and Modeling*, 2001, pp. 145–152.

[29] L. Wiesmann, R. Marcuzzi, C. Stachniss, and J.Behley, "Retriever: Point Cloud Retrieval in Compressed 3D Maps," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[30] L. Wiesmann, R. Marcuzzi, J.Behley, and C. Stachniss, "DCPR: Deep Compressed Point Cloud Registration," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[31] G. Kim, B. Park, and A. Kim, "1-day learning, 1-year localization: Long-term LiDAR localization using scan context image," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, pp. 1948–1955, 2019.

[32] T. Röhling, J. Mack, and D. Schulz, "A Fast Histogram-Based Similarity Measure for Detecting Loop Closures in 3-D LIDAR Data," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 736–741.

[33] L. He, X. Wang, and H. Zhang, "M2DP: A Novel 3D Point Cloud Descriptor and Its Application in Loop Closure Detection," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[34] B. Steder, G. Grisetti, and W. Burgard, "Robust Place Recognition for 3D Range Data Based on Point Features," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2010.

[35] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 1999.

[36] G. Kim, Y. Park, Y. Cho, J. Jeong, and A. Kim, "Mulran: Multimodal range dataset for urban place recognition," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[37] Y. Zhou, G. Wan, S. Hou, *et al.*, "Da4ad: End-to-end deep attention-based visual localization for autonomous driving," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[38] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[39] T. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft COCO: Common Objects in Context," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2014, pp. 740–755.

[40] M. Cordts, S. M. Omran, Ramos, *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[41] S. Manivasagam, S. Wang, K. Wong, *et al.*, "LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[42] F. Langer, A. Milioto, A. Haag, J. Behley, and C. Stachniss, "Domain Transfer for Semantic Segmentation of LiDAR Data using Deep Neural Networks," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[43] 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 *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[44] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[45] C. Qi, K. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[46] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent Convolutions for Dense Prediction in 3D," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[47] H. Su, V. Jampani, D. Sun, *et al.*, "SPLATNet: Sparse Lattice Networks for Point Cloud Processing," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[48] L. Landrieu and M. Simonovsky, "Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[49] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic Segmentation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder, "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes," in *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[51] B. D. Brabandere, D. Neven, and L. V. Gool, "Semantic instance segmentation with a discriminative loss function," in *Deep Learning for Robotic Vision workshop, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[52] R. Chang, R. Razani, E. Taghavi, E. Li, and B. Liu, "$(AF)^2$-S3Net: Attentive Feature Fusion with Adaptive Feature Selection for Sparse Semantic Segmentation Network," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[53] M. Jaritz, T. Vu, R. d. Charette, E. Wirbel, and P. Perez, "xMUDA: Cross-Modal Unsupervised Domain Adaptation for 3D Semantic Segmentation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[54] P. Jiang and S. Saripalli, "LiDARNet: A Boundary-Aware Domain," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[55] S. Kornblith, J. Shlens, and Q. V. Le, "Do Better ImageNet Models Transfer Better?" In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[56] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN Features off-the-shelf: an Astounding Baseline for Recognition," *arXiv preprint*, vol. 1403.6382v3, 2014.

[57] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," in *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2020.

[58] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum Contrast for Unsupervised Visual Representation Learning," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[59] J. Grill, F. Straub, F. Altache, *et al.*, "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning," in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[60] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2021.

[61] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is All You Need*, 2017.

[62] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, and H. Zhao, "SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances," *arXiv preprint:2002.09147*, 2020.

[63] X. Kong, X. Yang, G. Zhai, *et al.*, "Semantic Graph Based Place Recognition for 3D Point Clouds," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[64] L. Li, X. Kong, X. Zhao, T. Huang, and Y. Liu, "SSC: Semantic Scan Context for Large-Scale Place Recognition," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[65] H. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

Publications

[A] R. Marcuzzi, L. Nunes, L. Wiesmann, *et al.*, "Contrastive Instance Association for 4D Panoptic Segmentation using Sequences of 3D LiDAR Scans," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 1550–1557, 2022. DOI: 10.1109/LRA.2022.3140439

IEEE Robotics and Automation Letters (RAL) paper presented at the
2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

Contrastive Instance Association for 4D Panoptic Segmentation using Sequences of 3D LiDAR Scans

Rodrigo Marcuzzi Lucas Nunes Louis Wiesmann Ignacio Vizzo Jens Behley Cyrill Stachniss

***Abstract*—Scene understanding is critical for autonomous navigation in dynamic environments. Perception tasks in this domain like segmentation and tracking are usually tackled individually. In this paper, we address the problem of 4D panoptic segmentation using LiDAR scans, which requires to assign to each 3D point in a temporal sequence of scans a semantic class and for each object a temporally consistent instance ID. We propose a novel approach that builds on top of an arbitrary single-scan panoptic segmentation network and extends it to the temporal domain by associating instances across time. We propose a contrastive aggregation network that leverages the point-wise features from the panoptic network. It generates an embedding space in which encodings of the same instance at different timesteps lie close together and far from encodings belonging to other instances. The training is inspired by contrastive learning techniques for self-supervised metric learning. Our association module combines appearance and motion cues to associate instances across scans, allowing us to perform temporal perception. We evaluate our proposed method on the SemanticKITTI benchmark and achieve state-of-the-art results even without relying on pose information.**

I. Introduction

In the context of mobile robotics and self-driving cars, spatial-temporal semantic scene understanding is critical since it is necessary to segment and identify the other agents in the scene but also understand how they behave over time. Different tasks related to dynamic scene understanding, i.e., object detection, semantic segmentation, instance segmentation, and multi-object tracking, are often tackled separately.

In this paper, we address the problem of 4D panoptic segmentation using LiDAR scans recorded by an outdoor platform, see also [1]. This requires to assign to each 3D point in a temporal sequence of scans a semantic class and a temporally consistent instance ID, as illustrated in Fig. 1.

In panoptic segmentation for LiDAR scans, state-of-the-art methods [17], [12], [27] rely on their strong 3D feature extractors. We argue that these features can also be leveraged to generate temporally consistent instance-wise embeddings in order to extend the segmentation to the temporal domain by means of the tracking-by-detection paradigm: identify objects in individual 3D scans and then perform temporal instance associations via feature similarity.

The main contribution of this paper is a novel method based on a sparse convolutional neural network, which takes 3D point cloud representation of LiDAR scans as input and outputs, for each point, a semantic label and a temporally consistent instance ID over the whole sequence. We build on top of a single-scan panoptic segmentation network to obtain the semantic and instance predictions as well as point-wise features. Our contrastive aggregation network takes these features as input to generate instance-wise embeddings that are consistent over time. It learns through a contrastive approach using as positive samples the same instance at several timesteps and as negative samples different instances in the same and other scans. Finally, our association module combines appearance and motion cues to perform the instance associations and generate consistent instance IDs over time. In sum, we propose an approach that is able to perform 4D panoptic segmentation of LiDAR scans using a frozen panoptic segmentation backbone and associating instances across time via feature similarity. We achieve state-of-the-art performance in 4D panoptic segmentation even without sensor pose estimates provided by a SLAM approach.

The implementation of our approach is publicly available at https://github.com/PRBonn/contrastive_association.

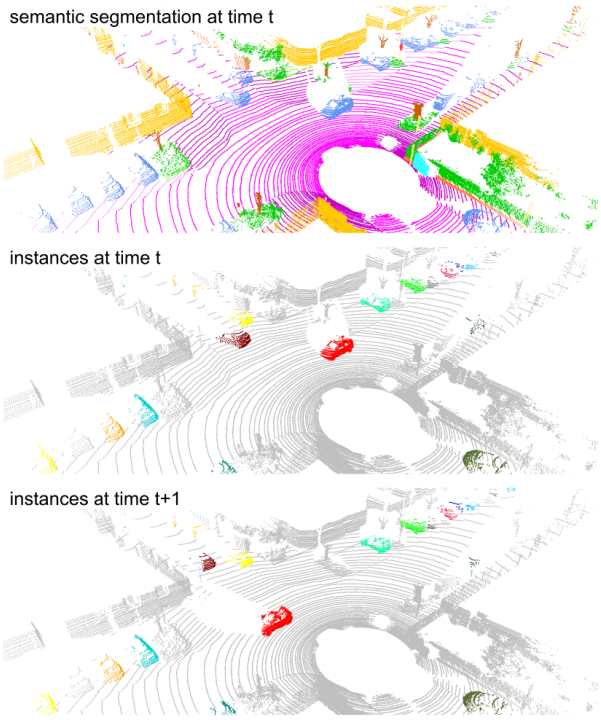


Fig. 1: Our method allows to perform semantic segmentation (top) and temporally consistent instance segmentation (bottom) of 3D LiDAR scans. The different instance colors depict different IDs.

Manuscript received: Sep 9, 2021; Revised: Nov 24, 2021; Accepted: Dec 23, 2021. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments.

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070–390732324–PhenoRob. by the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008 (Harmony). All authors are with the University of Bonn, Germany.

Copyright ©2022 IEEE

II. Related Work

Our work targets outdoor environments, for tasks such as, perception in autonomous driving, for which the point clouds are extensively used [15].

Panoptic Segmentation on Point Clouds. To unify semantic and instance segmentation, Kirillov et al. [23] propose panoptic segmentation for images and Milioto et al. [27] extend it to LiDAR point clouds. This task boils down to performing semantic segmentation of *stuff* classes and instance segmentation for *thing* classes. To deal with 3D data, some approaches project it into a 2D representation like range images [27] or bird's-eye-view [38] and use 2D convolutional networks. Another alternative is to divide the space into 3D partitions to maintain the geometric relations between the data points and apply 3D convolutional networks. Gasperini et al. [12] use point convolutions [32] to improve the results but require a higher computational cost and applying hard downsampling. Recently, Hong et al. [17] use sparse convolutions [14] and improve results by voxelizing the space using cylindrical coordinates to match the distance-dependent density of the LiDAR point clouds.

Multi-Object Tracking. Multi-object tracking in the vision domain is usually solved using the tracking-by-detection paradigm [30], [36] formulated as a data association task consisting of two steps: obtaining object detections in the current frame and associating them across time. The performance of the task relies on the quality of the appearance and motion models [6]. When using deep learning to tackle this task, the focus of the research is on learning these models and their effective combination. Some works [36] only rely on a motion model and others [11], [31] use metric learning to generate instance embeddings for data association. They provide positive and negative samples to enforce that positive pairs of detections have more similar features than negative pairs. Dong et al. [11] use a triplet loss with one positive and one negative sample. Hermans et al. [16] use hard batch triplet loss to mine a hard negative sample which is more beneficial for learning. Son et al. [31] add an extra positive sample to increase temporal consistency. We propose to leverage instance labels to train our contrastive aggregation network with several positive and negative samples to compute temporally consistent instance features and perform associations via feature similarity.

Segmentation and Tracking. There have been a few works to unify tracking and segmentation tasks. Kim et al. [22] extend panoptic segmentation [23] from 2D images to the video domain focusing on short and sparsely labeled video snippets in an offline setting. Voigtlaender et al. [35] extend the multi-object tracking task with segmentation. Aygun et al. [1] propose to use as input 4D volumes by aggregating LiDAR scans and cluster instances across time and space to perform instance segmentation and tracking simultaneously. Recently, Hurtado et al. [18] proposed a new architecture that includes a tracking head to perform the instance associations across time. Similarly, we add instance associations to a panoptic segmentation network using the extracted features from the encoder-decoder but we do not retrain it. This allows our method to leverage features from arbitrary panoptic backbones. Furthermore, we train our approach using contrastive learning to leverage several positive and negative samples instead of just one.

Contrastive Learning. Contrastive learning for self-supervised representation-learning [33], [8], [7] became popular due to the performance improvement on image classification tasks. In this setup, a data point is used as an anchor, positive samples are augmented views of it and the negative samples are the augmented views of other data points in the batch. The loss seeks to compare the generated encodings for each sample in an unsupervised way without labels. The goal is to learn a generic feature representation in which embeddings from positive samples lie close together and far from embeddings of negative samples. After this pretraining, the network is fine-tuned to solve a downstream task. Due to the availability of instance labels, we adopt the supervised contrastive loss [21] to use several positive along with many negative samples. However, the positive samples come from other timesteps of the same anchor instance and are not generated through augmentations of it.

In sum, our method is based on a panoptic segmentation backbone followed by temporal instance association via feature similarity learned using positive and negative samples.

III. Our Approach

The goal of our approach is to achieve 4D panoptic segmentation of LiDAR scans, which means tackling simultaneously semantic and instance segmentation in the *3D spatial* and *1D temporal* domain. We achieve this by predicting, for each 3D point in the LiDAR scan, a semantic label, and for each object, a temporally consistent instance ID. To address this task, we follow prior work [36], [18] and use the tracking-by-detection paradigm [30] to identify objects in single 3D scans and then perform temporal associations via feature similarity.

Fig. 2 gives an overview of our approach. We adopt the point cloud representation of LiDAR scans, which represents each point by its 3D coordinates. We use a frozen panoptic segmentation network (backbone), to obtain semantic predictions, instance predictions, and features for every point in the scan. Using the instance predictions, we select the points and features belonging to each instance and apply our contrastive aggregation network (CA-Net) to produce instance-wise features. Finally, our association module assigns instance IDs, which are consistent throughout the whole temporal sequence, by combining appearance information from the instance-wise features and motion information using a constant velocity motion model. Combining this with the semantic predictions, we obtain 4D panoptic segmentation.

To be able to learn descriptive features, the CA-Net is trained with the supervised contrastive loss [21]. Using the instance labels, we identify the same object over time and use the same instance in different scans as positive samples. The negative samples are all other instances in the same and other scans in the training batch.

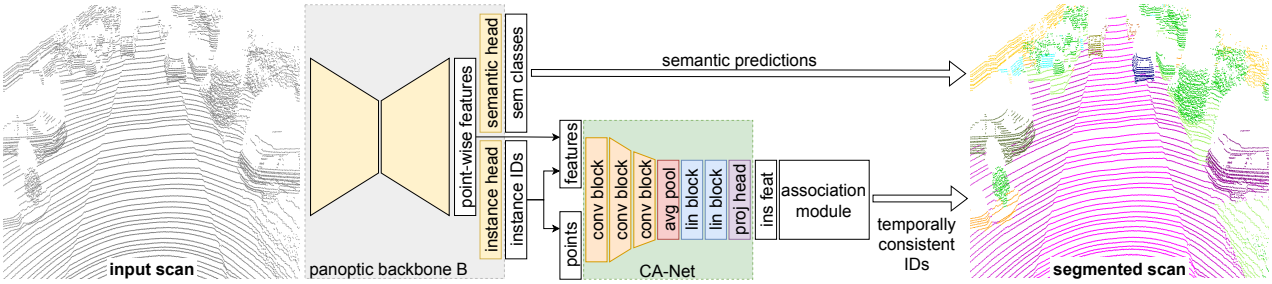


Fig. 2: Overview of our method. Given the current 3D LiDAR scan, we use a panoptic segmentation backbone B to obtain semantic classes, instance IDs, and features for each point. We select the points and features for each instance using the IDs and input them into the CA-Net to obtain instance-wise features. These features are used to perform instance associations using our association module to assign temporally consistent instance IDs. Combining this with the semantic predictions, we obtain 4D panoptic segmentation.

A. Panoptic Segmentation Backbone

The first element of our approach is a frozen panoptic segmentation network to obtain semantic predictions, instance predictions, and point-wise features. To achieve panoptic segmentation, bottom-up methods [27], [12], [38], [17] use a shared feature extractor to obtain point-wise features based on convolutions and apply task-specific heads to obtain semantic and instance features. The instance features are clustered in a post-processing step to produce instances.

Given a point cloud $\mathcal{P} = \{\underline{\mathrm{p}}_1^C, \dots, \underline{\mathrm{p}}_N^C\}$ with point coordinates $\underline{\mathrm{p}}_i^C \in \mathbb{R}^3$, we apply the backbone B and obtain point-wise semantic classes $\mathcal{P}^S = \{p_1^S, \dots, p_N^S\}$, where $p_i^S \in \{1, \dots, n_{\text{classes}}\}$ and point-wise instance IDs $\mathcal{P}^I = \{p_1^I, \dots, p_N^I\}$ with $p_i^I \in \mathbb{N}$ from the task specific heads. From the feature extractor, we obtain the point-wise features $\mathcal{P}^F = \{\mathbf{p}_1^F, \dots, \mathbf{p}_N^F\}$, where $\mathbf{p}_i^F \in \mathbb{R}^{D_B}$ with feature dimension D_B. We use the ID of each point to select for each instance $j \in \{1, \dots, M\}$, its points $I_j^P = \{p_i^C \in \mathcal{P} \mid p_i^I = j\}$ and its point-wise features I_j^F, which we use as input of our CA-Net.

In this paper, we use DS-Net [17] as backbone B. Since we are not modifying nor retraining the backbone and only use it to get predictions and features, our approach can potentially be applied to other panoptic segmentation networks.

B. Contrastive Aggregation Network

Our contrastive aggregation network (CA-Net) generates temporally consistent appearance features for instance associations. Given the input points I_j^P and point-wise features I_j^F for all M instances in the current point cloud $\mathcal{P}$, the output of the network are instance-wise features $I^F = \{\mathbf{f}_1, \dots, \mathbf{f}_M\}, \mathbf{f}_j \in \mathbb{R}^{D_A}$ with feature dimension D_A, i.e., a single feature vector for each instance in the scan.

The point-wise features from the backbone capture more general and high-level information, not specific to the instances. We are only interested in encoding information related to the individual instances. To learn from the shape of the instances and the relations between their points, we first apply three convolutional blocks with stride 2 to reduce the spatial dimension while increasing the feature dimension. This increases progressively the receptive field and allows us to capture low-resolution features to get information about the whole instance. After applying the convolutions, a pooling layer computes a single feature vector from all the features belonging to the instance points I_j^P, which summarizes the information of each point in a single instance feature. This is followed by two linear blocks and a projection head to obtain the final instance-wise feature $\mathbf{f}_j$. The goal is to project the intermediate feature representation into an embedding space, where embeddings belonging to the same instance across time have a high similarity and embeddings belonging to different instances have low similarity.

Due to the inherent sparsity of the data, we use sparse convolutions [9] to process the voxelized scan as it leads to better performance and reduces computational cost [14].

C. Association Module

After computing instance-wise embeddings I^F using our CA-Net, we maintain consistent instance-wise IDs by associating the instances $\{I_1 \dots I_M\}$ in the current scan with the corresponding instances $\{I_1 \dots I_K\}$ in the previous ones. To achieve this, we follow a similar procedure as previous works [36], [6]. We compute a cost matrix $\mathbf{C} \in \mathbb{R}^{M \times K}$ between all M instances in the current scan and all K instances identified in previous scans, by means of the similarity between their features. Then, the problem can be formulated as a bipartite graph matching problem that can be solved using the Hungarian method [24] to determine the pairs of associations between current and previous instances.

Tracking instances in consecutive frames is only part of the problem. We need to manage tracked objects that might leave the scene, newly detected objects that are not explained by any trajectory and re-identify objects, which have been occluded or missed in intermediate frames. To handle new targets, we add detected objects in the current frame only if the feature similarity with any of the already active instances is lower than a threshold T_{new}. For re-identification of objects, we store the deactivated trajectories for a fixed number of scans n_{old} and compare with the deactivated targets. We re-identify objects if the similarity is higher than a threshold T_{old}.

To add information about the movement of the targets, we apply a constant velocity assumption for all objects by adding a constant velocity motion model independent of the sensor ego-motion. This means that the model does not estimate the ego-motion but the motion of all the objects in the scene. We fuse the appearance and motion information in the cost matrix as a linear combination of the feature cost and the

center distance between current instances and the predicted positions of the previous instances. Each entry of the cost matrix $\mathbf{C}$ is an association cost between the new instance m and the previous instance k:

$$\mathbf{C}_{m,k} = \alpha_f \cdot (1 - sim(\mathbf{f}_m, \mathbf{f}_k)) + \alpha_d \cdot \|\mathbf{c}_m - \mathbf{c}_k\|, \quad (1)$$

where $\mathbf{f}_m$ is the feature and $\mathbf{c}_m \in \mathbb{R}^3$ the center coordinates of instance m, $sim(\cdot)$ is the cosine similarity function $sim(\underline{\mathbf{f}}_m, \underline{\mathbf{f}}_k) = \underline{\mathbf{f}}_m^\top \underline{\mathbf{f}}_k / \|\underline{\mathbf{f}}_m\| \|\underline{\mathbf{f}}_k\|$, and α_f, $\alpha_d \in \mathbb{R}$ are importance weights for the individual feature and distance costs. For re-identification, the motion model is applied continuously also to the deactivated targets (not associated with any other instance) to estimate their position in case of occlussions or missing detections.

D. Input Features

The point-wise features $\mathcal{P}^F$ from the panoptic segmentation backbone are used by the task-specific heads to perform segmentation of the points. They are similar for instances belonging to the same semantic class and thus cannot be directly used to perform instance associations for objects of the same semantic class.

Inspired by the recent findings in natural language processing [34], we seek to enhance the point-wise features with spatial knowledge using positional encodings. We compute for each instance point $\mathbf{p}_l \in I_j^P$ a fixed positional encoding $\mathbf{e}_l \in \mathbb{R}^{D_B}$. We generate for each point a Fourier feature position encoding [20] with the same dimension as the point-wise features. We select a maximum frequency $\max_{\text{freq}}$ and sample, for each coordinate, from a series of sines and cosine functions with frequencies evenly spaced in $[0, \max_{\text{freq}}]$ on a logarithmic scale. Then, we use a padding of zeros to reach the same dimension as the point feature. To get the final input to the network, the feature and the positional encoding are combined by performing the addition of both. Due to their similarity, using the point-wise features as the only input to our CA-Net does not provide enough information to learn distinct instance-wise features to do the associations. By adding the positional encodings as extra information, we exploit the spatial relations between the instances and create more distinct features, and help the network learn better instance-wise features for this task.

E. Instance Point Extraction

As the input of the CA-Net, we select from all the points and features (P, P^F) in the current point cloud $\mathcal{P}$, the ones belonging to the different instances (I^P, I^F). During training, we rely on the instance labels $\hat{\mathcal{P}}^I$ to group the points into instances and select their corresponding features. At inference time the instance points are selected using the predictions P^I from the backbone. These predictions are not perfect due to problems like wrong semantic predictions or errors in the clustering algorithm used to separate instances in the backbone after the instance head. The inputs for the CA-Net are then different at train and test time.

We can circumvent this by applying augmentations to the point instances to imitate what may happen during inference. We design specific augmentations to deal with the problems commonly observed at test time, mainly splitted instances and incomplete instances. Compared to other kinds of augmentations applied to the full point cloud, in our case we implement instance-wise augmentations since these are the input to our network.

The problem of splitted instances is due to the imperfect clustering applied in the backbone to obtain the instance predictions. It can wrongly cluster the points into instances, dividing one instance into two smaller ones, as shown in Fig. 3. To mimic this, we create an augmentation, which we call *split*, to separate the points on each side of an imaginary plane to split the instance. We sample a random unit normal vector $\mathbf{n} \in \mathbb{R}^3, \|\mathbf{n}\| = 1$ and a random instance point $\mathbf{p}_l \in I_j^P$ to generate a randomly oriented plane with Hessian normal form $\mathbf{n}^\top \mathbf{p}_l = d$, where d is the distance from the plane to the origin. The remaining points after the augmentation are the query points $\mathbf{q}_l$ which lie in the half-space of the normal:

$$I_{aug} = \left\{ \mathbf{q}_l \in I_j^P \mid \mathbf{n}^\top \mathbf{q}_l^C - d > 0 \right\}, \quad (2)$$

which leads to splitted instances.

The incomplete instances are caused by the wrong semantic class predicted for some instance points, usually at the borders of the instances. Their semantic class assigns them to the background, and they are not considered as part of the instance, as can be seen in Fig. 3. To address it, we create an augmentation, which we call *contour*, to discard points in the contour of the instances. We first normalize the point coordinates to the range $[-1, 1]$, generate a random maximum coordinate value $\gamma \in \mathbb{R}$ and save the indices of the points with smaller absolute coordinates that γ. We use these indices to select the instance points (with unnormalized coordinates) to keep after the augmentation, i.e.,

$$I_{aug} = \left\{ \mathbf{p}_l \in I_j^P \mid |\tilde{x}_l| < \gamma \wedge |\tilde{y}_l| < \gamma \wedge |\tilde{z}_l| < \gamma \right\}, \quad (3)$$

where $\tilde{x}_l, \tilde{y}_l, \tilde{z}_l$ are the normalized coordinate components of l^{th} point.

We illustrate our augmentations applied to instances in Fig. 3. It is important to notice that the augmentations do not change the position of any point but rather drop some of them. As an extra augmentation, we randomly drop cuboids of points belonging to the instance [37]. We discuss the influence of these augmentations in Sec. V-C.

F. Pose Information

Both, the positional encoding, which adds spatial knowledge to the computed features, and the constant velocity motion model, rely on the positions of the instances in the current scan, i.e., positions in a local coordinates frame. As these are local coordinates, the positions are not consistent for scans at different timesteps and the ego-motion of the sensor must be compensated.

By adding the sensor pose estimates using a SLAM approach [5], we can improve the instance associations by leveraging this extra information. We use the pose estimates to transform the positions of previously observed objects into the current local coordinates frame. This way, objects have

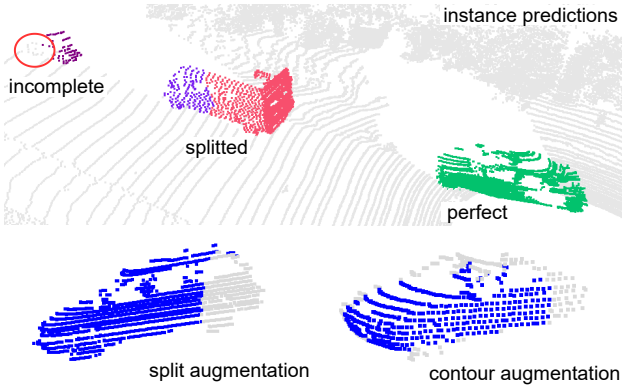


Fig. 3: Problems in the instance predictions using the backbone. Incomplete, splitted and perfect instance (top). Proposed *split* augmentation (left) and *contour* augmentation (right). The gray points are discarded.

consistent positions over time and the constant velocity motion model only predicts the motion of the other objects as the ego-motion is compensated. We recompute the features of the previously observed instances by updating their positional encoding using the consistent positions in the current coordinate frame. We obtain more similar features for the same instance at different points in time, which improves the instance associations as illustrated in Sec. IV-B

G. Contrastive Training

For the multi-object tracking problem, we use the instance head to get the predicted instances $\{I_1, \dots, I_M\}$ in the current point cloud $\mathcal{P}$ and associate them with previous instances across time. Several works [18], [16], [11], [31] use metric learning to learn instance representations by comparing the embedding of one anchor object with one or a few positive and negative samples. Contrastive learning is used in self-supervised representation learning [33], [8] to train a network, which is later fine-tuned on a downstream task. The main idea of those approaches is to use data augmentation to generate two versions of one anchor sample and train the network to learn to pull the embeddings of these samples (positives) together and push them away from all other samples (negatives).

Given a batch of samples $\mathcal{B}$, the loss function seeks to discriminate between the positive pairs (augmented versions of the anchor) and the negatives (augmented versions of different anchors). In the self-supervised contrastive loss InfoNCE [33], an encoder is used to obtain the feature vectors $\mathbf{z}_j$ for each augmented sample j. Let $r(j)$ be the index of the other augmented sample from the same anchor j and $A(j)$ the set of all indices in the batch except j. Then, the loss function takes the form:

$$\mathcal{L}_{self} = -\sum_{j \in \mathcal{B}} \log \frac{\exp\left(\mathbf{z}_j^\top \mathbf{z}_{r(j)}/\tau\right)}{\sum_{a \in A(j)} \exp\left(\mathbf{z}_j^\top \mathbf{z}_a/\tau\right)}, \tag{4}$$

where τ is a temperature parameter.

In our setup, the samples are all instances in the batch and their corresponding feature vectors $\{\mathbf{f}_1, \dots, \mathbf{f}_b\}$ are computed using our CA-Net. We want to enforce that features of the same instance are consistent. The appearance of the instances changes over time as the viewpoint changes due to the instance's and the sensor's motion. As the samples for the same instance are different across time but depict the same object, we can use this as an implicit augmentation. To obtain positive samples, we select the same instance in different scans over time instead of using one instance as an anchor and generating augmented versions of it.

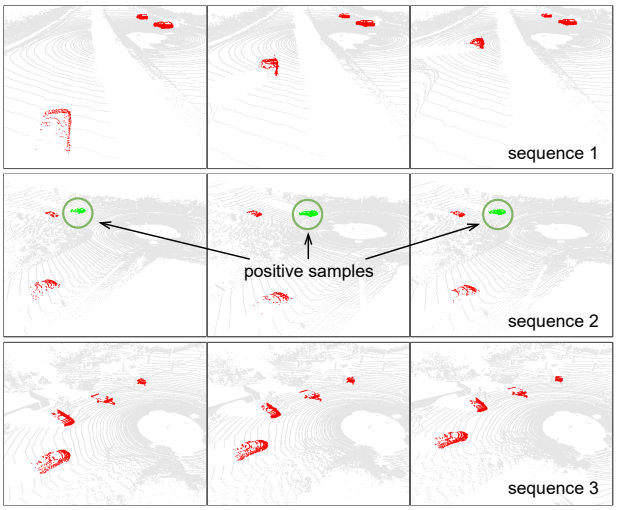


Fig. 4: In our contrastive setup, for each instance, positives samples are the same instance in different scans (green) in a sequence, and negatives samples are all the other instances (red) in all scans and all other sequences in the batch.

The appearance can, however, change significantly in scans temporally far from each other. We define a temporal window of scans $\Delta \in \mathbb{N}$, in which we consider instances similar enough to perform the associations and from which the positive samples are drawn. As Δ is the number of scans from which we extract instances, it is the maximum number of positive samples to consider in the loss function. Using the labels, we select a set of positive samples $P(j)$ considering the instances with the same ID in consecutive scans in the temporal window. As negative samples, we use all other instances in the batch, as depicted in Fig. 4. To learn from many positive in addition to the many negative samples, we use the supervised contrastive loss [21]:

$$\mathcal{L}_{sup} = -\sum_{j \in \mathcal{B}} \frac{1}{|P(j)|} \sum_{p \in P(j)} \log \frac{\exp\left(\mathbf{f}_j^\top \mathbf{f}_p/\tau\right)}{\sum_{a \in A(j)} \exp\left(\mathbf{f}_j^\top \mathbf{f}_a/\tau\right)}, \tag{5}$$

where $\mathbf{f}_j, \mathbf{f}_p, \mathbf{f}_a \in \mathbb{R}^{D_A}$ are respectively the feature for the anchor instance, the feature for the positive samples and, the feature for all the other instances in the batch.

We are leveraging the properties of the supervised contrastive loss for our task. First, we can use an arbitrary number of positives which are determined by the temporal window. Second, we increase the number of negatives compared to other losses by considering all other instances in the same and other scans. Third, we benefit from the intrinsic ability of the loss to perform hard positive/negative mining and avoid the need for explicit hard negative mining.

Method	LSTQ	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}
RangeNet++[28] + PP + MOT	35.52	24.06	52.43	64.52	35.82
KPConv [32] + PP + MOT	38.01	25.86	55.86	**66.90**	47.66
RangeNet++[28] + PP + SFP	34.91	23.25	52.43	64.52	35.82
KPConv [32] + PP + SFP	38.53	26.58	55.86	**66.90**	47.66
4DPLS[1]	56.89	56.36	57.43	66.86	51.64
Ours (without pose estimates)	60.04	59.49	**60.60**	66.88	**51.98**
Ours (with pose estimates[5])	**63.11**	**65.71**	**60.60***	66.88*	**51.98***

TABLE I: 4D panoptic segmentation on SemanticKITTI test set. MOT [36]; SFP - scene flow based propagation [29]; PP - PointPillars [25]. Numbers with * denote the same segmentation results. Adding pose information to our single-scan panoptic backbone does not influence the segmentation performance.

H. Implementation Details

We build on top of DS-Net [17] as panoptic segmentation backbone. The feature extractor provides point-wise feature vectors $\mathbf{p}_i^F \in \mathbb{R}^{D_B}, D_B = 128$. For the instance clustering, we use mean shift [10].

For our CA-Net, the convolutional blocks are 3D sparse convolutions [9], followed by a batch normalization layer [19] and leaky ReLU [26] as activation layer. The sparse linear blocks consist of a linear layer followed by a batch normalization layer and a leaky ReLU as activation layer. The final projection head is a linear layer that projects the intermediate instance-wise embeddings into a feature space of dimension $D_A = 1024$ in which we use feature similarities to associate instances across time.

We keep deactivated targets for $n_{old} = 8$ frames and use different thresholds to handle new targets and re-identification for the appearance and the motion model. For the feature similarity, we use $T_{new_f} = T_{old_f} = 0.7$ and for the center distance, we use $T_{new_d} = T_{old_d} = 2$. The weights for the feature cost and the center distance in Eq. (1) are $\alpha_f = 0.4$ and $\alpha_d = 0.7$. In the loss function, see Eq. (5) we use $\tau = 0.1$. At each step, the positive and negative samples are selected from a sequence of scans of random length $\Delta \in [2, 5]$. This way, at each step, the loss considers samples from sequences of variable length, which provides a different number of instances.

IV. Experimental Evaluation

We present our experiments to show the capabilities of our method and to support our statement that our approach achieves state-of-the-art performance in 4D panoptic segmentation of LiDAR scans. Furthermore, we show that using a frozen panoptic segmentation backbone and associating instances across time via feature similarity allows us to outperform previous methods that strongly rely on ego-motion estimates provided by SLAM techniques. Moreover, we illustrate that the performance of our approach increases when including the pose information.

A. Experimental Setup

We evaluate our method on SemanticKITTI [3], [2], which consist of 22 sequences from the KITTI odometry dataset [13]. Sequences 00 to 10 are used for training, leaving sequence 08 as validation set and the test set consists of sequences 11 to 21. The provided annotations are point-wise semantic and instance labels [4]. To evaluate our performance, we use the LiDAR Segmentation and Tracking Quality (LTSQ) metric $LSTQ = \sqrt{S_{\text{cls}} \times S_{\text{assoc}}}$ [1]. It consists of two terms, the classification score S_{cls} and the association score S_{assoc}. Since our panoptic segmentation backbone is frozen and we use its single-scan predictions, our results do not change the segmentation results represented by the S_{cls} term. Thus, we focus instead on the S_{assoc} term to evaluate the quality of our associations.

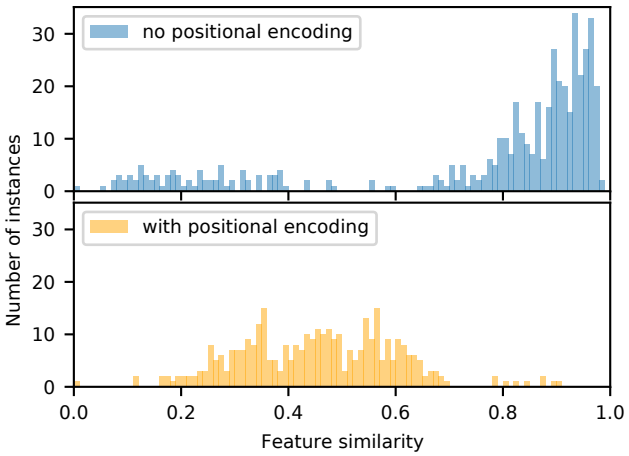


Fig. 5: Histograms of similarities between instances features in a sequence of scans. Without positional encoding (top), the histogram shows a high density at high similarities. Hence, there is a large number of instances with similar features, which makes it difficult to identify the same instance in different scans as different other instances in the sequence have similar features. With the positional encoding (bottom), most of the feature similarities are in the middle range and only a few features have high similarity. Therefore, features from different instances are more distinct, which makes them better suited for associating instances.

B. 4D Panoptic Segmentation Results

The first experiment evaluates the performance of our approach on the SemanticKITTI test set and supports the claim that we achieve state-of-the-art results on the 4D panoptic segmentation task. In this experiment, we compare our approach with the recent and high-performing method by Aygun et al. [1] and baselines using semantic segmentation networks [32], [28] and combining the predictions from the PointPillars (PP) object detector [25] with a constant velocity motion model [36] or using scene flow propagation (SFP) [29]. Note that the approach by Aygun et al. [1] and the baselines rely on the sensor poses to either aggregate scans or transform them into a global coordinates frame where objects have consistent positions and are easier to associate. For these approaches, this knowledge is a pre-requisite.

We outperform all previous methods *without* relying on the pose estimations from a SLAM approach. See Tab. I. Both experiments using our method show the same segmentation performance S_{cls}, IoU^{St}, and IoU^{Th} since we use the semantic predictions from the same single-scan backbone and the pose information does not modify semantic segmentation results of *a single scan*. More detailed results are shown in Tab. I.

instance feature		max pooling	average pooling	CA-Net
backbone features	positional encoding			
✓		15.5	31.8	59.9
	✓	32.1	52.2	58.4
✓	✓	**36.9**	**59.5**	**70.4**

TABLE II: Association performance S_{assoc} using different instance-wise features for association in SemanticKITTI validation set.

V. Ablation Studies

A. Positional Encoding

First, we analyze the influence of adding the positional encoding to the point-wise features.

We show the similarity between point-wise features from the backbone. To this end, we generate instance-wise features $\mathbf{f}_j$ by averaging the point-wise features for each instance:

$$\mathbf{f}_j = \frac{1}{|I_j^F|} \sum_{\mathbf{p}_l \in I_j^P} \mathbf{p}_l^F, \tag{6}$$

and compute their similarities.

In a sequence of scans, there is a large number of highly similar instance features, which makes it hard to distinguish them, as shown Fig. 5 (top).

To illustrate how the spatial encoding helps, we add the positional encoding generated from the point coordinates as explained in III-D, to each point-wise feature and then average them to get a instance feature:

$$\mathbf{f}_j = \frac{1}{|I_j^F|} \sum_{\mathbf{p}_l \in I_j^P} \mathbf{p}_l^F + \mathbf{e}_l, \tag{7}$$

where $\mathbf{p}_l^F \in \mathbb{R}^{D_B}$ is the point-wise feature from the backbone and $\mathbf{e}_l \in \mathbb{R}^{D_B}$ is the positional encoding for the l^{th} instance point. As can be seen in Fig. 5 (bottom), when adding the positional encoding, the amount of similar instance-wise features is visibly reduced. Thus, instances of different objects have a large distance in the feature space.

B. Feature Design

In the next experiment, we have a closer look at our feature design decisions. We generate instance-wise feature vectors and associate the instances using only their features similarity, without relying on any other information to better show the performance. Tab. II shows the association performance of the different feature options, namely pooling the features and positional encodings and using them as input to the CA-Net to obtain the instance-wise features.

As reflected in the experiments, the information about the position of the instances plays a crucial role in the association process. However, the features from the backbone allow us to exploit some extra information but, as discussed in Sec. III-D, they are similar for objects of the same semantic class and adding a positional encoding improves the performance. Lastly, adding the positional encoding to each point $\mathbf{p}_l + \mathbf{e}_l$, we obtain even better results. This shows that both parts are important and that leveraging the point-wise features from the backbone improves the instance association performance.

augmentations			S_{assoc}
split	contour	cuboids	
			49.9
✓			60.9
	✓		54.6
		✓	52.6
✓	✓		62.3
✓	✓	✓	**63.6**
instance predictions			50.2

TABLE III: Influence of the selection of input points and the different augmentations in the association quality S_{assoc}.

#	feature	encoding	motion model	poses	S_{assoc}
A	✓				59.9
B	✓	✓			70.4
C	✓	✓		✓	71.2
D			✓		68.0
E	✓	✓	✓		71.7
F	✓	✓	✓	✓	**72.9**

TABLE IV: Influence of the different components of the approach in S_{assoc} on SemanticKITTI validation set.

C. Instance Augmentations

In this experiment, we show that using instance labels to select point-wise features can lead to suboptimal results and how our proposed augmentations improve this. For this experiment, we fix the temporal window $\Delta = 3$, do not use the sensor poses nor motion model, and perform the associations using only feature similarity.

As shown in Tab. III, training using the labels to select the instance points gives the worst results and using the predictions only helps to increase S_{assoc} by 0.3 percent points. Each individual augmentation outperforms training with the labels or the predictions. Particularly, the *split* augmentation has the strongest influence. Combining all three augmentations results in a 13.7 percent points improvement compared to the training using only the labels.

D. Method Components

Tab. IV shows the influence of different components on on the performance in terms of S_{assoc} on the validation set.

In (A), we only use the point-wise features from the backbone as input to our CA-Net and perform the associations via instance feature similarity. Since no spatial information is included, instances at different positions in the scan can be wrongly associated. In (B), we add the positional encoding to the input features, see Sec. III-D, which adds spatial information, visibly improving the results. In (C), we add the SLAM poses [5] to update the previous instance positional encoding, see Sec. III-F. This way, objects have consistent positions and are easier to associate. In (D), we use only the constant velocity motion model to perform the associations. This result is better than (A) and shows that motion and appearance models are necessary to achieve good association results. In (E), we add the motion model to (B). We combine cues from the appearance and motion models, improving the results obtained when the models are used separately. In (F), we add the sensor ego-motion poses estimates and obtain the best performance.

VI. Conclusion

We presented a novel approach to perform 4D panoptic segmentation on 3D LiDAR scans. We identify objects in the current scene and associate them with previously seen targets across time combining appearance and motion information. We show that it is possible to leverage the point-wise features from a single feature extractor to tackle concurrently the tasks of semantic and instance segmentation over time. Furthermore, generating a descriptive appearance model improves the instance association and allows our approach to outperform previous methods strongly relying on ego-motion estimates. We evaluated the different parts of our approach and provided comparisons to other existing techniques. The experiments suggest that we are able to learn a descriptive representation for each instance that in turn allows us to perform temporal associations and extend a panoptic segmentation backbone to perform 4D panoptic segmentation.

References

[1] M. Aygun, A. Osep, M. Weber, M. Maximov, C. Stachniss, J. Behley, and L. Leal-Taixe. 4D Panoptic LiDAR Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, J. Gall, and C. Stachniss. Towards 3d lidar-based semantic scene understanding of 3d point cloud sequences: The semantickitti dataset. *Intl. Journal of Robotics Research (IJRR)*, 40(8-9):959–967, 2021.

[3] 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 Intl. Conf. on Computer Vision (ICCV)*, 2019.

[4] J. Behley, A. Milioto, and C. Stachniss. A Benchmark for LiDAR-based Panoptic Segmentation based on KITTI. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[5] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[6] P. Bergmann, T. Meinhardt, and L. Leal-Taixe. Tracking Without Bells and Whistles. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[7] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9912–9924, 2020.

[8] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2020.

[9] C. Choy, J. Gwak, and S. Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 24:603–619, 2002.

[11] X. Dong and J. Shen. Triplet Loss in Siamese Network for Object Tracking. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2018.

[12] S. Gasperini, M.N. Mahani, A. Marcos-Ramiro, N. Navab, and F. Tombari. Panoster: End-To-End Panoptic Segmentation of LiDAR Point Clouds. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):3216–3223, 2021.

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

[14] B. Graham, M. Engelcke, and L. van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 2020.

[16] A. Hermans, L. Beyer, and B. Leibe. In Defense of the Triplet Loss for Person Re-Identification. *arXiv preprint:1703.07737*, 2017.

[17] F. Hong, H. Zhou, X. Zhu, H. Li, and Z. Liu. LiDAR-Based Panoptic Segmentation via Dynamic Shifting Network. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[18] J. Hurtado, R. Mohan, W. Burgard, and A. Valada. MOPT: Multi-Object Panoptic Tracking. *arXiv preprint:2004.08189*, 2020.

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint*, abs/1502.03167, 2015.

[20] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver: General Perception with Iterative Attention. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2021.

[21] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised Contrastive Learning. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 18661–18673, 2020.

[22] D. Kim, S. Woo, J. Lee, and I.S. Kweon. Video Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[23] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[24] H. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[25] A. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] A.L. Maas, A.Y. Hannun, and A.Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[27] A. Milioto, J. Behley, C. McCool, and C. Stachniss. LiDAR Panoptic Segmentation for Autonomous Driving. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[28] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[29] H. Mittal, B. Okorn, and D. Held. Just Go With the Flow: Self-Supervised Scene Flow Estimation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[30] Reid, Donald B. An algorithm for tracking multiple targets. *IEEE Trans. on Automatic Control*, 1979.

[31] J. Son, M. Baek, M. Cho, and B. Han. Multi-Object Tracking With Quadruplet Convolutional Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[32] H. Thomas, C. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[33] A. van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv preprint:1807.03748*, 2019.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[35] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B.B.G. Sekar, A. Geiger, and B. Leibe. MOTS: Multi-Object Tracking and Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[36] X. Weng, J. Wang, D. Held, and K. Kitani. 3D Multi-Object Tracking A Baseline and New Evaluation Metrics. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[37] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra. Self-Supervised Pretraining of 3D Features on any Point-Cloud. *arXiv preprint:2101.02691*, 2021.

[38] Z. Zhou, Y. Zhang, and H. Foroosh. Panoptic-PolarNet: Proposal-Free LiDAR Point Cloud Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[B] L. Nunes, R. Marcuzzi, X. Chen, *et al.*, "SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segement Discrimination," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 2116–2123, 2022. DOI: 10.1109/LRA.2022.3142440

IEEE Robotics and Automation Letters (RAL) paper presented at the
2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination

Lucas Nunes Rodrigo Marcuzzi Xieyuanli Chen Jens Behley Cyrill Stachniss

***Abstract*—Semantic scene interpretation is essential for autonomous systems to operate in complex scenarios. While deep learning-based methods excel at this task, they rely on vast amounts of labeled data that is tedious to generate and might not cover all relevant classes sufficiently. Self-supervised representation learning has the prospect of reducing the amount of required labeled data by learning descriptive representations from unlabeled data. In this paper, we address the problem of representation learning for 3D point cloud data in the context of autonomous driving. We propose a new contrastive learning approach that aims at learning the structural context of the scene. Our approach extracts class-agnostic segments over the point cloud and applies the contrastive loss over these segments to discriminate between similar and dissimilar structures. We apply our method on data recorded with a 3D LiDAR. We show that our method achieves competitive performance and can learn a more descriptive feature representation than other state-of-the-art self-supervised contrastive point cloud methods.**

I. Introduction

Fine-grained scene interpretation is crucial to understand the surroundings of a robot or autonomous vehicle. Given the extensive research effort on convolutional neural networks (CNNs) [23], [56], [5], 2D image tasks have been defined and adapted to this scope, e.g., instance segmentation [33], [53] or panoptic segmentation [32], [57]. For 3D scene understanding, we can employ semantic segmentation [58], [43] to achieve a pixel-level classification of an image. Such semantic information is important for autonomous systems to interpret and safely interact with their surroundings.

Image data however does not provide straightforward 3D information, which is also essential in this domain. LiDAR sensors provide 3D information in the form of spatial positions and intensity values as point-wise information. Such data is often more challenging to interpret than images due to the lack of color information and the sparser representation of the objects. Therefore, many recent studies [34], [38], [40], [19], [10], [44] focus on convolution operations applied to point clouds to boost their performance.

Supervised methods need a substantial amount of labeled data to achieve high performance, which is particularly hard to acquire for LiDAR data. Data annotation is expensive [3] due to the density of measurements, which are sensor-specific and also depend on the sensor mounting. For semantic segmentation, point-wise labeling is required, making it even harder to annotate. There are also fewer labeled LiDAR datasets available compared to image datasets [11], [30]. Given the sensor-specific characteristics, learned features from one dataset cannot be easily transferred to a dataset with a different sensor setup. Therefore, recent work in this domain address label efficiency [51], [27], [47] or transferability across datasets [28], [29], [52], [1], [35].

Self-supervised representation learning has the prospect of using the data in an unsupervised way to learn a robust feature



Fig. 1: Results trained with only 0.1% of the labels, trained from scratch (middle), i.e., without any pre-training, and pre-training with our approach SegContrast (bottom). With SegContrast pre-training, the semantic segmentation can better delineate the structural information in the scene and better classify the more fine-grained objects, like trees, traffic signs and poles, highlighted by black arrows.

Manuscript received: Sep 8, 2021; Revised: Nov 29, 2021; Accepted: Dec 31, 2021. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments.

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070–390732324–PhenoRob. by the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008 (Harmony).
All authors are with the University of Bonn, Germany.
Copyright ©2022 IEEE

presentation that can be transferred to different downstream supervised tasks. One of the self-supervised learning methods that received increasing attention recently is contrastive learning [22], [6], [7], [20], [8]. These methods take advantage of data augmentation to generate augmented versions of one anchor sample, then learn a representation to approximate these augmented samples with similar features in the feature space and put it apart from different samples. Based on the pre-trained networks using this paradigm, the network is fine-tuned for different downstream tasks.

The main contribution of this paper is a contrastive representation learning method for 3D LiDAR point clouds that is able to learn structural information. Our method extracts segments from the LiDAR data and uses contrastive learning to discriminate between similar and dissimilar structures. The learned feature representation is then used as a starting point for supervised fine-tuning, reducing the number of labeled training data needed. Our results suggest that our method can better learn the structural information (see Fig. 1) and a more descriptive feature representation during the self-supervised pre-training, surpassing previous point cloud-based contrastive methods in different evaluations. We explicitly show that, compared to the state of the art, our approach (i) is more efficient when using fewer labels, (ii) can better describe fine-grained structures, and (iii) is more transferable between different datasets.

The implementation of our approach is publicly available at https://github.com/PRBonn/segcontrast.

II. Related Work

Contrastive learning methods are receiving increasing attention in the vision community. Such methods improve the performance of different image-based classification methods using fewer or even no labels [45], [6], [7], [22], [20], [54], [8]. Contrastive learning takes advantage of data augmentation to generate two distinct versions of one anchor sample, creating a positive pair. Then, they train the network to learn a feature representation that maximizes the similarity between this positive pair and minimizes the similarity with other so-called negative samples.

Recent contrastive learning works aimed at fine-grained tasks, such as semantic segmentation [46], or object detection [24]. These methods apply the contrastive loss over image segments extracted using a class-agnostic segmentation approach. Such works improved the network performance using the contrastive loss for pre-training [24], [46] or as an auxiliary supervised loss [48], [37].

Point cloud data has been the source of extensive research in the field of autonomous vehicles [21]. Much effort has been dedicated to applying convolutional operations over point cloud data either by projecting the point cloud to images [34], [32], [49], [12] or by defining 3D convolution operations [38], [40], [19], [10], [44]. Between both approaches, 3D convolutions gained recently more visibility on LiDAR data due to their performance on different tasks, e.g., semantic segmentation [58], [43] and panoptic segmentation [16], [57].

The compelling results of contrastive learning methods on 2D image data drew attention to the application of such techniques on 3D data, especially in autonomous driving, for example, for semantic segmentation [58], [43] and object detection [9], [39] tasks. Xie et al. [50] address the point cloud contrastive learning using a point-wise loss. However, the point-wise contrastive loss depends on a involved pre-processing to generate a map of correspondent points between sequential scans. Hou et al. [26] add more context to the contrastive pre-training by dividing the scene into different spatial partitions, learning both the feature representation and the spatial partitioning of the points. Zhang et al. [55] propose a more global contrastive loss. For each scan a pair of augmented views are generated, i.e., the positive pair. Then, the other augmented scans are taken as the negative samples, computing the contrastive loss over extracted features from the scans. However, their method relies on a multi-branch architecture using two backbones, one for points and one for voxels. Despite promising results, none of the methods focus on point clouds generated by an automotive LiDAR sensor.

Our contribution is a contrastive representation learning method for outdoor LiDAR data used in autonomous driving. We extract class-agnostic segments from the point cloud and propose a contrastive loss to be applied over the extracted segments. Distinct from prior work, our representation learning method learns more contextualized information by discriminating the segmented structures on the point cloud and it learns a more robust and descriptive embedding space.

III. Our Approach

Fig. 2 provides an overview of our approach. We rely on a class-agnostic point cloud segmentation to segment the structures in the scene. Unlike most prior work, our method uses a single backbone and does not require a point-wise mapping. We use a momentum encoder network and a feature bank [22] during training to increase the number of negatives samples and contrast structures segmented over different scans. Then, point-wise features are computed for the whole point cloud and the mapping of the segment is used to extract the segment-wise points and features. We apply dropout [42] and global max pooling for each segment and compute a feature vector using an MLP projection head [6]. After that, we calculate the contrastive loss over the segments and update the feature bank with these segments features. In the next sections, we provide more details on the individual parts of our method.

A. Unsupervised Segment Extraction

Our method relies on segments of different structures in the point cloud. In outdoor LiDAR data, the different objects in a scene are mainly connected by the ground and usually better separated compared to indoor scenes. Given this characteristic, it is comparably easy to extract segments without labels in two steps by removing first the ground and then clustering the remaining points [4], [25], [18].

Given a point cloud $\mathcal{P} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_N\}$ with $|\mathcal{P}| = N$ points $\boldsymbol{p}_i \in \mathbb{R}^3$, we can fit the ground plane and partition the point cloud into ground $\mathcal{G}$ and non-ground points $\mathcal{P}'$, such that $\mathcal{P} = \mathcal{P}' \cup \mathcal{G}$ and $\mathcal{P}' \cap \mathcal{G} = \emptyset$, similarly to prior approaches [4], [25]. Then, using a clustering algorithm, we

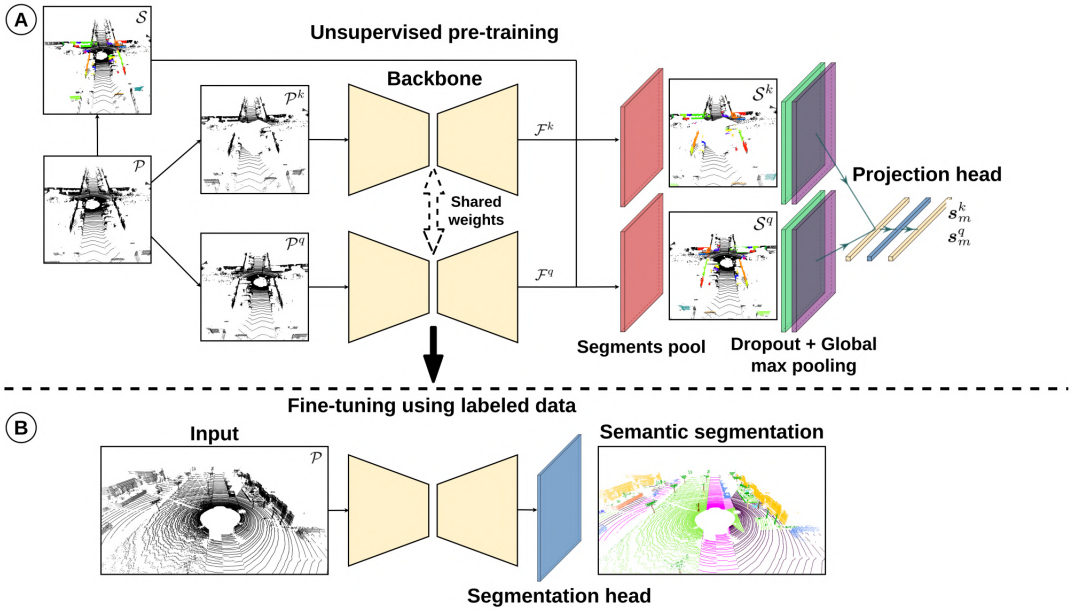


Fig. 2: (A) From a point cloud $\mathcal{P}$, we generate the augmented views $\mathcal{P}^q$ and $\mathcal{P}^k$ by data augmentation and extract class-agnostic segments $\mathcal{S}$ from $\mathcal{P}$. We compute the point-wise features $\mathcal{F}^q$ and $\mathcal{F}^k$ and determine the augmented segments $\mathcal{S}^q$ and $\mathcal{S}^k$ with its point-wise features using the point indexes of $\mathcal{S}$ extracted from $\mathcal{P}$. Then, we apply dropout followed by global max pooling over each segment, and we project the segment feature vectors using the projection head to get the final features $\boldsymbol{s}_m^q$ and $\boldsymbol{s}_m^k$ from the M segments and compute the contrastive loss. (B) After the pre-training, we fine-tune the pre-trained backbone for the downstream task, i.e., semantic segmentation.

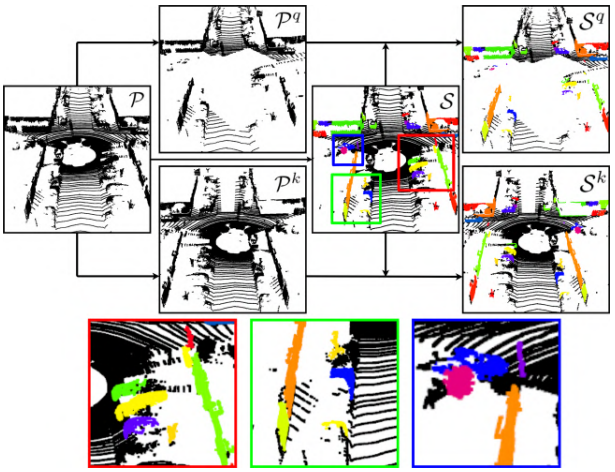


Fig. 3: From $\mathcal{P}$, we extract the segments $\mathcal{S}$ and generate the augmented views $\mathcal{P}^q$ and $\mathcal{P}^k$. With the augmented views and the segments given as point indexes, we can extract the set of augmented segments $\mathcal{S}^q$ and $\mathcal{S}^k$. We highlight some of the structures for a better visualization of the segments (solid colored squares).

can divide $\mathcal{P}'$ into M segments $\mathcal{S}_m$, such that $\mathcal{P}' = \bigcup_{m=1}^{M} \mathcal{S}_m$ and $\bigcap_{m=1}^{M} \mathcal{S}_m = \emptyset$, i.e., the segments are mutually exclusive. Each segment $\mathcal{S}_m$ in this partition represents a different structure from the original point cloud.

More specifically, we use RANSAC [15] to fit the ground plane and define the ground $\mathcal{G}$ and non-ground points $\mathcal{P}'$. Given the fitted plane and a distance threshold α, the inliers (ground) and outliers (non-ground) points are partitioned. We use DBSCAN [14] to cluster the non-ground points $\mathcal{P}'$ and determine the segments $\mathcal{S}_m$.

A common problem of such class-agnostic segmentation is the aspect of over- and under-segmentation [4]. To overcome this and extract representative segments, we define a minimum number of points ε in a cluster to be considered a segment. Moreover, since we will have a different number of segments for every point cloud segmentation, we select the δ segments with the highest number of points to avoid memory overflow during training. The remaining segments are also added to the set of ground points $\mathcal{G}$.

Despite its simplicity, this segmentation method can divide the scene into distinct structures, as illustrated in the Fig. 3. To save computations while training, we cache the segments after the first pass over the point clouds.

B. Segment Augmentation

With each point assigned to a segment, we apply data augmentations to generate the segments augmented pairs $\mathcal{S}^q$ and $\mathcal{S}^k$. We extract random views, $\mathcal{P}^q$ and $\mathcal{P}^k$, by cropping a random cuboid region from the anchor point cloud $\mathcal{P}$ [55]. Then, we apply random augmentations individually over the point clouds $\mathcal{P}^q$ and $\mathcal{P}^k$. We use random rotation around the z axis, random scale, random flip, random cuboid dropout [55], point jittering, and rotation perturbations around x, y and z axes to augment the views. All augmentations are combined and applied once to each augmented view $\mathcal{P}^q$ and $\mathcal{P}^k$.

By extracting a pair of views and applying those augmentations on the point clouds, we indirectly apply augmentations over the extracted segments, see Fig. 3 for an illustration. Since we maintain the point segment assignments via the point indexes during the augmentation, we can easily extract the augmented segments $\mathcal{S}^q$ and $\mathcal{S}^k$ from the augmented views and compute the contrastive loss.

C. Segment Contrastive Loss

The contrastive loss function is designed to discriminate between the positive and negative pairs. We use in this paper the InfoNCE loss [45], together with the momentum encoder and the feature bank proposed by He et al. [22]. For each augmented sample feature vector $\boldsymbol{q}$ and its positive pair $\boldsymbol{k}$ in the batch, the loss is calculated as:

$$\mathcal{L}_q = -\log \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}/\tau\right)}{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}/\tau\right) + \sum_{i=1}^{K} \exp\left(\boldsymbol{q}^\top \boldsymbol{f}_i/\tau\right)}, \quad (1)$$

where τ is a normalization temperature parameter and $\mathcal{Q} = \{\boldsymbol{f_1}, \ldots, \boldsymbol{f_K}\}$ is the feature bank with $|\mathcal{Q}| = K$ implemented as FIFO queue, which are used as negative samples.

In our case, we have two augmented views, $\mathcal{P}^q$ and $\mathcal{P}^k$, from the anchor point cloud $\mathcal{P}$. We compute the point-wise features $\mathcal{F}^q$ and $\mathcal{F}^k$ from both augmented views and extract the augmented segments $\mathcal{S}^q$ and $\mathcal{S}^k$ from them. Then, we pass the segments through the projection head to compute the segment-wise feature vectors $\boldsymbol{s}_m^q$ and $\boldsymbol{s}_m^k$. Therefore, we define our contrastive loss as a segment discrimination:

$$\mathcal{L}_q = -\sum_m \log \frac{\exp\left(\boldsymbol{s}_m^{q\top} \boldsymbol{s}_m^k/\tau\right)}{\exp\left(\boldsymbol{s}_m^{q\top} \boldsymbol{s}_m^k/\tau\right) + \sum_{i=1}^{K} \exp\left(\boldsymbol{s}_m^{q\top} \boldsymbol{f}_i/\tau\right)}, \quad (2)$$

where, as in Eq. (1), τ is a normalization temperature parameter and $\boldsymbol{f}_k$ are the features from the feature bank as before. Note that the number of segments M may vary for different point clouds but is the same for the positive pairs $\mathcal{S}^q$ and $\mathcal{S}^k$.

At the end of each iteration, the feature bank is updated with the segment features from the current batch, maintaining only the last K segments seen by the network.

D. Pre-training Pipeline

Given one input point cloud $\mathcal{P}$ and its augmented pair $\mathcal{P}^q$ and $\mathcal{P}^k$, we use the backbone to compute the point-wise features $\mathcal{F}^q$ and $\mathcal{F}^k$. We use the entire point cloud during the backbone forward pass to learn the relationship between the segments and the scene. Then, we extract the augmented segments $\mathcal{S}^q$ and $\mathcal{S}^k$ from the point cloud with its point-wise features. Next, we apply dropout and global max-pooling over each segment to compute a feature vector. This feature vectors are then passed through the projection head to get $\boldsymbol{s}_m^q$ and $\boldsymbol{s}_m^k$ and calculate the contrastive loss.

IV. Implementation and Experimental Setup

Our experimental setup follows the usual evaluations on contrastive learning methods. First, we pre-train the backbone using our contrastive method. Then, we fine-tune it on different setups for a more in-depth ablation. We used the SemanticKITTI [3], [2], [17] and SemanticPOSS [36] datasets for the self-supervised pre-training, both collected in an outdoor urban environment. We compare our approach to PointContrast [50] and DepthContrast [55], using their official implementations for pre-training and data pre-processing. We use MinkUNet [10] as the backbone, which employs sparse convolutions for 3D processing.

For the class-agnostic segmentation, see Sec. III-A, we set $\alpha = 0.25\,\text{cm}$, $\varepsilon = 20$ and $\delta = 50$, which are the RANSAC distance threshold, the minimum number of points per segment and the maximum number of segments extracted per point cloud, respectively.

For our pre-training, we use the stochastic gradient descent (SGD) optimizer with a momentum of 0.9 and set the learning rate to 0.12 and the weight decay to 0.0004. We use a cosine annealing learning rate schedule [31] with a minimum learning rate of 0.00012, following the pre-training scheme used by Zhang et al. [55]. For all the methods, we randomly sample $20,000$ points from the entire point cloud after data augmentation to limit the number of points per sample, and we pre-train the backbone for 200 epochs. Given the size of SemanticKITTI, we use $K = 8,152$ for the feature bank of the DepthContrast method. For our approach, we set it to $K = 65,536$, with the temperature parameter $\tau = 0.1$. It is important to highlight that we have a bigger feature bank since we save the M segments features extracted from the point cloud instead of the complete point cloud. We use the two linear layers proposed by Chen et al. [6] for our projection head, which is also used in DepthContrast, and we set $p = 0.4$ for the dropout layer. For our method and DepthContrast, we use batch size 8, and for PointContrast, we use batch size 16 to maintain the batch size 4 per GPU as in the original paper. Given the PointContrast method characteristics, it is known that batch size change may affect the approach performance. However, we used the same cluster with four NVIDIA GTX1080TI 12 GB GPUs for all evaluated methods to make the comparisons as fair as possible.

For fine-tuning, we use the same datasets used for pre-training, i.e., SemanticKITTI and SemanticPOSS. For the semantic segmentation experiments, we use SGD with momentum of 0.9, learning rate of 0.24, and weight decay of 0.0004. We also use a cosine annealing scheduler with a minimum learning rate equal to 0.00024, following the semantic segmentation setup used by Tang et al. [43]. We set the batch size to 2 and randomly sample $80,000$ points per point cloud during training. For the object detection experiment, we use PointRCNN [41] as the base detector, and the same 5% label set used by Zhang et al. [55] for a fair comparison. For all the experiments, we use one NVIDIA RTX2080 Super 8 GB GPU. The experimental results are collected in the validation sequences from both datasets, i.e., sequence 8 for SemanticKITTI and sequences 4 and 5 for SemanticPOSS. In both datasets, the validation sequences were not used for pre-training or fine-tuning.

V. Experimental Evaluation

We present our experiments to show the capabilities of our method. We compare our method to the state of the art and show that our approach (i) is more efficient when using fewer labels, (ii) can better describe fine-grained structures and (iii) is more transferable between different datasets.

A. Label Efficiency

The first experiment evaluates the robustness of the features learned from the different representation learning methods by fine-tuning it to the semantic segmentation task using

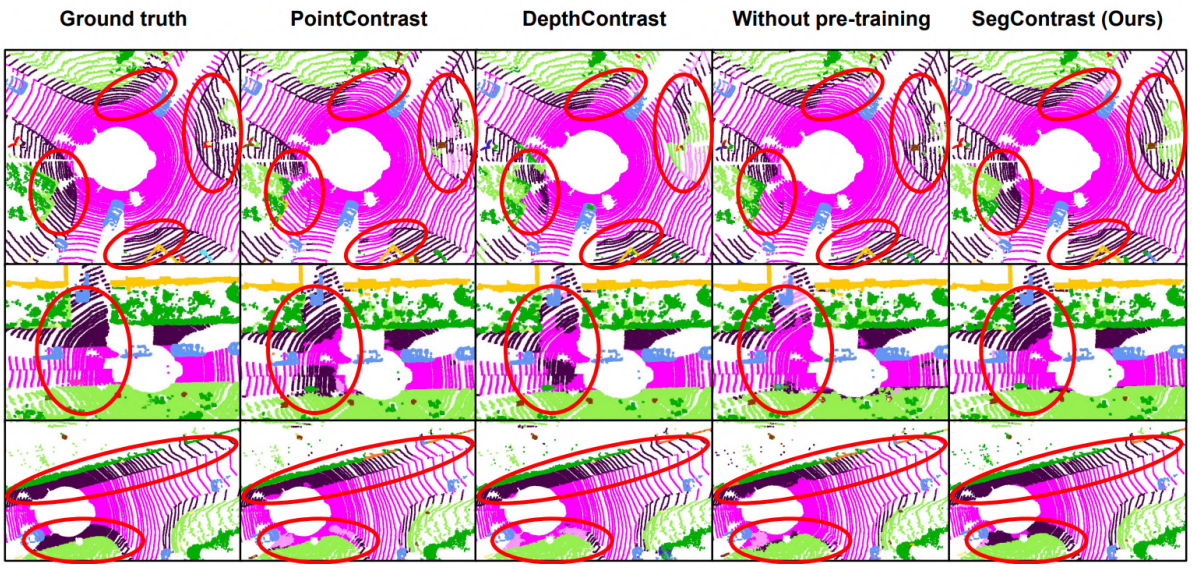


Fig. 4: Qualitative results on three different validation scans (rows). We show results of the networks fine-tuned with only 0.1% of the labels that are pre-trained with different contrastive methods or trained from scratch (without pre-training). We compare the results from PointContrast [50], DepthContrast [55] and our method, SegContrast. With our pre-training, SegContrast, the fine-tunned network can better distinguish the division between different structures, i.e., sidewalk and road, as shown by the highlighted areas (solid red circles).

TABLE I: Number of training epochs used for different label regimes on SemanticKITTI

Label regime	0.1%	1%	10%	50%	100%
Number of epochs	300	120	40	20	15

TABLE II: Fine-tuning at different label regimes on SemanticKITTI for semantic segmentation (mIoU)

	0.1%	1%	10%	50%	100%
Without pre-training	25.59	41.70	53.87	58.34	59.63
PointContrast [50]	28.52	43.40	53.79	57.30	59.77
DepthContrast [55]	33.51	46.41	**56.29**	**58.54**	59.88
SegContrast (Ours)	**34.78**	**47.41**	55.21	58.33	**60.53**

the SemanticKITTI dataset. Since it is a larger dataset, we can divide it into different label percentage regimes and increasingly compare it to support our first claim. We define five different label percentage regimes, i.e., using 0.1%, 1%, 10%, 50%, and 100% of the labeled training data, where we select a fixed subset of scans from the dataset given the regime percentage. Every subset is chosen from the entire dataset, such that all the classes are present. When training with fewer labels, the total number of training iterations will decrease accordingly. Therefore, we increase the number of epochs as we reduce the number of training scans to achieve convergence at every label regime (see Tab. I).

Fig. 4 gives a qualitative comparison between the different contrastive methods and the network without pre-training when training with 0.1% of the labels. From the top views, we observe that the network trained from scratch could not learn much structural information, leading to a noisy division between different structures. This same noisy division can be seen in the other contrastive methods. Our approach better learns the structural information of the point clouds, which leads to better boundaries between different structures in the scene. This improvement can also be seen in more fine-grained classes, e.g., pole and traffic signs, shown in Fig. 1.

Tab. II shows the results on the different label regimes. All methods perform better than training from scratch, i.e., without pre-training, and the gap between the results diminishes as the number of scans used for training increases. Our method is better at lower label regimes. As the amount of labels increases, DepthContrast achieves a comparable performance. At the full label training, all the methods converge to a similar result as training from scratch. This is expected since the data used for pre-training, and fine-tuning are the same. Moreover, Tab. III presents the per-class IoU at the lowest label regime. It is possible to see that our method performs better in the per-class IoU, outperforming previous approaches in the majority of the classes. This evaluation shows that our approach performs better when using fewer labels compared to the other contrastive learning methods, being able to better describe the different structures in the point cloud.

For a more complete evaluation, we also compare the methods fine-tuning to object detection. Tab. IV presents our results on the object detection task on the KITTI dataset [17]. In this experiment, we use the same 5% labels set used by Zhang et al. [55]. With 100% of labels, the comparison between the network without pre-training and the pre-trained models shows no significant difference. Since we use KITTI for pre-training and fine-tuning, this is expected since no new data is seen during pre-training. With 5% of labels, it is possible to show the gain of the pre-trained network. Except for the pedestrian class, all the methods outperform the network without pre-training by a large margin. In the car and cyclist classes, our method surpasses previous methods in almost all the difficulties. These results indicate that our approach can learn a robust feature representation, outperforming previous methods on different tasks when using fewer labels.

TABLE III: Per-class IoU fine-tuning with 0.1% labels

Method	mIoU	road	sidewalk	parking	building	car	vegetation	trunk	terrain	fence	pole	traffic-sign
Without pre-training	25.59	72.46	46.53	7.51	78.43	78.79	82.15	26.01	63.48	17.68	9.02	1.29
PointContrast [50]	28.52	72.71	50.24	9.06	80.20	85.78	83.29	39.29	68.18	19.56	19.68	7.61
DepthContrast [55]	33.51	79.13	58.85	12.91	**82.41**	**89.18**	**84.82**	45.07	**71.52**	21.34	50.80	17.30
SegContrast (Ours)	**34.78**	**80.32**	**61.53**	**14.70**	81.99	89.05	84.18	**49.34**	70.25	**21.53**	**50.89**	**29.93**

TABLE IV: Fine-tuning with 5% and 100% of labels on KITTI for object detection (mAP with 40 recall positions)

	Without pre-training			PointContrast [50]			DepthContrast [55]			SegContrast (ours)		
	100% labels											
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Car	91.46	81.77	**79.92**	91.44	**81.83**	79.85	91.79	81.48	79.90	**91.97**	80.17	79.79
Pedestrian	68.29	**63.76**	58.07	**69.81**	63.56	**58.13**	66.39	60.44	55.41	67.46	62.71	56.42
Cyclist	90.65	73.35	70.05	90.48	73.87	69.81	**92.45**	74.69	**71.18**	90.71	**75.38**	70.79
	5% labels											
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Car	79.33	63.83	58.92	80.83	66.52	61.70	82.32	67.61	62.59	**82.62**	**68.27**	**64.41**
Pedestrian	69.34	61.72	54.49	69.75	**62.77**	**55.71**	**69.81**	62.37	55.13	69.10	61.59	54.49
Cyclist	84.48	61.82	57.80	**88.82**	67.53	63.17	87.52	66.91	62.50	88.70	**68.48**	**64.35**

TABLE V: Linear evaluation mIoU and per-class IoU (%) pre-trained and evaluated on SemanticKITTI

Method	mIoU	road	sidewalk	parking	building	car	vegetation	trunk	terrain	fence	pole	traffic-sign
Without pre-training	4.29	30.61	2.72	0.0	5.56	0.02	42.39	0.0	0.05	0.24	0.0	0.0
PointContrast [50]	**24.02**	**70.61**	**37.00**	0.0	**83.37**	**88.93**	**75.10**	33.26	**53.93**	8.66	5.55	0.0
DepthContrast [55]	15.91	40.68	5.35	0.0	54.81	59.43	65.30	24.55	17.56	13.50	20.47	0.0
SegContrast (Ours)	23.36	59.02	30.83	**0.5**	71.68	80.23	73.13	**34.56**	37.51	**15.68**	**40.59**	**0.19**

TABLE VI: Pre-training with SemanticKITTI and fine-tuning on SemanticPOSS with different label regimes (mIoU)

	0.1%	1%	10%	50%	100%
Without pre-training	33.09	43.14	57.27	63.34	64.22
Supervised pre-training	42.88	54.40	60.22	64.26	64.54
PointContrast [50]	31.78	49.06	56.49	62.93	64.30
DepthContrast [55]	41.94	52.66	59.27	64.09	64.65
SegContrast (Ours)	**43.69**	**55.21**	**60.33**	**64.58**	**64.86**

B. Linear Evaluation

A typical experiment used for image-based contrastive methods is the so-called linear evaluation. This evaluation freezes the pre-trained backbone and trains only a linear layer on top of it to compare how well the feature representation can describe the different classes even without fine-tuning the whole network. Our experiment follows the same setup, the pre-trained backbone weights are frozen, and we train only a linear segmentation head. The result of this experiment supports our claim that our method can learn a feature representation that better describe fine-grained structures.

Tab. V displays the results of the linear evaluation over the different self-supervised contrastive methods and over the randomly initialized network without pre-training for a lower bound on the performance. DepthContrast shows the worst performance in this evaluation, which indicates that the method cannot learn a feature representation as descriptive as the other methods. PointContrast shows a better performance, since the method uses a point-wise contrastive loss. Furthermore, when looking at the underrepresented classes, e.g., parking, trunk, fence, pole or traffic sign, our method outperforms the other methods. Thus, PointContrast achieves a higher mIoU by learning the more represented classes, e.g., road, building. In contrast, our method seems better suited to represent the fine-grained classes.

C. Feature Representation Transferability

In this third experiment, we evaluate the transferability of our learned feature representation. The results support our claim that our method is more transferable between different datasets. We use SemanticKITTI for unsupervised pre-training of the backbone with different contrastive methods or use the commonly used supervised pre-training. Then, we fine-tune the differently pre-trained networks on the SemanticPOSS dataset and compare their performance. SemanticPOSS is smaller than SemanticKITTI, which gives a setup aligned to standard image-based setting, where a larger dataset, e.g., ImageNet [13], is used for pre-training and then fine-tuning is performed on smaller datasets.

Fig. 5 displays the network performance in different training epochs during fine-tuning. Here, we see a comparable performance of the unsupervised and supervised pre-training. As shown, after only one training epoch, our method shows a considerably better performance than the other contrastive methods and achieves the same performance as the supervised pre-training. Even though our approach does not use any labels, our method learns a feature representation comparable with the supervised pre-training on SemanticKITTI. This result suggests that our method can learn a more general feature representation than previous methods and it seems to be more suitable for fine-tuning on a different dataset.

TABLE VII: Linear evaluation on SemanticPOSS with pre-training on SemanticKITTI (mIoU)

Method	mIoU (%)	accuracy (%)
Without pre-training	6.32	42.46
PointContrast [50]	24.79	72.55
DepthContrast [55]	16.41	63.09
SegContrast (Ours)	**31.51**	**73.51**

TABLE VIII: Fine-tuning to SemanticPOSS with pre-training on SemanticKITTI and SemanticPOSS using our method (mIoU)

Dataset	Linear evaluation	Fine-tune
Without pre-training	6.32	64.22
SemanticPOSS	29.68	64.31
SemanticKITTI	**31.51**	**64.86**

Tab. VI displays the results of the different contrastive pre-training methods and the supervised pre-training on SemanticKITTI. The pre-training methods improve the network performance at the lower label regimes, but the previous contrastive methods cannot surpass the supervised pre-training. However, our method outperforms both self-supervised and supervised pre-training at all different label regimes. This experiment indicates the robustness of the learned feature representation when fine-tuning to a different LiDAR data, exceeding even the supervised pre-training.

In Tab. VII, we show the linear evaluation over the SemanticPOSS with the network pre-trained on SemanticKITTI. DepthContrast shows the lowest performance in this evaluation, showing that the feature representation is less descriptive when transferring to a different dataset compared to the other methods. Our method surpasses the other approaches, outperforming them by a large margin. These results suggest that our approach can learn a point cloud representation transferable across different datasets and LiDAR sensors.

Finally, we evaluate the self-supervised pre-training on SemanticKITTI and SemanticPOSS, fine-tuning it on SemanticPOSS. In Tab. VIII, we show both the linear evaluation and the fine-tuning. As we can see, the performance of the pre-training on SemanticKITTI is better on both experiments. We can see that we obtain a performance gain using a large dataset for self-supervised pre-training. This also highlights the generalization of the feature representation achieved by our method. The pre-training on SemanticKITTI performed better than on SemanticPOSS, even though both datasets were collected with a different LiDAR sensors and different sensor mounting positions.

VI. Conclusion

In this paper, we present a novel representation learning approach for LiDAR point clouds in outdoor environments. Our approach exploits the characteristics of outdoor LiDAR data to extract class-agnostic segments and applies the contrastive loss over these segments. We evaluate our strategy on different datasets and provide comparisons with other state-of-the-art feature representation learning approaches. The experiments suggest that our approach can learn a more robust feature representation than previous works outperforming them on different downstream tasks. Furthermore, our self-supervised feature representation seems to be more transferable when fine-tuning on a different target dataset outperforming even the supervised pre-training.

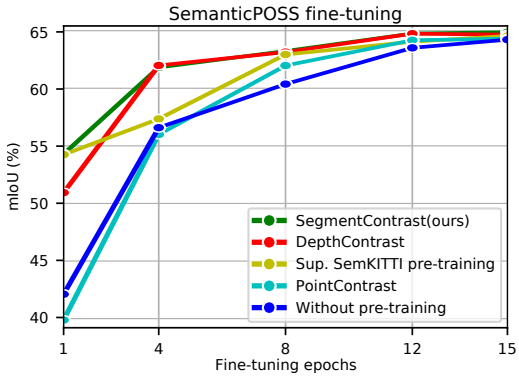


Fig. 5: Comparison between contrastive pre-trained networks fine-tuned to the SemanticPOSS dataset with the network trained from scratch. At the beginning of training, our method shows a comparable performance to supervised pre-training on SemanticKITTI, evidencing that our learned feature representation is as robust as the supervised pre-training.

References

[1] I. Achituve, H. Maron, and G. Chechik. Self-supervised learning for domain adaptation on point clouds. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2021.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, J. Gall, and C. Stachniss. Towards 3d lidar-based semantic scene understanding of 3d point cloud sequences: The semantickitti dataset. *Intl. Journal of Robotics Research (IJRR)*, 40(8-9):959–967, 2021.

[3] 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 Intl. Conf. on Computer Vision (ICCV)*, 2019.

[4] J. Behley, V. Steinhage, and A. Cremers. Laser-based Segment Classification Using a Mixture of Bag-of-Words. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[5] Q. Chen, Y. Wang, T. Yang, X. Zhang, J. Cheng, and J. Sun. You Only Look One-Level Feature. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2020.

[7] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G.E. Hinton. Big Self-Supervised Models are Strong Semi-Supervised Learners. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[8] X. Chen and K. He. Exploring Simple Siamese Representation Learning. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[9] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-View 3D Object Detection Network for Autonomous Driving. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] C. Choy, J. Gwak, and S. Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[11] M. Cordts, S.M. Omran, Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] T. Cortinhal, G. Tzelepis, and E. Aksoy. SalsaNext: Fast Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving. *arXiv preprint*, 2003.03653v1, 2020.

[13] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, June 2009.

[14] M. Ester, H. Kriegel, J. Sander, and X.Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the Conf. on Knowledge Discovery and Data Mining (KDD)*, 1996.

[15] M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, 1981.

[16] S. Gasperini, M.N. mahani, A. Marcos-Ramiro, N. Navab, and F. Tombari. Panoster: End-To-End Panoptic Segmentation of LiDAR Point Clouds. *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[17] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[18] Z. Gojcic, O. Litany, A. Wieser, L.J. Guibas, and T. Birdal. Weakly Supervised Learning of Rigid 3D Scene Flow. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[19] B. Graham and L. van der Maaten. Submanifold Sparse Convolutional Networks. *arXiv preprint*, 2017.

[20] J.B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[21] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 2020.

[22] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] O.J. Hénaff, S. Koppula, J.B. Alayrac, A. van den Oord, O. Vinyals, and J. Carreira. Efficient Visual Pretraining with Contrastive Detection. *arXiv preprint*, 2021.

[25] M. Himmelsbach, F.v. Hundelshausen, and H.J. Wuensche. Fast Segmentation of 3D Point Clouds for Ground Vehicles. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2010.

[26] J. Hou, B. Graham, M. Niessner, and S. Xie. Exploring Data-Efficient 3D Scene Understanding With Contrastive Scene Contexts. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[27] Q. Hu, B. Yang, G. Fang, Y. Guo, A. Leonardis, N. Trigoni, and A. Markham. SQN: Weakly-Supervised Semantic Segmentation of Large-Scale 3D Point Clouds with 1000x Fewer Labels. *arXiv preprint*, abs/2104.04891, 2021.

[28] P. Jiang and S. Saripalli. LiDARNet: A Boundary-Aware Domain Adaptation Model for Point Cloud Semantic Segmentation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[29] F. Langer, A. Milioto, A. Haag, J. Behley, and C. Stachniss. Domain Transfer for Semantic Segmentation of LiDAR Data using Deep Neural Networks. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[30] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick. Microsoft COCO: Common Objects in Context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 740–755, 2014.

[31] I. Loshchilov and F. Hutter. SGDR: Stochastic Gradient Descent with Restarts. *arXiv preprint*, abs/1608.03983, 2016.

[32] A. Milioto, J. Behley, C. McCool, and C. Stachniss. LiDAR Panoptic Segmentation for Autonomous Driving. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[33] A. Milioto, L. Mandtler, and C. Stachniss. Fast Instance and Semantic Segmentation Exploiting Local Connectivity, Metric Learning, and One-Shot Detection for Robotics. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[34] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[35] A. Nekrasov, J. Schult, O. Litany, B. Leibe, and F. Engelmann. Mix3D: Out-of-Context Data Augmentation for 3D Scenes. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2021.

[36] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, and H. Zhao. SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances. *arXiv preprint:2002.09147*, 2020.

[37] T. Park, A.A. Efros, R. Zhang, and J.Y. Zhu. Contrastive Learning for Unpaired Image-to-Image Translation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[38] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[39] C.R. Qi, O. Litany, K. He, and L.J. Guibas. Deep Hough Voting for 3D Object Detection in Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[40] C. Qi, K. Yi, H. Su, and L.J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[41] S. Shi, X. Wang, and H. Li. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal on Machine Learning Research (JMLR)*, 15:1929–1958, 2014.

[43] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han. Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[44] H. Thomas, C. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[45] A. van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv preprint*, 2018.

[46] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, and L. Van Gool. Unsupervised Semantic Segmentation by Contrasting Object Mask Proposals. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[47] H. Wang, Y. Cong, O. Litany, Y. Gao, and L.J. Guibas. 3DIoUMatch: Leveraging IoU Prediction for Semi-Supervised 3D Object Detection. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[48] W. Wang, T. Zhou, F. Yu, J. Dai, E. Konukoglu, and L. Gool. Exploring Cross-Image Pixel Contrast for Semantic Segmentation. *arXiv preprint*, 2021.

[49] 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 *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[50] S. Xie, J. Gu, D. Guo, C.R. Qi, L. Guibas, and O. Litany. PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[51] X. Xu and G.H. Lee. Weakly Supervised Semantic Point Cloud Segmentation: Towards 10x Fewer Labels. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[52] L. Yi, B. Gong, and T. Funkhouser. Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[53] X. Yuan, A. Kortylewski, Y. Sun, and A. Yuille. Robust Instance Segmentation Through Reasoning About Multi-Object Occlusion. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[54] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. *arXiv preprint*, abs/2103.03230, 2021.

[55] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra. Self-Supervised Pretraining of 3D Features on any Point-Cloud. *arXiv preprint*, 2021.

[56] L. Zheng, M. Tang, Y. Chen, G. Zhu, J. Wang, and H. Lu. Improving Multiple Object Tracking With Single Object Tracking. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[57] Z. Zhou, Y. Zhang, and H. Foroosh. Panoptic-PolarNet: Proposal-Free LiDAR Point Cloud Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[58] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin. Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[C] J. Behley, M. Garbade, A. Milioto, *et al.*, "Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset," *Intl. Journal of Robotics Research (IJRR)*, vol. 40, no. 8-9, pp. 959–967, 2021. DOI: 10.1177/02783649211006735

Check for updates

Article

The International Journal of Robotics Research
2021, Vol. 40(8-9) 959–967
© The Author(s) 2021
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/02783649211006735
journals.sagepub.com/home/ijr
SAGE

Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset

Jens Behley[1], Martin Garbade[2], Andres Milioto[1], Jan Quenzel[3], Sven Behnke[3], Jürgen Gall[2] and Cyrill Stachniss[1]

Abstract

A holistic semantic scene understanding exploiting all available sensor modalities is a core capability to master self-driving in complex everyday traffic. To this end, we present the SemanticKITTI dataset that provides point-wise semantic annotations of Velodyne HDL-64E point clouds of the KITTI Odometry Benchmark. Together with the data, we also published three benchmark tasks for semantic scene understanding covering different aspects of semantic scene understanding: (1) semantic segmentation for point-wise classification using single or multiple point clouds as input; (2) semantic scene completion for predictive reasoning on the semantics and occluded regions; and (3) panoptic segmentation combining point-wise classification and assigning individual instance identities to separate objects of the same class. In this article, we provide details on our dataset showing an unprecedented number of fully annotated point cloud sequences, more information on our labeling process to efficiently annotate such a vast amount of point clouds, and lessons learned in this process. The dataset and resources are available at http://www.semantic-kitti.org.

Keywords

Dataset, LiDAR, point clouds, semantic segmentation, panoptic segmentation, semantic scene completion

1. Introduction

Since autonomous vehicles successfully completed the driving tasks at the DARPA Urban Challenge (Montemerlo et al., 2008; Urmson et al., 2008), the prospect of fully autonomous cars led to founding of many startups pursuing the endeavor of creating always attentive robotic cars that prevent and avoid traffic fatalities occurring today due to human error. After the Urban Challenge, many expected and predicted that cars capable of driving fully autonomously in dense urban traffic would be already reality by today. However, solving the driving task in real-world environments with many non-compliant traffic participants that might violate rules and with the complexity of sometimes contradictory signage is a stupendous endeavor.

Perception is a centerpiece of every intelligent robotic system to handle the complexities of operating in semi-structured and natural environments. In particular, self-driving cars rely on robust and accurate perception systems that allow them to perform safe and efficient driving maneuvers. For a holistic semantic scene understanding, they need to perceive obstacles, identify other traffic participants, but also reason about functional street surface types, e.g., parking areas, lanes, and sidewalks.

Recent progress in perception using images, but also LiDAR sensors, is driven by advances in deep learning enabling end-to-end trainable perception systems without the need to hand-craft features (LeCun et al., 2015). Training deep neural networks with millions of parameters was mainly enabled by two essential developments: (1) the possibility to repurpose graphics processing units (GPUs) from producing pixels on a computer screen to compute matrix products in a highly parallel fashion; and (2) the availability of large-scale labeled datasets, such as ImageNet (Deng et al., 2009), that enabled training networks without over-fitting to the training data.

Real-world datasets play an important role in the aforementioned endeavor to attain fully autonomous cars, as they provide realistic data on the one hand, but also allow

[1]Photogrammetry and Robotics Lab, University of Bonn, Germany
[2]Computer Vision Group, University of Bonn, Germany
[3]Autonomous Intelligent Systems, University of Bonn, Germany

Corresponding author:
Jens Behley, Photogrammetry and Robotics Lab, University of Bonn, Nussallee 15, 53155 Bonn, Germany.
Email: jens.behley@igg.uni-bonn.de

us to measure progress towards our goal on the other hand. Datasets, such as Cityscapes (Cordts et al., 2016) and Mapillary Vistas (Neuhold et al., 2017), enable fine-grained perception tasks to be investigated, such as semantic segmentation (Everingham et al., 2010) providing classes for each pixel, but also panoptic segmentation (Kirillov et al., 2019), which additionally distinguishes between individual instances of the same class. More specifically, semantic segmentation distinguishes between classes but assigns different objects the same label, e.g., different cars cannot be distinguished. Panoptic segmentation differentiates additionally between objects leading to clear object boundaries or instances. However, panoptic segmentation requires instance IDs only for so-called thing classes, which have clear boundaries, such as cars, pedestrians, and bicyclists. The remaining classes are called stuff classes and do not get an instance ID assigned, such as vegetation, road, or sidewalks.

We present a dataset based on the KITTI Vision Benchmark (Geiger et al., 2013, 2012) that enables the investigation of semantic segmentation and panoptic segmentation using point clouds from an automotive LiDAR sensor. To this end, we annotated all 22 sequences of odometry evaluation of the KITTI Vision Benchmark (Geiger et al., 2013, 2012) consisting of over 43,000 scans using 28 classes. We labeled each point of the point cloud such that corresponding instances of object classes get temporally consistent instance annotations. In addition, we use the annotated sequential data and accurate poses to generate a real-world dataset for semantic scene completion, where an algorithm needs to provide class labels for voxels, but also predict the completed scene which is not visible in the given input voxelized scene. As commonly done with other datasets (Cordts et al., 2016; Lin et al., 2014; Neuhold et al., 2017), the test set labels are not published to ensure an unbiased and fair evaluation. We use CodaLab Competitions (see https://competitions.codalab.org/ for more information) that provides a platform to upload results for a hidden test set, which are then evaluated using a custom evaluation script on cloud-based evaluation servers without revealing the testset annotations.

This article complements our other papers (Behley et al., 2019, 2020), because it focuses on the dataset itself. We provide more details on the annotation process and statistics about the automatic instance extraction process. We, furthermore, discuss lessons learned in the process of annotating a large-scale dataset and maintaining an online evaluation with a hidden test set. Thus, the article provides additional information, compared with the more task-oriented papers, and we refer to these for more details on the tasks, baselines, and evaluation metrics for semantic segmentation and scene completion (Behley et al., 2019) and panoptic segmentation (Behley et al., 2020).

2. Related work

In the following, we focus on datasets providing LiDAR point clouds and, in particular, on datasets that provide annotations for perception tasks, such as object detection, semantic segmentation, or panoptic segmentation.

The seminal KITTI Vision Benchmark (Geiger et al., 2012) aimed at providing a collection of different benchmark tasks to evaluate perception algorithms for autonomous driving. It made a vast collection of data (Geiger et al., 2013) available that was recorded with a sensor suite commonly used by self-driving cars including a stereo camera, a Velodyne HDL-64E LiDAR, and an inertial navigation system (INS) to generate ground-truth data for pose information. The provided benchmarks propelled research in the areas of motion estimation and traffic scene perception and enabled reproducible experiments with standardized metrics.

Since then, only a few LiDAR datasets have been recently published that are recorded either with a terrestrial laser scanner (TLS), such as the Semantic3d dataset (Hackel et al., 2017), or using automotive LiDARs, such as the Paris-Lille-3D dataset (Roynard et al., 2018).

Recently, several major self-driving car ventures released datasets providing camera images in addition to LiDAR point clouds, including Waymo (Sun et al., 2020), Lyft (Kesten et al., 2019), Audi (Geyer et al., 2020), Argo (Chang et al., 2019), Honda (Patil et al., 2019), and Motional (Caesar et al., 2020). Although all these datasets provide instance annotations using bounding boxes, only a few datasets provide point-wise semantic annotation (Geyer et al., 2020; PandaSet, 2020).

The A2D2 dataset (Geyer et al., 2020) provides annotations for semantic segmentation of images that can be used to obtain point-wise labels by projecting LiDAR points into the images and using the associated semantic class from the pixel-level annotation. However, this projection will never cover all LiDAR points owing to the sensor placement and the resulting different view point. The very recently published PandaSet (PandaSet, 2020) provides point-wise annotations of LiDAR point clouds with 42 classes focusing on objects on the road, such as traffic participants, barriers, and cones, and more fine-grained distinction between different vehicle types compared with our annotation. SemanticPOSS (Pan et al., 2020) also provides semantic annotation of point clouds with a focus on scenes with pedestrians captured in a campus environment. The classes are compatible with our classes and the authors provided labels in the same format as our annotation data. Pan et al. (2020) used our annotation tool presented in Section 3.1, but used tracking information to extract instances. NuScenes (Caesar et al., 2020) also recently added annotations for LiDAR point clouds with more diverse categories for different traffic participants. Together with the bounding box annotations, this dataset can also be used for panoptic segmentation. Owing to the large number of different scenes, it provides a highly diverse set of situations.

Table 1 provides an overview of the aforementioned datasets and their characteristics. Other automotive datasets might provide more diversity in terms of cities or number of different scenes. However, our dataset is the only dataset

Table 1. Overview of other point cloud datasets with bounding box (top) and semantic annotations (bottom).

Name	#Scans[1]	#Boxes[2]	#Classes[3]	Data[4]	FoV[5]	Sequential	Reference
KITTI (Detection)	7k/7k	1k	3(3)	B	F	×	(Geiger et al., 2012)
Argoverse	22k	993k	17	B	C	✓	(Chang et al., 2019)
Lyft	46k	1.3M	9	B	C	✓	(Kesten et al., 2019)
CADC	7k	305k	10	B	C	✓	(Pitropov et al., 2021)
nuScenes	44k	1.4M	10 (23)	B	C	✓	(Caesar et al., 2020)
Waymo	200k	12M	4	B	C	✓	(Sun et al., 2020)
A2D2	12k	12k	14	B	F	×	(Geyer et al., 2020)
H3D	27k	1.1M	8	B	F	×	(Patil et al., 2019)
PandaSet	16k	1.4M	12	B	C	✓	(PandaSet, 2020)
SemanticKITTI	23k/20k	682k	8	P	C	✓	—
Name	#Scans[1]	#Points[6]	#Classes[3]	Data[4]	FoV[5]	Sequential	Reference
Oakland3d	17	1.6M	5 (44)	P	C	×	(Munoz et al., 2009)
Freiburg	77	1.1M	4 (11)	P	C	×	(Behley et al., 2012)
Wachtberg	5	400k	5 (5)	P	C	×	(Behley et al., 2012)
Semantic3d	15/15	4.009B	8 (8)	P	C	×	(Hackel et al., 2017)
Paris-Lille-3D	3	143M	9 (50)	P	C	×	(Roynard et al., 2018)
Zhang et al.	140/112	32M	10 (10)	P	F	×	(Zhang et al., 2015)
SemanticPOSS	2k	216M	14	P	C	×	(Pan et al., 2020)
A2D2	31k	930k	38	P†	F	×	(Geyer et al., 2020)
PandaSet	16k	1.388B	42	P	C	✓	(PandaSet, 2020)
nuScenes-lidarseg	34k/6k	1.4B	16 (32)	P	C	✓+	(Caesar et al., 2020)
SemanticKITTI	23k/20k	4.549B	25 (28)	P	C	✓	—

[1]Number of scans for train and test set (k, thousand).
[2]Number of boxes (k, thousand; M, million).
[3]Number of classes used for evaluation and number of classes annotated in brackets.
[4]Type of annotations, where B and P correspond to bounding boxes (B) and point-wise (P).
[5]Field of view of LiDAR sensor with annotations, where F denotes frontal and C denotes complete 360°.
[6]Number of points (k, thousand; M, million; B, billion).
†Point-wise annotations via projection to annotated image and using corresponding image label.
+Frames labeled at 2 Hz.

that combines point-wise semantic annotations directly made in sequences of three-dimensional point clouds with temporally consistent instance annotations for both non-moving and moving traffic participants.

3. Dataset

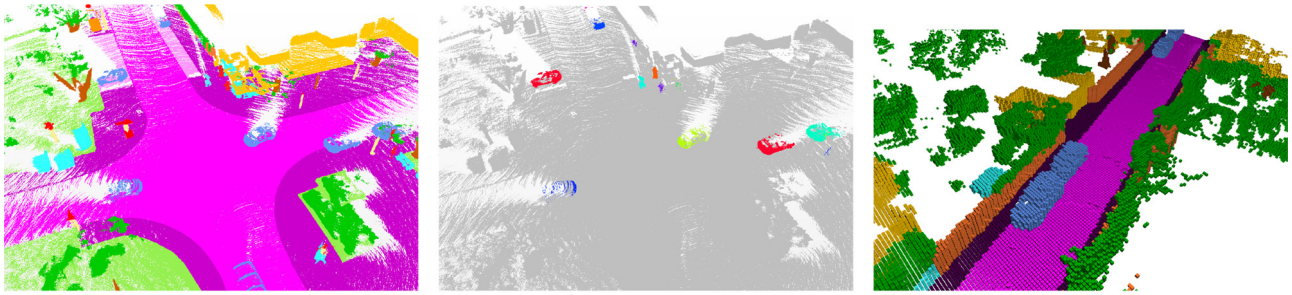
Our dataset provides point-wise semantic annotations for the odometry sequences of the KITTI Vision Benchmark Suite (Geiger et al., 2013), which was the first large-scale dataset providing data recorded with a platform equipped with sensors commonly used on self-driving cars since the DARPA Urban Challenge (Montemerlo et al., 2008). The recording vehicle was equipped with a stereo camera covering the frontal field of view and a rotating 3D LiDAR sensor, the Velodyne HDL-64E S2, covering the full 360° field of view. Both modalities are synchronized such that the cameras are triggered when the spinning LiDAR sensor faces in forward direction (Geiger et al., 2013). The vehicle is additionally equipped with an INS integrating an automotive-grade inertial measurement unit (IMU) with GPS providing position measurements.

In the case of the odometry benchmark,[1] we use the data provided that uses point clouds of a single turn of the LiDAR sensor which are compensated for sensor motion, i.e., individual points in the point cloud are transformed to account for the movement of the sensor during a single turn of the rotating LiDAR sensor.

The odometry dataset comprises 22 sequences: 11 training sequences (sequences 00–10) with ground-truth poses and 11 testing sequences (sequences 11–21) without pose information for which an odometry approach should estimate the poses. The pose error is only locally evaluated and therefore the provided poses are not closed loop or optimized to give globally consistent poses, such that already visited areas would be consistently mapped and old and new point cloud observations of the same place would be aligned properly.

3.1. Point cloud annotation

Our primary objective is to generate a consistent, accurate labeling of the sequential point clouds. It is essential to have consistent and closed-loop poses to facilitate the consistent annotation by accumulating point clouds from multiple scans. Figure 1 shows some qualitative examples of the provided annotations and the achieved fidelity of the point cloud annotations.



(a) (b) (c)

Fig. 1. Qualitative examples of the provided annotations from sequence 08. In (a) we show the semantic annotation and (b) shows the corresponding instance annotation of 50 aggregated scans. We also provide input (not shown) and target voxel grids aggregated from multiple scans for the semantic scene completion task as shown in (c).

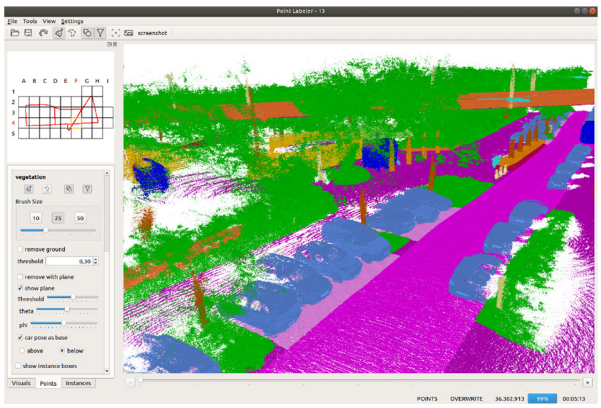


Fig. 2. Our point cloud labeling application for sequential point clouds that we provide together with the dataset.

3.1.1. Pre-processing. To estimate globally consistent poses, we employed our surfel-based simultaneous localization and mapping (SLAM) approach (Behley and Stachniss, 2018). Our mapping approach finds heuristically loop closures using a map-based criterion and performs then pose graph optimization using loop closure constraints to obtain globally consistent poses. For sequences 02 and 07, where the automatic loop closure detection missed loop closures, we manually inserted loop closure constraints to ensure consistent mapping results.

3.1.2. Point cloud annotation. Using the estimated poses, we split the complete trajectory of a sequence into tiles of a given size (we used 100×100 m^2), where we always show an overlap of 15 m with neighboring tiles to ensure consistent labeling at tile boundaries. We collect all points overlapping with a tile and its boundary, which allows us to consistently label points. The tiling is needed to enable consistent labeling of the overlapping point clouds and still allow to visualize all points inside a tile.

Our point cloud labeling graphical user interface provides different filtering methods to facilitate labeling, such as hiding points from a specific class or above/below an adjustable plane. The filtering allows us to accurately label points even in complex situations with overhanging vegetation that cover the view onto parts of the point cloud. Figure 2 shows our labeling application with a challenging situation, where we have a bridge over a street and foliage from overhanging trees.

We provide two tools for annotating the point cloud: (1) a brush, which labels all points in a circular region around the current mouse position and (2) a polygon, which labels all points inside the polygon area specified by setting the polygon corners. Both operations happen on the projected points, where we check the projected three-dimensional point in the image plane for inclusion in the brushed or selected area. Thus, labeling of the point cloud is viewpoint dependent and requires a viewpoint to be found that does not affect points that lie in the line of sight. Here, the aforementioned filtering tools allow such points or simply all already annotated points to be filtered.

3.1.3. Class annotation. Following best practices for dataset labeling, we compiled a labeling instruction based on Mapillary's instructions (Neuhold et al., 2017) and provided instruction videos on how to label certain objects such as cars and bicycles standing near a wall to our annotators.

Compared with image-based annotation, the annotation process with point clouds is more involved owing to the aforementioned view dependency. The annotator often needs to change the viewpoint to find a perspective where an annotation is possible without annotating unwanted or already labeled points. An annotator needs on average 4.5 h per 100×100 m^2 tile, when labeling residential areas: the most complex encountered scenes. Highway scenes are easier to label and an annotator needs on average 1.5 h to label a complete highway tile.

We provided feedback to the annotators to improve the quality and accuracy of labels. Furthermore, we also checked the labels in a second pass and inspected already labeled point clouds. During this check, inconsistencies were corrected and missing labels were added.

To ensure consistent annotations, a single annotator performed the verification of all sequences. Table 2 lists the fraction of points for each sequence, which we relabeled in the verification process. Thus, relabeling percentages are low for sequences mainly handled by this person, i.e., sequences 00, 09, and 13. The number of relabeled points

Table 2. Relabeling and instance correction statistics for training (top) and test data (bottom).

		00	01	02	03	04	05	06	07	08	09	10
All	Number of scans	4,541	1,101	4,661	801	271	2761	1,101	1,101	4,071	1,591	1,201
	Relabeled (%)	0.2	13.1	7.0	15.1	4.5	4.2	8.8	2.5	5.4	0.5	2.6
Non-moving	Number of instances	670	0	296	41	15	210	144	205	523	178	86
	Number of boxes	83,138	0	21,056	3,344	450	21,270	20,012	23,071	55,079	11,933	5,409
	Over-segmented (%)	20.3	0.0	37.2	46.3	46.7	27.6	25.7	31.2	24.5	25.8	34.9
	Under-segmented (%)	16.3	0.0	31.1	7.3	6.7	9.5	17.4	13.7	14.3	13.5	10.5
Moving	Number of instances	32	317	70	3	23	24	14	20	114	28	15
	Number of boxes	2,222	2,509	1,273	384	1,075	1,860	491	1,986	6,726	1,570	796
	ID switches (%)	50.0	19.2	11.4	66.7	60.9	50.0	50.0	60.0	31.6	42.9	20.0
		11	12	13	14	15	16	17	18	19	20	21
All	Number of scans	921	1,061	3,281	631	1,901	1,731	491	1,801	4,981	831	2,721
	Relabeled (%)	8.3	14.2	0.1	1.7	5.7	5.5	2.0	5.1	5.4	4.7	7.1
Non-moving	Number of instances	80	7	1589	0	189	276	5	371	919	6	3
	Number of boxes	4,636	540	175,147	0	17,911	24,810	130	35,697	86,101	441	96
	Over-segmented (%)	31.2	42.9	15.9	0.0	23.8	22.5	0.0	10.8	28.0	0.0	0.0
	Under-segmented (%)	17.5	14.3	37.5	0.0	10.6	12.7	40.0	32.6	15.9	0.0	0.0
Moving	Number of instances	7	84	102	0	13	31	26	174	142	456	1,853
	Number of boxes	197	1,909	5,208	0	571	416	537	8,133	5,630	15,945	32,332
	ID switches (%)	14.3	45.2	41.2	0.0	30.8	6.5	46.2	23.6	22.5	30.5	32.9

with the remaining sequences show the range of agreement between our annotators ranging from 2% up to 15%. The rather large spread compared with other image datasets (Cordts et al., 2016; Gupta et al., 2019; Lin et al., 2014) can be attributed to having annotators from different backgrounds, who had not previously annotated point clouds. Thus, we and the annotators had to learn and refine the process while creating the annotations.

Figure 3 shows examples of corrected labels during the verification process. Usually, we refined the boundaries of the road/sidewalk (e.g., curbs are always labeled as sidewalk), added details that where missed (e.g., tree trunks, guard rails, etc.), or refined the boundaries between different classes (e.g., building and sidewalk, car and road, etc.).

3.1.4. Instance annotation. For non-moving objects, we first cluster all points for each class using a fast grid-based segmentation approach (Behley et al., 2013). We use the aforementioned tiles to build a two-dimensional grid with cell size 0.1×0.1 m^2, where we insert all points using their x and y coordinates into the corresponding grid cells. Finally, only grid cells with points exceeding a height threshold of 0.5 m are considered and grouped into segments using a simple flood fill algorithm.

For moving objects, we cluster each scan individually using a distance-based clustering as this provided more reliable results and it can be used to associate instances between consecutive scans using the same principle. First, we search for each point its radius neighbors within 0.5 m and group points that share neighbors. To find associations with the previous four scans, we use a slightly larger radius of 1.0 m to find neighbors in the previous scans. If we find enough neighbors (i.e., we used at least 10 points) with the previous segments at different timestamps, we assign the same instance ID.

The described clustering leads inevitably to over- and under-segmentation, but also to wrong or missing associations between consecutive timestamps. We correct these issues manually using our point labeling tool, which allows us to create, join, and split instances.

Table 2 shows the fraction of over- and under-segmented non-moving objects that where manually corrected. For determining over- and under-segmented segments, we inspect all segments after the correction and compared the instance IDs before and after the correction of the corresponding segments. If we find multiple instance IDs in the segment before the correction, we record an over-segmentation. If we find that the segment before the correction is larger then the segment after the correction, we record this segment as an under-segmentation.

Note that we can exploit the pose information of the LiDAR sensor for non-moving objects, since we can cluster points based on their global coordinates. Only this allows us to inspect all 682,000 bounding boxes, because a large part of instances originate from non-moving objects.

However, Table 2 also reveals that 30–50% of all segments were affected by over- and under-segmentation, which had to be manually corrected. Our axis-aligned grid often under-segmented nearby parked cars in non-axis aligned directions and over-segmented cars that were farther away owing to the point distance.

For moving objects, we report in Table 2 the fraction of object trajectories that have at least once a different instance ID

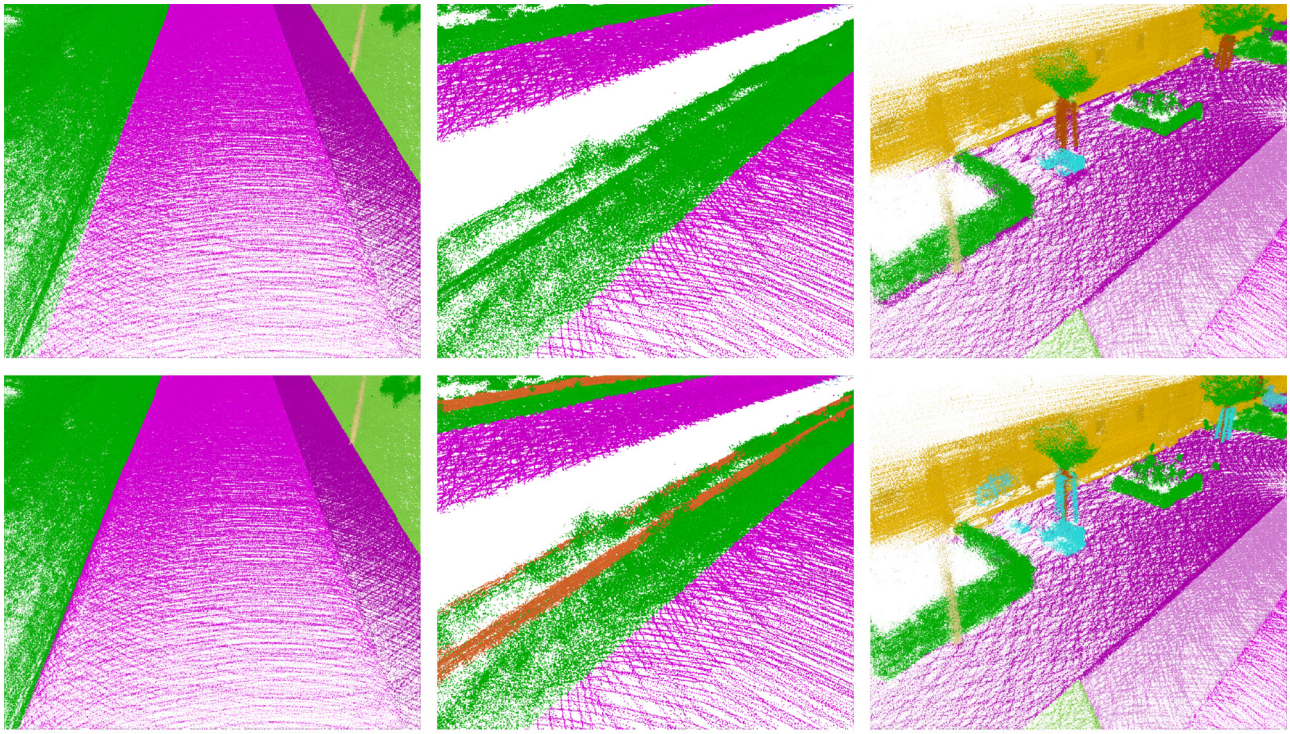


Fig. 3. Examples of correction during the verification process (top shows before validation and bottom after the validation). Here, we show (a) refined road boundaries, (b) added details (guard rail in the vegetation), and (c) refined boundaries between classes.

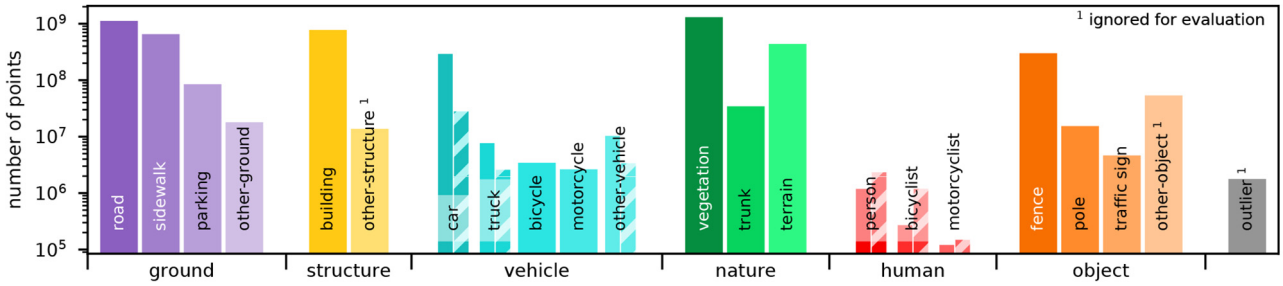


Fig. 4. Class-wise distribution over the complete dataset. For the potentially moving classes car, truck, other-vehicle, person, bicyclist, motorcyclist, we furthermore distinguish between non-moving (solid) and moving instances (dashed bars).

in the automatically extracted instances. These data association errors were mainly caused by occlusions or by cars moving in the same direction with a small gap. Using here a tracking-based data association, which accounts for the object motion, would have resolved many of these issues.

3.2. Class and instance distribution

Overall, we annotated over 500 tiles resulting in the class distribution shown in Figure 4. In addition to traffic participants, such as car, truck, bicyclist, motorcyclist, we also included more functional ground classes relevant for autonomous driving, such as sidewalk and parking areas. Note that we also differentiate for potentially moving classes between non-moving, i.e., the object did not move while observing it, and moving objects, i.e., the object moved at least for a single frame while the sensor observed it. In total, we distinguish between 28 classes including the moving and non-moving classes.

Note that we label points belonging to a bicycle or a motorcycle also as bicyclist or motorcyclist, when the vehicle has a rider. This simplification was needed, because persons on a bicycle or motorcycle cannot be separated reliably and accurately from the vehicle. Owing to the sparser resolution of the point clouds compared with images, we preferred to label only the complete objects.

Naturally, road, vegetation, and building points are by far the most frequently occurring classes. Motorcyclist is a class with only few examples present in the data.

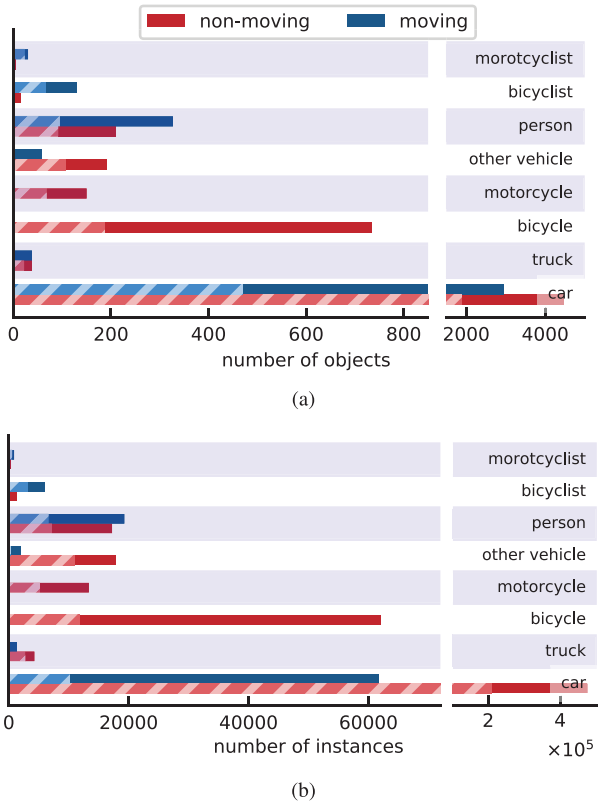


Fig. 5. Instance-wise statistics over individual (a) object occurrences and (b) total count of bounding boxes or instances. Dashed bars show the proportion of objects in the training and validation data (sequences 00–10).

The class car is by far the most often occurring traffic participant, which can be also seen in the object count depicted in Figure 5 . We show in the upper part of Figure 5 the sequence-wise counts of instance annotations, i.e., we count each object only once even if it is seen multiple times by the sensor. The lower part of the figure shows the accumulated scan-wise counts of instances, where we count the instances without considering the temporally consistent instance ID.

3.3. Dataset organization and format

We tried to keep the data organization as close as possible to the original KITTI Vision Benchmark. Thus, we use the same format for our pose files and a similar binary format corresponding to the point cloud format of KITTI. Pose information is given in the coordinate frame of the camera, therefore the calibration data of each sequence provided by the KITTI odometry benchmark is needed to correctly transform the point clouds.

We added the folder labels containing for each point cloud a binary label file. Corresponding to each point, this file contains a 32-bit unsigned integer, where the upper 16 bits contain the instance ID and the lower 16 bits contain the class ID. Here, a value of zero corresponds to no instance assigned to the corresponding point and a zero label corresponds to an unlabeled point.

The folder voxels contains the voxelized point clouds, the voxel-wise labels, the invalid voxels, and the occluded voxels. Here, we opted for reducing the size of the data by using only bit flags to encode 8 voxels inside a single byte. Labels are represented by a 16-bit integer and we do not distinguish between different instances.

3.4. Benchmark competitions and development kit

Together with the data, we also provide competitions on CodaLab Competitions that allow evaluating approaches on a hidden test set, i.e., part of the data for which we do not provide annotations. We use the original test sequences 11–21 of the odometry benchmark, because we do not want to interfere with the original benchmark which could exploit the provided labels to get better results on the original odometry benchmark. The large test set might furthermore incentivize the development of fast approaches to process the large number of scans.

We currently provide competitions for semantic segmentation, semantic scene completion, and panoptic segmentation. More information regarding the baselines and metrics are provided in the corresponding papers (Behley et al., 2019, 2020).

Together with the dataset, we also published a development kit[2] implemented in Python that provides methods to read the data and contains all evaluation scripts used in the aforementioned competitions. We furthermore provide tools to visualize the point clouds and the voxel grids.

4. Lessons learned

In the process of labeling around 43,000 point clouds with a team of nine annotators, we learned quite a bit about managing the annotation process, but also encountered pitfalls. From these experiences, we want to share some insights that might help others to organize annotating data at such scale and avoid some failures.

We relied on university students from computing related, but also unrelated fields. Accurate and still efficient annotators are hard to find and we had a selection process or casting, where we let the prospective annotators use our point cloud annotation application and could directly see how they performed in the session. During this process, we recognized that point cloud annotation is quite challenging and that even annotators that are comfortable with labeling images had big problems to navigate and transfer this knowledge to the annotation of point clouds. Thus, annotator screening turned out to be a mandatory first step to achieve good results.

However, we learned most about the problems and challenges by labeling ourselves. This also helps to rectify the expectations and helps in the selection of the targeted classes. We found early on that labeling lane markings

would have increased the time per tile to much, such that we decided not to include these in our labeling effort.

Next, we had to invest additional time in training and correcting the annotators continuously to ensure sufficient quality of the provided annotations. Only this ensured a high quality of the resulting annotations and a speed up in the process. Interestingly, all annotators improved during this process substantially and could adopt our suggestions to label more efficiently. As we also used our own tools, we could develop and refine our annotation application over time and introduce new ways of filtering points, which also decreased the time needed to label a point cloud. Initially, we relied mostly on a height-based ground removal, which was not sufficient to consistently remove ground points. Adding a plane-based removal, where the annotator could set the angles and the threshold along the z-axis of the associated plane, proved to be much more robust.

Nevertheless, some of our early attempts to speed up the process by using pre-trained and fine-tuned image segmentation methods did not succeed. First, we aimed at having consistent labels for the aggregated point cloud and therefore the projection of single image labels to the point cloud needed to be aggregated using a point-wise majority vote. Second, the view point differences between the LiDAR sensor and the camera led to occlusions that could not always be resolved by aggregating multiple projected image labels, because one had only a frontal view available. Third, the misalignment between camera and LiDAR caused by pixel shifts or simply the rotation of the LiDAR resulted in "bleeding" of semantic labels into background regions or inconsistent labels on objects where parts of nearby classes got projected onto. Surprisingly, correction of these boundary issues took nearly as much time as labeling the whole tile from scratch. In addition, our impression was also that the transferred labels biased the annotations in being "good enough," but being actually inferior to a labeling from scratch, because error or small problems with the projection of annotations are hardly visible.

5. Conclusion

SemanticKITTI provides point-wise semantic and instance annotations for all sequences of the KITTI odometry benchmark. At the time of writing, it is still the largest dataset that provides such annotations for point cloud sequences. Based on these annotation, we provide benchmarks for semantic segmentation and semantic scene completion (Behley et al., 2019). We added recently panoptic segmentation joining semantic and instance segmentation to the set of benchmarks (Behley et al., 2020). Furthermore, we plan to extend the set of benchmarks with additional tasks.

In addition to these benchmarks, we also see already adoption of the data for other domains benefiting from semantics, such as semantic SLAM (Chen et al., 2019; Gan et al., 2020), LiDAR-based localization (Yan et al., 2019), and loop closure detection (Chen et al., 2020). Moreover, domain adaptation (Jaritz et al., 2020; Langer et al., 2020) was also quite recently investigated to exploit our annotations for other sensors with different sensor geometries. We are curious in which ways the annotations and our tools will be used in the future.

Acknowledgements

We thank all the students that helped with annotating the data.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under FOR 1505 Mapping on Demand (grant numbers BE 5996/1-1 and GA 1927/5-2 (FOR 2535)) and under Germany's Excellence Strategy (grant number EXC-2070 – 390732324 (PhenoRob)).

ORCID iD

Jens Behley https://orcid.org/0000-0001-6483-0319

Notes

1. See http://www.cvlibs.net/datasets/kitti/eval_odometry.php for more information and for downloading the data.
2. Available at: https://github.com/PRBonn/semantic-kitti-api

References

Behley J, Garbade M, Milioto A, et al. (2019) SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Behley J, Milioto A and Stachniss C (2020) A benchmark for LiDAR-based panoptic segmentation based on KITTI. *arXiv preprint*.

Behley J and Stachniss C (2018) Efficient surfel-based SLAM using 3D laser range data in urban environments. In: *Proceedings of Robotics: Science and Systems (RSS)*.

Behley J, Steinhage V and Cremers A (2013) Laser-based segment classification using a mixture of bag-of-words. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Behley J, Steinhage V and Cremers AB (2012) Performance of histogram descriptors for the classification of 3D laser range data in urban environments. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Caesar H, Bankiti V, Lang AH, et al. (2020) NuScenes: A multimodal dataset for autonomous driving. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chang MF, Lambert J, Sangkloy P, et al. (2019) Argoverse: 3D tracking and forecasting with rich maps. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chen X, Läbe T, Milioto A, et al. (2020) OverlapNet: Loop closing for LiDAR-based SLAM. In: *Proceedings of Robotics: Science and Systems (RSS)*.

Chen X, Milioto A, Palazzolo E, Giguère P, Behley J and Stachniss C (2019) SuMa++: Efficient LiDAR-based semantic SLAM. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Cordts M, Omran M, Ramos S, et al. (2016) The Cityscapes Dataset for semantic urban scene understanding. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Deng J, Dong W, Socher R, Li L, Li K and Fei-Fei L (2009) ImageNet: A large-scale hierarchical image database. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2009.5206848.

Everingham M, Van Gool L, Williams C, Winn J and Zisserman A (2010) The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88(2): 303–338.

Gan L, Zhang R, Grizzle J, Eustice R and Ghaffri M (2020) Bayesian spatial kernel smoothing for scalable dense semantic mapping. *IEEE Robotics and Automation Letters* 5(2): 790–797.

Geiger A, Lenz P and Urtasun R (2012) Are we ready for autonomous driving? The KITTI Vision Benchmark Suite. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361.

Geiger A, Lenz P, Stiller C and Urtasun R (2013) Vision meets robotics: The KITTI Dataset. *The International Journal of Robotics Research* 32(11): 1231–1237.

Geyer J, Kassahun Y, Mahmudi M, et al. (2020) A2D2: Audi Autonomous Driving Dataset. *arXiv preprint* 2004.06320.

Gupta A, Dollar P and Girshick R (2019) LVIS: A dataset for large vocabulary instance segmentation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hackel T, Savinov N, Ladicky L, Wegner JD, Schindler K and Pollefeys M (2017) SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-1-W1: 91–98.

Jaritz M, Vu TH, Charette R, Wirbel E and Perez P (2020) XMUDA: Cross-modal unsupervised domain adaptation for 3D semantic segmentation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kesten R, Usman M, Houston J, et al. (2019) Lyft Level 5 AV Dataset 2019. Available at: https://level5.lyft.com/dataset/ (accessed 25 March 2021).

Kirillov A, He K, Girshick R, Rother C and Dollár P (2019) Panoptic segmentation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Langer F, Milioto A, Haag A, Behley J and Stachniss C (2020) Domain transfer for semantic segmentation of LiDAR data using deep neural networks. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

LeCun Y, Bengio Y and Hinton G (2015) Deep learning. *Nature* 521: 436–444.

Lin TY, Maire M, Belongie S, et al. (2014) Microsoft COCO: Common objects in context. In: *Proceedings of the European Conference on Computer Vision (ECCV)*.

Montemerlo M, Becker J, Bhat S, et al. (2008) Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics* 25(9): 569–597.

Munoz D, Bagnell JA, Vandapel N and Hebert M (2009) Contextual classification with functional max-margin Markov networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Neuhold G, Ollmann T, Bulo SR and Kontschieder P (2017) The Mapillary Vistas Dataset for semantic understanding of street scenes. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Pan Y, Gao B, Mei J, Geng S, Li C and Zhao H (2020) SemanticPOSS: A point cloud dataset with large quantity of dynamic instances. *arXiv preprint* 2002.09147.

PandaSet (2020) PandaSet Dataset. Available at: https://scale.com/open-datasets/pandaset (accessed 25 March 2021).

Patil A, Malla S, Gang H and Chen YT (2019) The H3D dataset forfull-surround 3D multi-object detection and tracking in crowded urban scenes. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Pitropov M, Garcia DE, Rebello J, et al. (2021) Canadian Adverse Driving Conditions dataset. *The International Journal of Robotics Research*. DOI: 10.1177/0278364920979368.

Roynard X, Deschaud JE and Goulette F (2018) Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *The International Journal of Robotics Research* 37(6): 545–557.

Sun P, Kretzschmar H, Dotiwalla X, et al. (2020) Scalability in perception for autonomous driving: Waymo Open Dataset. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Urmson C, Anhalt J, Bagnell D, et al. (2008) Autonomous driving in urban environments: Boss and the Urban Challenge. *Journal of Field Robotics* 25(8): 425–466.

Yan F, Vysotska O and Stachniss C (2019) Global localization on OpenStreetMap using 4-bit semantic descriptors. In: *Proceedings of the European Conference on Mobile Robotics (ECMR)*.

Zhang R, Candra SA, Vetter K and Zakhor A (2015) Sensor fusion for semantic segmentation of urban scenes. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[D] N. Chebrolu, T. Läbe, O. Vysotska, *et al.*, "Adaptive Robust Kernels for Non-Linear Least Squares Problems," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2240–2247, 2021. DOI: 10.1109/LRA.2021.3061331

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

Adaptive Robust Kernels for Non-Linear Least Squares Problems

Nived Chebrolu Thomas Läbe Olga Vysotska Jens Behley Cyrill Stachniss

***Abstract*— State estimation is a key ingredient in most robotic systems. Often, state estimation is performed using some form of least squares minimization. Basically, all error minimization procedures that work on real-world data use robust kernels as the standard way for dealing with outliers in the data. These kernels, however, are often hand-picked, sometimes in different combinations, and their parameters need to be tuned manually for a particular problem. In this paper, we propose the use of a generalized robust kernel family, which is automatically tuned based on the distribution of the residuals and includes the common m-estimators. We tested our adaptive kernel with two popular estimation problems in robotics, namely ICP and bundle adjustment. The experiments presented in this paper suggest that our approach provides higher robustness while avoiding a manual tuning of the kernel parameters.**

I. Introduction

State estimation is a central building block in robotics and is used in a variety of different components, such as simultaneous localization and mapping (SLAM) [22]. A large number of state estimation solvers perform some form of non-linear least squares minimization. Prominent examples are the optimization of SLAM graphs, the ICP algorithm, visual odometry, or bundle adjustment (BA), which all seek to find the minimum of some error function. As soon as real-world data is involved, outliers will occur in the data. A common source of such outliers stems from data association mistakes, for example, when matching features.

To avoid that even a few such outliers have strong effects on the final solution, robust kernel functions are used to down-weight the effect of gross errors. Several robust kernels have been developed to deal with outliers arising in different situations. Prominent examples include the Huber, Cauchy, Geman-McClure, or Welsch functions that can be used to obtain a robustified estimator [27].

However, the proper choice of the best kernel for a given problem is not straightforward. As the robust kernels define the distribution from which the outliers are generated, their choice is problem-specific. In practice, the choice of the kernel is often done in a trial and error manner, as in most situations there is no prior knowledge of the outlier process. For some approaches such as bundle adjustment, today's implementations even vary the kernel between iterations or pair them with outlier rejection heuristics. Moreover, for several robotics applications such as SLAM, the outlier distribution itself changes continuously depending on the structure of the environment, dynamic objects in the scene and other environmental factors like lighting. This often means that a fixed robust kernel chosen a-priori cannot deal effectively with all situations.

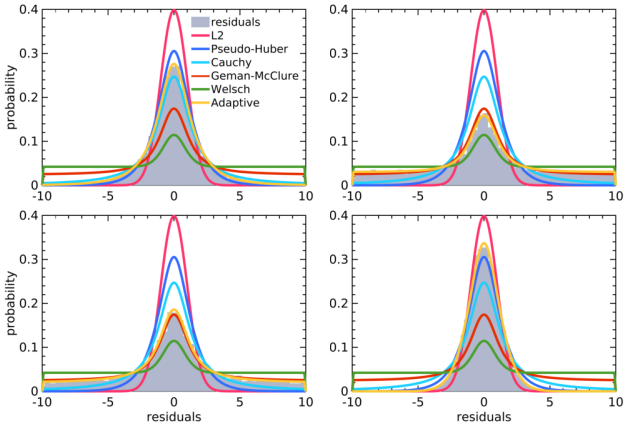


Fig. 1. Probability densities of different robust kernels. The adaptive robust kernel (in yellow) is able to describe the actual residual distribution in different situations better than a fixed robust kernel for all cases. As a result, it provides better robustness to different types of outliers during the state estimation process.

In this paper, we aim at circumventing the trial and error process for choosing a kernel and at exploring the automatic adaptation of kernels to the outliers online. To achieve this, we use a family of robust loss functions proposed by Barron [6], which generalizes several popular robust kernels such as Huber, Cauchy, Geman-McClure, Welsch, etc. The key idea is to dynamically tune this generalized loss function automatically based on the current residual distribution so that one can blend between such robust kernels and make the choice a part of the optimization problems.

The main contribution of this paper is an easy-to-implement approach for dynamically adapting the robust kernels in non-linear least squares (NLS) solvers, which builds on top of the generalized formulation of Barron [6]. We achieve this by estimating a hyper-parameter for a generalized loss function, which controls the shape of the robust kernel. This parameter becomes part of the estimation process and we determine it along with the unknown parameters of the model. We extend the usable range of this parameter compared to the formulation of Barron [6]. This allows us to better deal with a larger set of outlier distributions compared to fixed kernels and to the Barron formulation. See Fig. 1 for a visualization.

In sum, we make the following key claims. Our approach can (i) perform robust estimation without committing to a

All authors are with the University of Bonn, Germany.

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob as well as by the EC under grant agreement no. 101017008 - Harmony.

fixed kernel beforehand, (ii) adapt the shape of the kernel to the actual outlier distribution, and (iii) illustrate the performance on two common example problems, namely ICP and bundle adjustment.

II. Related Work

Robust kernels are the de-facto solution to perform state estimation using least-squares minimization in the presence of outliers. To deal with different outlier distributions, several robust kernels such as Huber, Cauchy, Geman-McClure, or Welsch have been proposed in the literature. Zhang [27] and Bosse [9] apply these kernels to different kind of estimation problems in vision and robotics. Black and Rangarajan [8] investigate equivalence between robust loss minimization and outlier processes, and apply this idea to several vision problems such as surface reconstruction, segmentation, optical flow etc. Babin *et al*. [5] analyzed several popular robust kernels for registration problems and provide advice for using different kernels depending on the scenario. Similar analysis and recommendations exists for visual odometry and BA in [19], [26].

In this work, instead of choosing a specific robust kernel for a particular scenario, we dynamically adapt a robust kernel to the actual outlier distribution during the optimization process. To do this, we build upon the generalized kernel formulation recently proposed by Barron [6] for training neural networks. It generalizes over popular robust kernels and we formulate an approximation of it for the use in NLS estimation.

For pose graph SLAM problems, several approaches exist to deal with the outliers dynamically [22], [2]. Sünderhauf and Protzel [23] propose introducing additional switch variables to the original optimization problem, which determines whether an observation should be used or discarded during optimization. The RRR approach [18] is a robust SLAM back-end that targets to reject false constraints through constraint clustering and mutually consistency checks. Agarwal *et al*. [3] propose a robust kernel, which dynamically weighs the observations without requiring to estimate any additional variables. Lajoie *et al*. [17] and Yang *et al*. [25] further this idea as a truncated least squares problem which can be solved efficiently as a semi-definite program. They also provide solutions with certain robustness guarantees for the registration and SLAM problem. Recently, Yang *et al*. [24] propose a robust estimation framework based on graduated non-convexity (GNC) methods which solves a sequence of minimization problems which are convex initially, and converge eventually to the original non-convex robust loss.

Taking a probabilistic view, several robust kernels are understood to arise from a probability distribution, which can be used to determine the best kernel type based on the actual observations. Agamennoni *et al*. [1] propose to use an elliptical distribution to represent several popular robust kernels. They estimate hyper-parameters for each kernel type based on the residual distribution and perform a model comparison to determine the best kernel for the situation at hand. In this paper, we take a different approach and adapt the robust kernel shape by using the probability distribution of a generalized loss function [6]. We do not require an explicit model comparison to choose the best kernel and estimate the kernel shape through an alternating minimization procedure.

III. Least Squares with an Adapting Kernel

Our approach targets dynamically adapting robust kernels when solving NLS problems by estimating a hyper-parameter that controls the shape of the robust kernel. This parameter becomes part of the estimation process in an alternating error minimization procedure. Before explaining our approach, we first explain robust NLS estimation and generalized kernels to give the reader a complete view (Sec. III-A). We then present the generalized robust kernel proposed by Barron [6], which is the foundation of our work (Sec. III-B). We extend Barron's robust kernel to deal with strong outliers typically encountered in robotics applications (Sec. III-C) and use it for solving typical state estimation problems (Sec. III-D).

A. Robust Least Squares Estimation

Several state estimation problems in robotics involve estimating unknown parameters θ of a model given noisy observations z_i with $i = 1, \dots, N$. These problems are often framed as non-linear least squares optimization, which aims to minimize the squared loss:

$$\theta^* \quad = \quad \underset{\theta}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{N} w_i \|r_i(\theta)\|^2, \tag{1}$$

where $r_i(\theta) = f_i(\theta) - z_i$ is the residual and w_i is the weight for the i^{th} observation. The estimate θ^* is statistically optimal if the error on the observations z_i is Gaussian. In case of non-Gaussian noise, however, the estimate θ^* can be arbitrarily poor [13]. To reduce this impact of outliers, subquadratic losses can be applied. The main idea of a robust loss is to downweight large residuals that are assumed to be caused from outliers such that their influence on the solution is reduced. This is achieved by optimizing:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \rho(r_i(\theta)), \tag{2}$$

where $\rho(r)$ is also called the robust loss or kernel. Several robust kernels have been proposed to deal with different kinds of outliers such as Huber, Cauchy, and others [27]. A summary of several popular robust kernels can be found in the work by MacTavish *et al*. [19].

The optimization problem in Eq. (2) can be solved using the iteratively reweighted least squares (IRLS) approach [27], which solves a sequence of weighted least squares problems. We can see the relation between the least squares optimization in Eq. (1) and robust loss optimization in Eq. (2) by comparing the respective gradients which go to zero at the optimum (illustrated only for the i^{th} residual):

$$\frac{1}{2} \frac{\partial (w_i r_i^2(\theta))}{\partial \theta} \quad = \quad w_i r_i(\theta) \frac{\partial r_i(\theta)}{\partial \theta} \tag{3}$$

$$\frac{\partial (\rho(r_i(\theta)))}{\partial \theta} \quad = \quad \rho'(r_i(\theta)) \frac{\partial r_i(\theta)}{\partial \theta}. \tag{4}$$

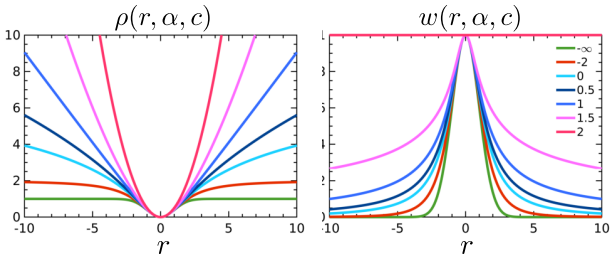


Fig. 2. Left: General robust loss $\rho(r, \alpha, c)$ takes different shapes depending on the value of α. Right: Corresponding weights for kernels with different α values. A smaller α corresponds to a larger down-weighting of the residuals.

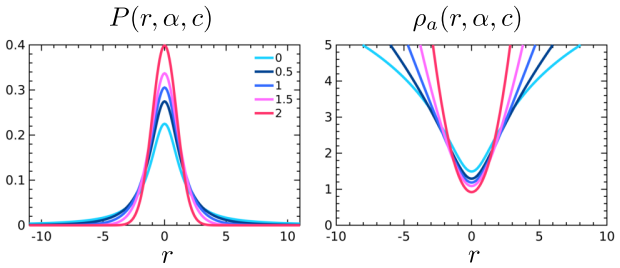


Fig. 3. Left: Probability distribution $P(r, \alpha, c)$ of generalized robust loss function for different values of α. Right: Adaptive robust loss $\rho_a(r, \alpha, c)$ obtained as the negative log-likelihood of $P(r, \alpha, c)$. This adaptive loss enables automatic tuning of $\alpha \in [0, 2]$.

By setting the weight $w_i = \frac{1}{r_i(\theta)}\rho'(r_i(\theta))$, we can solve the robust loss optimization problem by using the existing techniques for weighted least-squares. This scheme allows standard solvers using Gauss-Newton and Levenberg-Marquardt algorithms to optimize for robust losses and is implemented in popular optimization frameworks such as Ceres [4], g2o [15], and iSAM [14].

B. Adaptive Robust Kernel

Barron [6] proposes a single robust kernel that generalizes for several popular kernels such as pseudo-Huber/L1-L2, Cauchy, Geman-McClure, Welsch. The generalized kernel ρ is given by:

$$\rho(r, \alpha, c) = \frac{|\alpha - 2|}{\alpha}\left(\left(\frac{(r/c)^2}{|\alpha - 2|} + 1\right)^{\alpha/2} - 1\right), \quad (5)$$

where α is a real-valued parameter that controls the shape of the kernel and $c > 0$ is the scale parameter that determines the size of quadratic loss region around $r = 0$. Adjusting the parameter α essentially allows us to realize different robust kernels. Some special cases are squared/L2 loss ($\alpha = 2$), pseudo-Huber/L1-L2 ($\alpha = 1$), Cauchy ($\alpha = 0$), Geman-McClure ($\alpha = -2$), and Welsch ($\alpha = -\infty$). Note that $\rho(r, \alpha, c)$ in Eq. (5) is not defined for $\alpha = 0$ and $\alpha = 2$, and are instead defined as the respective pointwise limits at these values (see Eq. (8) of Barron's paper [6]).

The general loss function $\rho(r, \alpha, c)$ and the corresponding weights curve $w(r, \alpha, c)$ are illustrated in Fig. 2 for several values of α. The shape of the weights curve provides an insight into the influence that a residual has on the solution

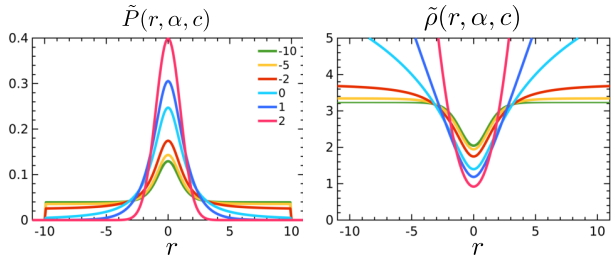


Fig. 4. Left: Modified probability distribution $\tilde{P}(r, \alpha, c)$ obtained by truncating $P(r, \alpha, c)$ at $|r| < \tau$. Right: The truncated robust loss $\tilde{\rho}_a(r, \alpha, c)$ allows the automatic tuning of α in its complete range, including $\alpha < 0$.

while minimizing the robust loss function in Eq. (2). For example, for $\alpha = 2$, the weights for all residuals are one, meaning that all residuals are treated the same. Whereas for $\alpha = -\infty$, all residuals greater than $3c$ will not affect the solution θ^* significantly as they are weighed down to very small values.

With this generalized robust loss, we can interpolate between a range of robust kernels simply by tuning α. To automatically determine the best kernel shape through the parameter α, we treat α as an additional unknown parameter while minimizing the generalized loss:

$$(\theta^*, \alpha^*) = \underset{(\theta,\alpha)}{\operatorname{argmin}} \sum_{i=1}^{N} \rho(r_i(\theta), \alpha). \quad (6)$$

However, this optimization problem in Eq. (6) can be trivially minimized by choosing an α that weighs down all residuals to small values without affecting the model parameters θ, essentially treating all data points as outliers. Barron [6] avoids this situation by constructing a probability distribution based on the generalized loss function $\rho(r, \alpha, c)$ as

$$P(r, \alpha, c) = \frac{1}{cZ(\alpha)} e^{-\rho(r,\alpha,c)} \quad (7)$$

$$Z(\alpha) = \int_{-\infty}^{\infty} e^{-\rho(r,\alpha,1)}\, dr, \quad (8)$$

where $Z(\alpha)$ is a normalization term, also called partition function, which defines an adaptive general loss as the negative log-likelihood of Eq. (7),

$$\rho_a(r, \alpha, c) = -\log P(r, \alpha, c) \quad (9)$$

$$= \rho(r, \alpha, c) + \log cZ(\alpha). \quad (10)$$

The adaptive loss $\rho_a(\cdot)$ is simply the general loss $\rho(\cdot)$ shifted by the log partition. This shift introduces an interesting trade-off. A lower cost for increasing the set of outliers comes with a penalty for the inliers and vice versa. This trade-off forces the optimization in Eq. (6) to choose a suitable value for α instead of trivially ignoring all residuals by turning every data point into an outlier. The probability distribution $P(r, \alpha, c)$ and the adaptive loss function are plotted in Fig. 3 for visualization.

Algorithm 1 Optimization with adaptive robust kernel

1: Initialize $\theta^0 = \theta, \alpha^0 = 2, c$
2: **while** !converged **do**
3: Step 1: Minimize for α
4: $\alpha^t = \text{argmin}_\alpha - \sum_{i=1}^{N} \log P(r_i(\theta^{t-1}), \alpha^{t-1}, c)$
5: Step 2: Minimize robust loss using IRLS
6: $\theta^t = \text{argmin}_\theta \sum_{i=1}^{N} \rho(r_i(\theta), \alpha^t, c)$,

C. Truncated Robust Kernel

The probability distribution $P(r, \alpha, c)$ is only defined for $\alpha \geq 0$, as the integral in the partition function $Z(\alpha)$ is unbounded for $\alpha < 0$. This means that values for $\alpha < 0$ cannot be achieved while minimizing the adaptive loss $\rho_a(\cdot)$ in Eq. (9). This limits the range of kernels that we can dynamically adapt to. As we can see in Fig. 2, the smaller the parameter α is, the stronger is the down-weighting of outliers. Such a behavior is often desired in situations where a large number of outliers are present in the data.

In this paper, we propose an extension to the adaptive loss in Eq. (9), which allows the parameter α to be dynamically adapted for a larger range of values. We achieve this by limiting the partition to bounded values. To re-gain the kernels corresponding to the negative range of α with the adaptive loss function, we compute an approximate partition function $\tilde{Z}(\alpha)$ as

$$\tilde{Z}(\alpha) = \int_{-\tau}^{\tau} e^{-\rho(r,\alpha,1)} dr, \tag{11}$$

where τ is the truncation limit for approximating the integral. This results in a finite partition $\tilde{Z}(\alpha)$ for all α as the integral is computed within the limits $[-\tau, \tau]$. We use this to define our truncated loss function as

$$\tilde{\rho}_a(r, \alpha, c) = \rho(r, \alpha, c) + \log c\tilde{Z}(\alpha). \tag{12}$$

The truncated probability distribution $\tilde{P}(r, \alpha, c)$ and the corresponding truncated loss $\tilde{\rho}_a(r, \alpha, c)$ is shown in Fig. 4. Since the truncated loss is defined for all values of α including $\alpha < 0$, we can adapt α in its entire range during the optimization procedure. We discuss the effect of the truncation of the loss function below in Sec. III-E.

D. Optimization of α via Alternating Minimization

We propose to solve the joint optimization problem over θ and α defined in Eq. (6) in an iterative manner using an alternating minimization procedure. The procedure alternates between two steps: (i) the first step where the maximum likelihood value for α is computed, and (ii) the second step, where the optimal parameters for the model given the α from the previous step is computed. This can be seen as a variation of a coordinate descent approach. By solving the joint optimization in this manner, we decouple the estimation of the robust kernel parameter α from the original optimization problem. This allows to us solve for the model parameters θ in the same way as before α was introduced.

We estimate the parameters α in the first step by minimizing the negative log-likelihood of observing the current residuals,

$$L(\alpha) = -\sum_{i=1}^{N} \log P(r_i(\theta), \alpha, c) \tag{13}$$

$$= \sum_{i=1}^{N} \log c\tilde{Z}(\alpha) + \rho_a(r_i(\theta), \alpha, c), \tag{14}$$

i.e.,

$$\alpha^* = \underset{\alpha}{\text{argmin}}\, L(\alpha). \tag{15}$$

The solution to Eq. (15) can be obtained by setting its first derivative $\frac{dL(\alpha)}{d\alpha} = 0$. Since its not possible to derive the partition function $\tilde{Z}(\alpha)$ analytically, we settle for a numerical solution. As α is a scalar value, $L(\alpha)$ can be minimized simply by performing a 1-D grid search for $\alpha \in [\alpha_{min}, 2]$.

In terms of a practical implementation, we chose lower bound $\alpha_{min} = -10$ as its difference to the corresponding weights for $\alpha = -\infty$ for large residuals ($|r| > \tau$) is small. The maximum value for α is set to 2 as this corresponds to the standard least squares problem. The scale c of the robust loss is fixed beforehand and not adapted during the optimization. This value for c is usually fixed based on the measurement noise for an inlier observation z and the accuracy of the initial solution. To be computationally efficient, we pre-compute $\tilde{Z}(\alpha)$ as a lookup table for values $\alpha \in [\alpha_{min}, 2]$ with a resolution of 0.1 and use the lookup table during optimization. This leads us to the overall minimization approach shown in Alg. 1.

E. Effect of Using the Truncated Loss

The second step, which is the *minimization* in the NLS estimation, is not affected by the truncated loss approximation. It can, however, affect the first step, i.e., determining the parameter α. By using our truncated loss, we are implicitly assuming that no outliers have a residual $|r| > \tau$ during the first step. If we choose a large enough value for τ, the error that we introduce affects situations with large outliers only and therefore results in small values of α. The effect of small α values such as $\alpha = -10$ vs. $\alpha = -\infty$ on the optimization, however, is negligible as the outliers will be down-weighted to basically zero. We observe in our experiments that by setting choosing $\tau = 10c$, we are able to deal with almost all of the outlier distributions occurring in practice for ICP, SLAM, BA applications.

IV. Experimental Evaluation

In the experimental section, we evaluate the performance of our adaptive robust kernel approach on two common problems in robotics, (i) registration using Iterative Closest Point (ICP) algorithm, and (ii) bundle adjustment as an example state estimation task. The experiments are designed to evaluate effectiveness of our approach in presence of strong outliers, and showcase its applicability for common NLS problems. We compare the performance of our approach

against a hand-crafted outlier rejection mechanism using fixed robust kernels and the original adaptive kernel formulation by Barron [6] on a benchmarking dataset and conduct analysis on simulated data to understand the convergence properties.

A. Application to Iterative Closest Point

The first experiment is designed to show the advantages of our approach for LiDAR-based registration in form of ICP. We integrated our truncated adaptive robust kernel into an existing SLAM system, called surfel-based mapping (SuMa) [7], which performs point-to-plane projective ICP for 3D LiDAR scans. The ICP registration is performed in a frame-to-frame fashion on consecutive scans. We compare the performance of our approach against two fixed robust kernels, i.e., Huber and Geman-McClure, as well as to a hand-crafted outlier rejection scheme as used in the original implementation of SuMa [7]. This hand-crafted scheme combines a Huber kernel with an additional outlier rejection step that removes all correspondences, which have a distance of more than $2\,m$ or which have an angular difference greater than 30° between the estimated normals of observations and the corresponding normals of the surfels. Finally, we also compare it to the original adaptive kernel by Barron [6].

We evaluate all these approaches on the odometry datasets of the KITTI vision benchmark [12] and summarize the results in Tab. I. The best performance in terms of relative translation error for each sequence is highlighted in bold. We observe that our proposed approach, which does *not* require an outlier rejection step at all, performs better or is on-par with fixed kernel plus outlier rejection scheme for many of the sequences. At the same time, using only the fixed kernel without the outlier rejection step fails for some of the sequences. In particular for Sequence 04, both the fixed kernel using Huber and the hand-crafted outlier rejection scheme fail, whereas our approach performs the best on this sequence. Our approach performs slightly better with respect to the adaptive kernel by Barron [6], with the biggest gain in Sequence 01, which requires negative α values to deal with the outliers coming from dynamic objects in the scene. On average over all the sequences, our approach provides the best accuracy in terms relative translation error. Here, we note that our approach is not the best in terms of relative rotational error, but is around the 1% mark, which is on-par with other approaches. These results are promising as by using our adaptive robust kernel, we do not need any hand-crafted outlier rejection mechanism, which in practice requires manual tuning for new data or different sensor configurations.

As a qualitative evaluation, we illustrate the advantage of using the adaptive robust kernel for a challenging dataset (Sequence 01), which contains several moving cars moving with the vehicle itself along the highway with little additional geometric structures. In Fig. 5 (top-left), we plot the values of α for each iteration while mapping the sequence. We observe that α adapts to smaller and more negative values whenever there are more outliers, which arise mainly from moving vehicles in the scan. This effect can be seen in Fig. 5 (bottom-left) where the translation error for the fixed Huber kernel increases as it cannot handle the outlier situation well. At the same time, the error remains small for our adaptive kernel.

The two 3D scenes show the registrations at the same point in time, once computed with the adaptive kernel and once with a fixed one. The adaptive kernel results in a successful alignment while the fixed kernel fails to find the correct solution due to the outlier in the data association, see Fig. 5 (middle and right). The adaptive kernel-based ICP can correctly treat the observations belonging to the moving car as outliers, and nullify their effect during optimization automatically. For this sequence, we note that for large portions of the scans, α is negative and even reaches down to $\alpha_{\text{min}} = -10$ in some instances. This suggests that our truncated adaptive loss proposed in Eq. (12) is critical for the successful application of ICP as it enables using values $\alpha < 0$, whereas the original formulation of the adaptive loss is limited to $\alpha \in [0, 2]$. Thus, our approach greatly supports ICP-based registration as it avoids hand-crafted outlier strategies and adapts to the outlier challenges present in each pairs of scans automatically.

B. Application to Bundle Adjustment (BA)

The second experiment is designed to illustrate the performance of our approach and its advantages for the bundle adjustment problem using a monocular camera. We integrated the adaptive robust kernel to an existing bundle adjustment framework proposed by Schneider *et al*. [21]. The initial estimate for camera poses and 3D points is obtained by three commonly used steps. First, extract SIFT features and compute possible matches between all image pairs. Second, compute the relative orientation using Nister's 5-point algorithm [20] together with RANSAC for outlier rejection and chaining the subsequent images to obtain the initial camera trajectory. Third, compute the 3D points as described by Läbe *et al*. [16] given the camera trajectory of the second step.

To test the bundle adjustment performance, we created four datasets covering different scenarios using the CARLA simulator [11] generating near-realistic images. The advantage of the simulator is that ground truth information for the cameras poses is available. The first dataset contains images from a front looking camera mounted on a car, the second dataset simulates downward looking aerial images from a UAV, the third dataset contains images where around half of each image shows strong shadows, and the fourth dataset simulates side-ward looking camera where close-by objects suffer from significant motion blur. We have generated these datasets to cover situations where feature matching is challenging, thereby resulting in a large number of outliers. Example images from the all the four datasets are depicted in Fig. 6.

For each of these datasets, we evaluate the bundle adjustment results by comparing the performance of our approach against squared error loss as well as the standard Huber loss as a fixed kernel. We also compare the results to two adap-

TABLE I

RESULTS ON KITTI ODOMETRY DATASETS [RELATIVE ROT. ERROR IN degrees PER 100 M / RELATIVE TRANS. ERROR IN %]

Approach	Sequence 00	01	02	03	04	05	06	07	08	09	10	Average
Our Approach	1.5/2.8	1.3/**3.8**	0.91/**1.8**	1.5/1.9	0.81/**0.95**	0.97/1.7	0.51/1.1	2.1/2.6	1.3/2.7	0.80/**1.4**	1.3/**1.7**	1.18/**2.03**
Adaptive Kernel (Barron [6])	1.6/3.0	1.2/6.7	0.93/1.9	1.4/1.8	0.82/1.0	0.97/1.8	0.51/1.1	2.2/2.7	1.3/2.8	0.88/**1.4**	1.2/**1.7**	1.19/2.35
Fixed Kernel (Huber)	0.93/**2.1**	1.2/4.5	0.79/2.3	0.7/**1.4**	1.1/49	0.79/1.5	0.64/**0.95**	1.2/**1.8**	0.96/**2.5**	0.78/1.9	0.97/1.8	0.92/6.34
Fixed Kernel (Geman-McClure)	1.8/3.4	1.3/**3.8**	1.0/1.9	1.5/2.0	0.88/1.2	0.98/1.7	0.62/1.3	2.6/3.0	1.5/3.0	1.0/1.6	1.3/1.9	1.32/2.27
Hand-Crafted Outlier Rejection [7]	0.9/**2.1**	1.2/4.0	0.8/2.3	0.7/**1.4**	1.1/11.9	0.8/1.5	0.6/1.0	1.2/**1.8**	1.0/**2.5**	0.8/1.9	1.0/1.8	0.9/2.90

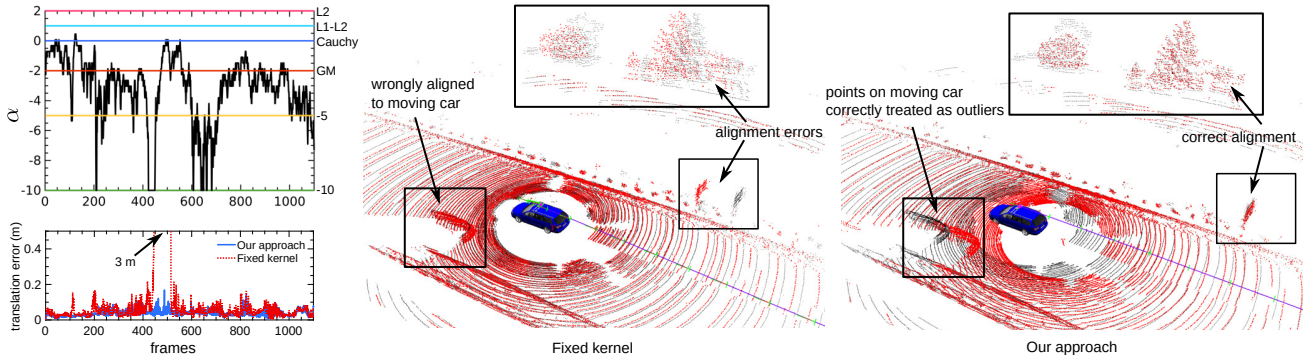


Fig. 5. Registration by ICP and our approach on a challenging dataset for ICP. Top-left: Plot showing α values estimated at each frame based on the residual distribution for KITTI 01 sequence. Bottom-Left: Translation error (in meters) for our approach and fixed kernel ICP. We use Huber kernel as used as the fixed robust kernel in this example. We observe lower α values (stronger outlier down-weighting) for scan matches with outliers arising from dynamic objects in the scene. Middle-right: ICP result from an example frame where ICP converges for adaptive kernel whereas it diverges for a fixed kernel.

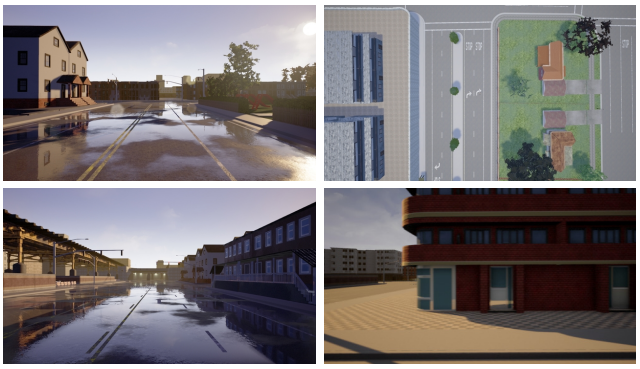


Fig. 6. Examples images from CARLA simulated datasets. Top left: front-looking image mounted on a car from the first dataset. Top right: UAV image in a nadir view looking downwards from the second dataset. Bottom left: front-looking image with strong shadows and reflections from the third dataset. Bottom right: side facing image from the fourth dataset where portions of the image have significant motion blur.

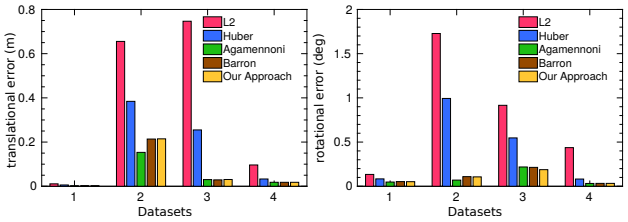


Fig. 7. Performance of different fixed and adaptive robust kernels for bundle adjustment problem on CARLA datasets. We show the translation error (in meters) on the left and rotational errors (in degrees) on the right.

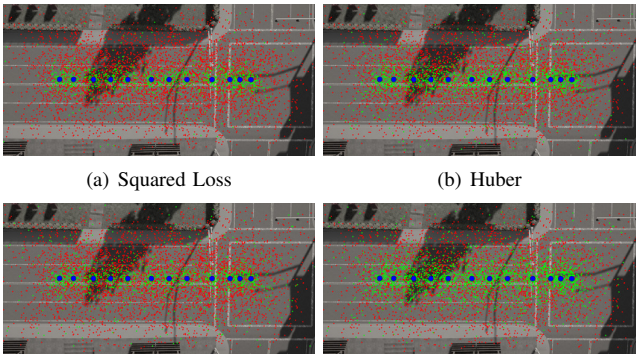
tive kernels which include the original adaptive kernel by Barron [6] as well as a prior work by Agamennoni *et al.* [1] based on a family of robust kernels derived from an elliptical distribution. We re-implemented Agamennoni's [1] approach using the parameters provided for the visual SLAM problem in the paper as it is the closest one to the bundle adjustment problem evaluated here. To evaluate the performance, we compute the accuracy of the camera poses estimated by each of the approached by comparing them against the ground truth poses from the simulator as described by Dickscheid *et al.* [10]. This difference is computed by estimating the optimal transformation between the bundle adjustment result and the ground truth using all 6 DoF pose parameters with the approach by Dickscheid *et al.* [10]. Note that in this analysis, we perform the ground truth comparison only based on the camera poses and do not consider the 3D point as they have been extracted using the SIFT descriptor from the simulated images and thus no ground truth 3D information is available.

We show the results of the bundle adjustment experiment for all the four datasets in Fig. 7. Here, we see that our approach has a lower translation and rotational error than using squared error or the fixed Huber kernel. We obtain a translation and rotational error, which is between 2 to 5 times better as compared to using Huber depending on the dataset. Our approach also shows a similar performance as compared to the adaptive kernel by Barron [6] with a marginal gain on the rotational error some datasets. We believe that the relatively similar performance of the two approaches in these experiments is due to the fact that large negative α's were not needed as often to deal with the outlier distributions in our example datasets. We also obtained similar accuracy results as compared to Agamennoni's approach for most of the datasets, except for second dataset where it outperforms our approach in terms of the translational error. While showing better accuracy results on these datasets, we note that Agamennoni's approach comes at computational overhead. This approach has a two step process to determine the best robust kernel given a residual distribution. It first computes the best hyper-parameter value (similar to α in this paper) for each of the kernels which can be derived from an elliptical distribution (five such kernels are provided in the paper), and then a second step to performed by computing the Kullback-Leibler divergence to determine the best kernel.

The last experiment in this section is designed to analyze the influence of our approach on the convergence properties of BA. A large basin of convergence is important for robust operation, especially for BA due to the missing range information with the image data. We initialized the bundle adjustment procedure by adding significant noise to the initial camera poses, i.e., $\sigma \in [0.1\,\text{m}, 5\,\text{m}]$ to the ground truth poses of the camera. The noise in the camera poses is propagated to the 3D points during the forward intersection step. We sample 20 instances of each noise level (500 instances in total) and run the bundle adjustment for our approach, using squared loss, the Geman-McClure as well as the Huber kernel. We consider the adjustment to have converged if the final RMS error of the camera center is less than 1 cm from the true position. We visualize the results in Fig. 8 where the poses from which the BA has converged are shown in green and the ones that caused divergence in red. We can clearly see that our approach has a larger convergence radius as the green points are spread over a larger area compared to the squared loss or fixed Huber or Geman-McClure kernel. We obtain successful convergence rate for 45% of all instances for our approach against 24.8% for squared loss, 33% for Huber, and 28.2% for Geman-McClure. Overall, the experiments suggest that by using our approach, we can obtain a more accurate estimate and have a larger convergence area as compared to a fixed kernel. Thus, our approach is an effective and useful approach for optimization in bundle adjustment problems.

C. Effect of the Truncation Parameter

In this experiment, our goal is to analyze the effect of the truncation parameter τ on the state estimation task. The parameter τ is used to define the integral limits for the partition function in Eq. (11). By choosing a value of τ from the set $\{10c, 20c, 50c, 100c\}$, we define a truncated partition function $\tilde{Z}(.)$ for each value of τ. This results in multiple robust kernels $\tilde{\rho}_a(.)$ which are then used for solving the bundle adjustment problem on the four datasets as described in Sec. IV-B. The results for these experiments are shown in Fig. 9. We observe that the accuracy results using different values of τ is similar for all the datasets. The maximum difference in the translation error is about 5%, and the difference in rotational error is 8% using different values for τ. We also note that during the experiments, the α estimated in Step 1 of Alg. 1 of the optimization process belongs to a similar range of α values for each of the τ used. These results in this experiment suggests that the approach is not critically sensitive to the value of the truncation parameter used.



(a) Squared Loss (b) Huber

(c) Geman-McClure (d) Our Approach

Fig. 8. Convergence analysis for BA. Green points indicate poses for which BA converged, whereas red points indicate divergence. The blue circles represent the ground truth camera poses.

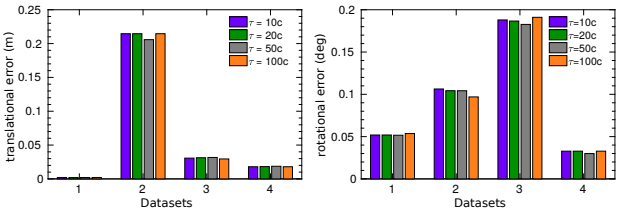


Fig. 9. Effect of the truncation parameter τ on the accuracy on the bundle adjustment task.

V. Limitations and Future Work

In this paper, we have mainly focused on extending the generalized robust kernel formulation by Barron [6] for its use in common state estimation problems in robotics. There are several interesting directions in which this work can be extended. We see two such aspects, which offer space for further investigations:

Adapting scale parameter c: In our current implementation, we use a fixed scale parameter c, which is set based on the measurement noise for an inlier observation. In principle, both, the shape parameter α and the scale parameter c can be adapted simultaneously. In practice, however, learning

both these parameters jointly becomes tricky as each of them can individually find *some explanation* of the residual distribution. Typically, a large change in the residuals can be captured by a rather small change in c without affecting α at all. This results in a situation where the shape parameter α cannot be estimated properly. This suggests that a different scheme needs to be developed to adapt the parameter α and the scale parameter c jointly to respond to a change in the outlier distribution.

Use of multiple α parameters: We use a single α value to capture the outlier distribution for the overall optimization problem in our example applications. For example, in the ICP case, we estimate only one α value for all the points for a scan pair. This means, the α value is adapted over time (between different scan pairs), however, is the same for all the points in the scan pair at one timestep t. By estimating a different α value for groups of points belonging to different objects (e.g. vegetation, road, vehicles etc.) in the scan, we can model an inlier/outlier weight for each group of points in addition to time. Similarly, in the bundle adjustment application, we adapt α at each iteration of the optimization procedure. Here, one could estimate different α values for each sub-block of the optimization problem, where each of the sub-block could consist of the camera poses and measurements of a particular location in the environment.

Use of an alternative regularization term for the truncated loss: The truncated partition function $\tilde{Z}(\alpha)$ can be seen as playing the role of a regularizer for α in Eq. (12). An alternative approach to regularizing α would be to replace the truncated integral term $\tilde{Z}(\alpha)$ with a suitable regularizer that is defined for any α in the range $[-\infty, 2]$. This is possible as there is no strict requirement that a robust loss $\rho(\cdot)$ must correspond to the negative log-likelihood of a probability distribution function as we have defined in this paper. This opens up the possibility of designing regularization terms that have potentially better outlier rejection properties, and provides an interesting direction for future work.

VI. Conclusion

In this paper, we presented a novel approach to robust optimization that avoids the need to commit to a fixed robust kernel and potentially has a broad application area for state estimation in robotics. We proposed the use of a generalized robust kernel that can adapt its shape with an additional parameter that has recently been proposed by Barron [6]. We modified the original formulation, which enables us to use the adaptive kernel also in situations with strong outliers. We integrated our adaptive kernel into and tested it for two popular state estimation problems in robotics, namely ICP and bundle adjustment. The experiments showcase that we are better or on-par with fixed kernels such as Huber or Geman-McClure but do not require hand-crafted outlier rejection schemes in the case of ICP and can increase the radius of convergence for bundle adjustment. We believe that several other problems in robotics, which rely on robust least-squares estimation, can benefit from our proposed approach.

References

[1] G. Agamennoni, P.T. Furgale, and R. Siegwart. Self-Tuning M-Estimators. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[2] P. Agarwal, W. Burgard, and C. Stachniss. A Survey of Geodetic Approaches to Mapping and the Relationship to Graph-Based SLAM. *IEEE Robotics and Automation Magazine (RAM)*, 2014.

[3] P. Agarwal, G.D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust Map Optimization using Dynamic Covariance Scaling. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[4] S. Agarwal, K. Mierle, and Others. Ceres Solver. http://ceres-solver.org, 2010.

[5] P. Babin, P. Giguere, and F. Pomerleau. Analysis of robust functions for registration algorithms. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[6] J. T. Barron. A General and Adaptive Robust Loss Function. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[8] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Intl. Journal of Computer Vision (IJCV)*, 19(1):57–91, 1996.

[9] M. Bosse, G. Agamennoni, and I. Gilitschenski. Robust estimation and applications in robotics. *Foundations and Trends in Robotics*, 4(4):225–269, 2016.

[10] T. Dickscheid, T. Läbe, and W. Förstner. Benchmarking automatic bundle adjustment results. In *Cong. of the Intl. Society for Photogrammetry and Remote Sensing (ISPRS)*, 2008.

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proc. of the Conf. on Robot Learning*, 2017.

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

[13] P. J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[14] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics (TRO)*, 24(6), 2008.

[15] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3607–3613, 2011.

[16] T. Läbe and W. Förstner. Automatic relative orientation of images. In *Proc. of the Turkish-German Joint Geodetic Days*, 2006.

[17] P. Lajoie, S. Hu, G. Beltrame, and L. Carlone. Modeling perceptual aliasing in SLAM via discrete-continuous graphical models. *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[18] Y. Latif, C. Cadena, and J. Neira. Robust loop closing over time. *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[19] K. MacTavish and T. D. Barfoot. At all costs: A comparison of robust cost functions for camera correspondence outliers. In *Proc. of the Conf. on Computer and Robot Vision*, pages 62–69, 2015.

[20] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 26(6):756–770, 2004.

[21] J. Schneider, F. Schindler, T. Läbe, and W. Förstner. Bundle adjustment for multi-camera systems with points at infinity. In *ISPRS Annals of Photogrammetry, Rem. Sens. and Spatial Inf. Sci.*, volume I-3, 2012.

[22] C. Stachniss, J. Leonard, and S. Thrun. *Springer Handbook of Robotics, 2nd edition*, chapter Chapt. 46: Simultaneous Localization and Mapping. Springer Verlag, 2016.

[23] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph slam. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1879–1884, 2012.

[24] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):1127–1134, 2020.

[25] H. Yang, J. Shi, and L. Carlone. TEASER: Fast and Certifiable Point Cloud Registration. *arXiv preprint arXiv: 2001.07715*, 2020.

[26] C. Zach. Robust bundle adjustment revisited. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 772–787, 2014.

[27] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing*, 15:59–76, 1997.

[E] X. Chen, T. Läbe, A. Milioto, *et al.*, "OverlapNet: A Siamese Network for Computing LiDAR Scan Similarity with Applications to Loop Closing and Localization," *Autonomous Robots (AURO)*, vol. 46, pp. 61–81, 2021. DOI: 10.1007/s10514-021-09999-0

Autonomous Robots
https://doi.org/10.1007/s10514-021-09999-0

OverlapNet: a siamese network for computing LiDAR scan similarity with applications to loop closing and localization

Xieyuanli Chen[1] · Thomas Läbe[1] · Andres Milioto[1] · Timo Röhling[1] · Jens Behley[1] · Cyrill Stachniss[1]

Received: 30 January 2021 / Accepted: 29 June 2021
© The Author(s) 2021

Abstract
Localization and mapping are key capabilities of autonomous systems. In this paper, we propose a modified Siamese network to estimate the similarity between pairs of LiDAR scans recorded by autonomous cars. This can be used to address both, loop closing for SLAM and global localization. Our approach utilizes a deep neural network exploiting different cues generated from LiDAR data. It estimates the similarity between pairs of scans using the concept of image overlap generalized to range images and furthermore provides a relative yaw angle estimate. Based on such predictions, our method is able to detect loop closures in a SLAM system or to globally localize in a given map. For loop closure detection, we use the overlap prediction as the similarity measurement to find loop closure candidates and integrate the candidate selection into an existing SLAM system to improve the mapping performance. For global localization, we propose a novel observation model using the predictions provided by OverlapNet and integrate it into a Monte-Carlo localization framework. We evaluate our approach on multiple datasets collected using different LiDAR scanners in various environments. The experimental results show that our method can effectively detect loop closures surpassing the detection performance of state-of-the-art methods and that it generalizes well to different environments. Furthermore, our method reliably localizes a vehicle in typical urban environments globally using LiDAR data collected in different seasons.

Keywords Loop closing · Localization · SLAM

1 Introduction

Over the past decades, Light Detection and Ranging (LiDAR) sensors became key components of the sensor suite of autonomous vehicles that allows to perceive and navigate the world. Especially mapping and localization systems can leverage the geometric information provided by LiDAR sensors covering the 360° surroundings of the vehicle. Accurate odometry estimation allows to build locally consistent maps, and a reliable loop closure detection enables SLAM systems to correct accumulated drift and to build globally consistent maps. These globally consistent maps can then be used to localize the vehicle. Both tasks, the loop closure detection and localization, need to determine the similarity between pairs of laser range scans. The similarity of laser range scans of the same scene should be high regardless of the sensor locations used to capture them, but should be low if the sensor observes different parts of the environment.

In this article, we show that our prior work (Chen et al. 2020a, b) can be formulated as a general approach, which exploits a neural network to estimate the similarity between laser range scans produced by a rotating 3D LiDAR sensor mounted on a wheeled robot or autonomous vehicle to tackle loop closing and global localization as shown in Fig. 1. The proposed network predicts both, a so-called overlap defined on range images corresponding to the similarity between the 3D LiDAR scans and a yaw angle offset between the two scans. The concept of overlap is used in photogrammetry to describe the configuration of image blocks, e.g., of aerial surveys, and we extend this to LiDAR range images. It is a useful tool for estimating the similarity between pairs of LiDAR scans, which can be used to find loop closure candidates and

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob, and by the Chinese Scholarship Committee.

This is one of the several papers published in *Autonomous Robots* comprising the Special Issue on Robotics: Science and Systems 2020.

✉ Xieyuanli Chen
xieyuanli.chen@igg.uni-bonn.de

[1] University of Bonn, Bonn, Germany

Published online: 05 August 2021

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

estimate the likelihood of an observation given the sensor position. The yaw estimate can serve as an initial guess for a subsequent application of the iterative closest point (ICP) algorithm (Besl and McKay 1992) to determine the precise relative pose between scans to derive loop closures constraints for pose graph-based optimization in SLAM. During localization, the yaw estimate can be used to estimate the heading likelihood of the current observation.

The main contribution of this article is a deep neural network that exploits different types of information generated from LiDAR scans to provide overlap and relative yaw angle estimates between pairs of 3D scans. This information includes depth, normals, and intensity/remission values. We additionally exploit a probability distribution over semantic classes, which can be computed for each laser beam. Our approach relies on a spherical projection of LiDAR scans rather than the raw point clouds, which makes the proposed OverlapNet comparably lightweight. We furthermore test our method in different applications for autonomous vehicles and mobile robots. We integrate our OverlapNet into a state-of-the-art SLAM system (Behley and Stachniss 2018) for loop closure detection and evaluate its performance also with respect to generalization to different environments. We also integrate our OverlapNet as the observation model in a Monte-Carlo localization (MCL) approach for updating the importance weights of the particles.

To test LiDAR-based loop closing, we train the proposed OverlapNet on parts of the KITTI odometry dataset and evaluate it on unseen data. We thoroughly evaluate our approach, provide ablation studies using different modalities, and test the integrated SLAM system in an online manner. Furthermore, we provide results for the Ford campus dataset, which was recorded using a different sensor setup in a different country and a differently structured environment. The experimental results suggest that our method outperforms other state-of-the-art baseline methods and is able to generalize well to unseen environments.

To test LiDAR-based global localization, a dataset has been collected in different seasons with multiple sequences repeatedly exploring the same crowded urban area using our own car setup. Based on our novel observation model, MCL achieves global localization using 3D LiDAR scans over different seasons with a comparably small number of particles.

In sum, our approach is able to (i) predict the overlap and relative yaw angle between pairs of LiDAR scans by exploiting multiple cues without using relative poses, (ii) combine odometry information with overlap predictions to detect correct loop closure candidates, (iii) improve the overall pose estimation results in a state-of-the-art SLAM system yielding more globally consistent maps, (iv) initialize ICP using the OverlapNet predictions yielding correct scan matching results, (v) build a novel observation model and achieve global localization. The source code of the implementation of our approach is publicly available (see Sect. 6.1).

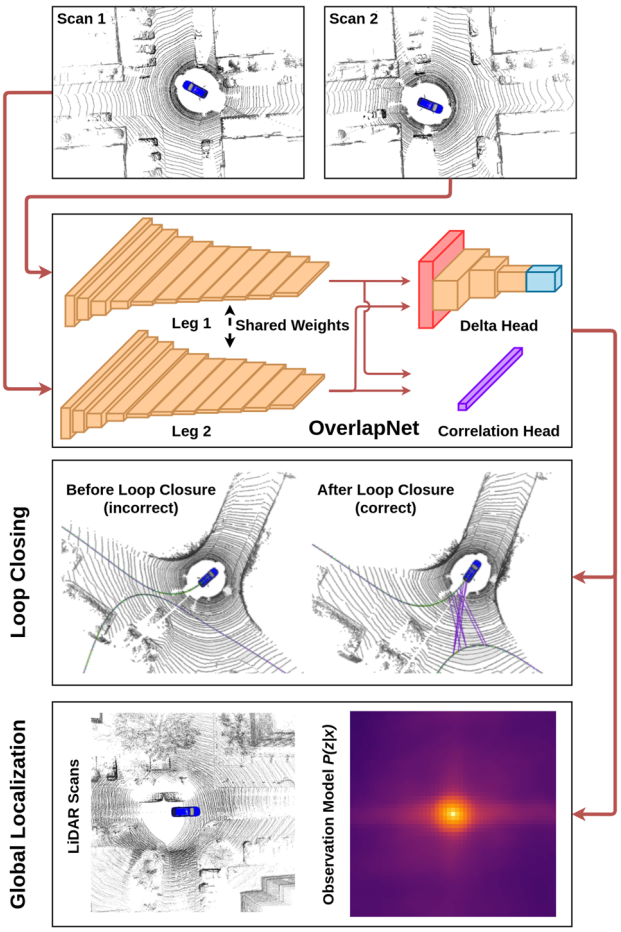


Fig. 1 Overview: We use a siamese network with two LiDAR scans as input. The result of OverlapNet consisting of the overlap percentage and yaw angle offset can be used for loop closing and/or global localization

2 Related work

Various sensor modalities have been used in autonomous cars and wheeled robots for loop closing and localization (Thrun et al. 2005; Stachniss et al. 2016). Here, we mainly concentrate on related work addressing 3D LiDAR-based approaches. Since our method address both loop closing and global localization, we discuss the related work in both topics separately and concentrate on LiDAR-based approaches.

2.1 Loop closing for SLAM

Loop closing is the problem of correctly identifying that a robot has returned to a previously visited place and is a key component in SLAM systems. For example, Steder et al.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

(2011) propose a place recognition system operating on range images generated from 3D LiDAR data that uses a combination of bag-of-words and NARF features (Steder et al. 2010) for relative poses estimation. Röhling et al. (2015) present an efficient method for detecting loop closures through the use of similarity measures on histograms extracted from 3D LiDAR scans of an autonomous ground vehicle and an optimization-based SLAM system. The work by He et al. (2016) presents M2DP, which projects a LiDAR scan into multiple reference planes to generate a descriptor using a density signature of points in each plane. Besides using pure geometric information, there are also methods (Cop et al. 2018; Guo et al. 2019) exploiting the remission information, i.e., how well LiDAR beams are reflected by a surface, to create descriptors for localization and loop closure detection with 3D LiDAR data. Other researchers link LiDAR observations with background knowledge about the environment to create global constraints, similar to a loop closure (Vysotska and Stachniss 2017).

Recently, deep learning-based methods that yield features in an end-to-end fashion have been proposed for loop closing to overcome the need for hand-crafting features. For example, Dubé et al. (2017) investigated an approach that matches segments extracted from a scan to find loop closures via segment-based features. A geometric test via RANS- AC is used to verify a potential loop closure identified by the matching procedure. Based on such segments, Cramariuc et al. (2018) train a CNN to extract descriptors from segments and use it to retrieve near-by place candidates. Uy and Lee (2018) propose PointNetVLAD which is a deep network used to generate global descriptors for 3D point clouds. It aggregates multiple LiDAR scans and end-to-end encodes a global descriptor to tackle the retrieval and place recognition task. Yin et al. (2019) develop LocNet, which uses semi-handcrafted features learning based on a siamese network to solve LiDAR-based place recognition. Schaupp et al. (2019) propose a system called OREOS for place recognition. They use a convolutional neural network to extract compact descriptors from LiDAR scans and use the features to retrieve near-by place candidates from a map and to estimate the yaw discrepancy. Most recently, Zaganidis et al. (2019) proposed a normal distributions transform histogram-based loop closure detection method, which is assisted by semantic information. Kong et al. (2020) also use semantic graphs for place recognition for 3D point clouds. Their network is capable of capturing topological and semantic information from the point cloud and also achieves rotational invariance.

Contrary to the above-mentioned methods, our method exploits multiple types of information extracted from 3D LiDAR scans, including depth, normal information, intensity/remission and probabilities of semantic classes generated by a semantic segmentation system developed and reported by Milioto et al. (2019) and Milioto and Stachniss (2019). Our method furthermore uses a siamese network to learn features and yield predictions end-to-end, which directly provides estimates for overlap and the relative yaw angle between pairs of LiDAR scans. Based on that, our method can be used not only for detecting loop closure candidates but also provides an estimate of the matching quality in terms of the overlap.

2.2 Global localization

Besides loop closing, our method can be further used for LiDAR-based localization. Localization is a classical topic in robotics (Dellaert et al. 1999; Thrun et al. 2005). For localization given a map, one often distinguishes between pose tracking and global localization. In pose tracking, an approach needs to locally localize the vehicle from a known pose and the pose is updated over time. In global localization, no pose prior is available and an approach needs to localize the vehicle globally. In this work, we address global localization using 3D LIDAR sensors without assuming any pose prior from GPS or other sensors.

A popular traditional framework for robot localization relies on probabilistic state estimation techniques is Monte-Carlo localization (Dellaert et al. 1999; Fox et al. 1999), which uses a particle filter to estimate the pose of the robot.

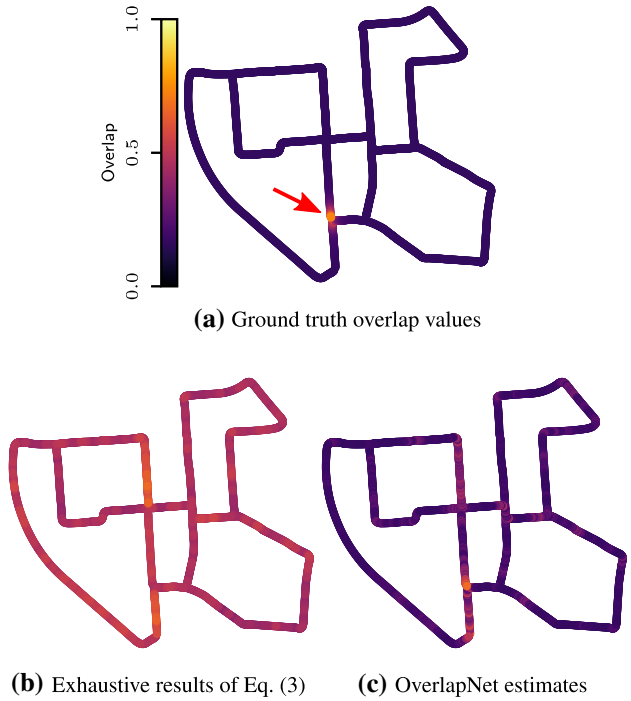
In the context of autonomous cars, there are many approaches building and using high-definition (HD) maps for localization. For example, Levinson et al. (2007) utilize GPS, IMU, and LiDAR data to build HD maps for localization. They generate a 2D surface image of ground reflectivity in the infrared spectrum and define an observation model that uses these intensities. Wolcott and Eustice (2015) propose a new scan matching algorithm that leverages Gaussian mixture maps to exploit the structure in the environment. The uncertainty in intensity values has been handled by building a prior HD map. Barsan et al. (2018) use a fully convolutional neural network to perform online-to-map matching for improving the robustness to dynamic objects and eliminating the need for LiDAR intensity calibration. Their approach requires a good GPS prior to achieve good performance. Based on this approach, Wei et al. (2019) proposed a learning-based compression method for HD maps, which compresses the intermediate representations of the neural network while retaining important information for downstream tasks. Merfels and Stachniss (2016) present an efficient chain graph-like pose graph for vehicle localization exploiting graph optimization techniques and different sensing modalities. Based on that work, Wilbers et al. (2019) propose a LiDAR-based localization system performing a combination of local data association between laser scans and HD maps, temporal data association smoothing, and a map matching approach for robustification. To compress the map for online localization,

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Wiesmann et al. (2021) propose a deep network to compress and decompress the point cloud maps. Recently, Vizzo et al. (2021) also propose a light-weighted mesh mapping algorithm using Poisson surface reconstruction for LiDAR scans and later using the mesh maps for LiDAR-based localization (Chen et al. 2021).

Other approaches aim at performing LiDAR-based place recognition to initialize localization. For example, Kim et al. (2019) transform point clouds into scan context images and train a CNN based on such images. They generate scan context images for both the current frame and all grid cells of the map and compare them to estimate the current location as the cell presenting the largest score. Yin et al. (2019) propose a siamese network to first generate fingerprints for LiDAR-based place recognition and then use iterative closest points to estimate the metric poses. Cop et al. (2018) propose a descriptor for LiDAR scans based on intensity information. Using this descriptor, they first perform place recognition to find a coarse location of the robot, eliminate inconsistent matches using RANSAC (Fischler and Bolles 1981), and then refine the estimated transformation using iterative closest points.

Recently, several approaches exploiting semantic information for 3D LiDAR localization have been proposed. Zhang et al. (2018) utilize both ground reflectivity features and vertical features for localizing autonomous cars in rainy conditions. Both a histogram filter and a particle filter are integrated to estimate the posterior distributions of the vehicle poses. Using a similar idea, Ma et al. (2019) combine semantic information such as lanes and traffic signs in a Bayesian filtering framework to achieve accurate and robust localization within sparse HD maps. Yan et al. (2019) exploit buildings and intersections information from a LiDAR-based semantic segmentation system to localize on OpenStreetMap data using 4-bit semantic descriptors. Not targeting tiny descriptors but the exploitation of prior knowledge for localization has also been tackled by Silver and Stentz (2011). They use satellite imagery of outdoor scenes and propose a sensor model to link the observation of the robot with the aerial imagery. Schaefer et al. (2019) detect and extract pole landmarks from 3D LiDAR scans for long-term urban vehicle localization. The proposed pole detector considers both the laser ray endpoints and the free space in between the laser sensor and the endpoints, which demonstrates reliable and accurate localization performance. Tinchev et al. (2019) propose a learning-based method to match segments of trees and localize in both urban and natural environments. The approach learns a light feature space representation which can be deployed using only a CPU. In our previous work (Chen et al. 2019), we also exploit semantic information to improve the localization and mapping results by detecting and removing dynamic objects. Sun et al. (2020) use a deep-probabilistic model to accelerate the initialization of the Monte Carlo localization and achieve a fast localization in outdoor campus environments. This hybrid LiDAR-based localization approach integrates a learning-based with a Markov-filter-based method, which makes it possible to effectively and efficiently provide global localization results.



(a) Ground truth overlap values

(b) Exhaustive results of Eq. (3)

(c) OverlapNet estimates

Fig. 2 Overlap estimations of one frame to all others. **a** The red arrow points out the position of the query scan. **b** If we directly use Eq. (3) to estimate the overlap between two LiDAR scans without knowing the accurate relative poses, it is hard to decide which pairs of scans are true loop closures, since most evaluations of Eq. (3) show high values. **c** In contrast, our OverlapNet can predict the overlaps between two LiDAR scans without the relative transformation between the scans such that only the correct location get a high overlap

Different to the above discussed methods (Barsan et al. 2018; Ma et al. 2019; Wei et al. 2019), which use GPS as prior for localization, our method only exploits LiDAR information to achieve global localization without using any GPS information. Moreover, our approach uses range scans without explicitly exploiting semantics or extracting landmarks. In contrast to approaches that perform place recognition (Kim et al. 2019; Cop et al. 2018; Yin et al. 2019), our approach relies on convolutional neural networks to predict the overlap between range scans and their yaw angle offset and we exploit this information as an observation model for Monte-Carlo localization.

3 OverlapNet

The idea of overlap that we are using here has its origin in the photogrammetry and computer vision community (Hussain

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

and Bethel 2004). To successfully match two images and to calculate their relative orientation, the images must overlap. This can be quantified by defining the overlap value as the percentage of pixels in the first image, which can successfully be projected back into the second image without occlusion. Note that this measure is not symmetric. If there is a large scale difference of the image pair, e.g., one image shows a wall and the other shows many buildings around that wall, the overlap percentage for the first to the second image can be large and from the second to the first image low. In this paper, we use the idea of overlap for range images and exploit the range information explicitly.

3.1 Definition of the overlap between LiDAR scans

We use spherical projections of LiDAR scans as input data, which is often used to speed up computations (Bogoslavskyi and Stachniss 2016; Behley and Stachniss 2018; Chen et al. 2019). We project the point cloud $\mathcal{P} = \{\mathbf{p}_i\}, i \in \{1, \ldots, N\}$ to a so-called vertex map $\mathcal{V} : \mathbb{R}^2 \mapsto \mathbb{R}^3$, where each pixel is mapped to the nearest 3D point. Each point $\mathbf{p}_i = (x, y, z)$ is converted via the function $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates and finally to image coordinates (u, v) by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\pi^{-1}\right] w \\ \left[1 - \left(\arcsin(zr^{-1}) + \mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right] h \end{pmatrix}, \tag{1}$$

where $r = ||\mathbf{p}||_2$ is the range, $\mathrm{f} = \mathrm{f}_{\mathrm{up}} + \mathrm{f}_{\mathrm{down}}$ is the vertical field-of-view of the sensor, and w, h are the width and height of the resulting vertex map $\mathcal{V}$.

For the LiDAR scans $\mathcal{P}_1$ and $\mathcal{P}_2$, we generate the corresponding vertex maps $\mathcal{V}_1, \mathcal{V}_2$. We denote the sensor-centered coordinate frame at time step t as C_t. Each pixel in coordinate frame C_t is associated with the world frame W by a pose $\mathbf{T}_{WC_t} \in \mathbb{R}^{4\times4}$. Given the poses $\mathbf{T}_{WC_1}$ and $\mathbf{T}_{WC_2}$, we can reproject scan $\mathcal{P}_1$ into the other's vertex map $\mathcal{V}_2$ and generate a reprojected vertex map $\mathcal{V}_1'$:

$$\mathcal{V}_1' = \Pi\left(\mathbf{T}_{WC_1}^{-1}\mathbf{T}_{WC_2}\mathcal{P}_1\right). \tag{2}$$

We calculate the absolute difference of all corresponding pixels in $\mathcal{V}_1'$ and $\mathcal{V}_2$, considering only those pixels that correspond to valid range readings in both range images. The overlap is then calculated as the percentage of all differences in a certain distance ϵ relative to all valid entries, i.e., the overlap of two LiDAR scans $O_{C_1C_2}$ is defined as:

$$O_{C_1C_2} = \frac{\sum_{(u,v)} \mathbb{I}\left\{ ||\mathcal{V}_1'(u, v) - \mathcal{V}_2(u, v)|| \leq \epsilon \right\}}{\min\left(\mathrm{valid}(\mathcal{V}_1'), \mathrm{valid}(\mathcal{V}_2)\right)}, \tag{3}$$

where $\mathbb{I}\{a\} = 1$ if a is true and 0 otherwise. The function valid$(\mathcal{V})$ refers to the number of valid pixels in $\mathcal{V}$, since not all pixel might have a valid LiDAR measurement associated after the projection.

We use Eq. (3) only for *creating training data*, i.e., only positive examples of correct loop closures get a non-zero overlap assigned using the relative poses between scans, as shown in Fig. 2. Thus, the proposed approach can learn not only high overlap values for nearby scans, but also to estimate high overlap values for scans that correspond to the same part of the environment and lower overlap values for different parts. During test time, no (ground-truth) poses are available and not required by our proposed approach. When performing loop closure detection for online SLAM, the approximate relative poses computed by the SLAM system before loop closure are not accurate enough to calculate suitable overlaps by using Eq. (3) because of accumulated drift.

To verify that the estimated overlap captures more than just the similarity of the raw scans, we tried directly calculating overlaps using Eq. (3) assuming the relative pose to be the identity and applying different orientations, e.g., every 30 degrees rotation around the vertical axis, and using the maximum over all these overlaps as an estimate. Figure 2 shows the estimated overlaps for all scans using a query scan produced by this method and the result of the estimated overlap for all scans using OverlapNet. We leave out the 100 most recent scans because they will not be loop closure candidates. In the case of the exhaustive approach based on Eq. (3), many scans which are far away get high overlap values, which makes this method unsuitable for loop closing or global localization. Our approach, however, correctly identifies the correct similarity as it produces a highly distinctive peak around the correct location.

3.2 Overlap network architecture

A visual overview of our proposed OverlapNet is depicted in Fig. 3. We exploit multiple cues, which can be generated from a single LiDAR scan, including depth, normal, intensity, and semantic class probability information. The depth information is stored in the range map $\mathcal{R}$, which defines one input channel. We use neighborhood information of the vertex map to generate a normal map $\mathcal{N}$, which gives three channels encoding the normal coordinates. We directly obtain the intensity information, also called remission, from the sensor and represent the intensity information as a one-channel intensity map $\mathcal{I}$. The point-wise semantic class probabilities are computed using RangeNet++ by Milioto et al. (2019) and we represent them as a semantic map $\mathcal{S}$. The semantic segmentation network delivers probabilities for 20 different classes. For efficiency reasons, we reduce the 20 dimensional RangeNet++ output to a compressed 3 dimensional vector using principal component analysis. All the information is combined as the input of the OverlapNet with the size of $64 \times 900 \times 8$.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

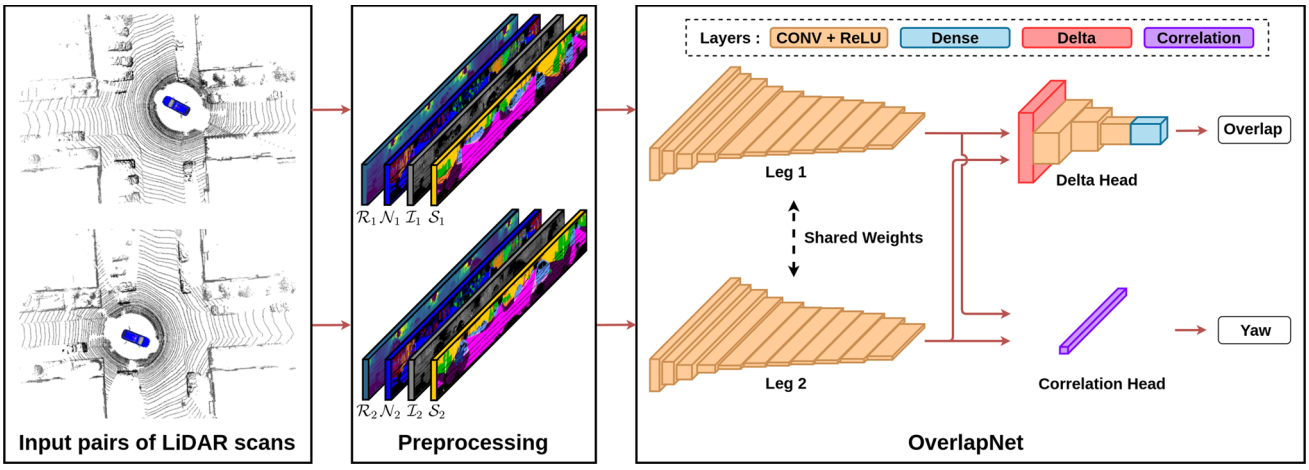


Fig. 3 Pipeline overview of our proposed approach. The left-hand side shows the preprocessing of the input data which exploits multiple cues generated from a single LiDAR scan, including range $\mathcal{R}$, normal $\mathcal{N}$, intensity $\mathcal{I}$, and semantic class probability $\mathcal{S}$ information. The right-hand side shows the proposed OverlapNet which consists of two legs sharing weights and the two heads use the same pair of feature volumes generated by the two legs. The outputs are the overlap and relative yaw angle between two LiDAR scans

Our proposed OverlapNet is a siamese network architecture (Bromley et al. 1993), which consists of two legs sharing weights and two heads that use the same pair of feature volumes generated by the two legs. The trainable layers are listed in Table 1.

3.2.1 Legs

The proposed OverlapNet has two legs, which have the same architecture and share weights. Each leg is a fully convolutional network consisting of 11 convolutional layers. This architecture is quite lightweight, i.e., it only consists of 1.8 million parameters, and generates feature volumes of size $1 \times 360 \times 128$.

Note that our range images are cyclic projections and that a change in the yaw angle of the vehicle results in a cyclic column shift of the range image. Thus, the single row in the feature volume can represent a relative yaw angle estimate because a yaw angle rotation results in a pure horizontal shift of the input maps. As the fully convolutional network is translation-equivariant, the feature volume will be shifted horizontally in the same manner. The number of columns of the feature volume defines the resolution of the yaw estimation, which is 1° in the case of our leg architecture.

3.2.2 Delta head

The delta head is designed to estimate the overlap between two scans. It consists of a delta layer, three convolutional layers, and one fully connected layer.

The delta layer, which is shown in Fig. 4, computes all possible absolute differences of all pixels. It takes the output feature volumes $\mathbf{L}^l \in \mathbb{R}^{H \times W \times C}$ from the two legs $l \in \{0, 1\}$ as input. These are stacked in a tiled tensor $\mathbf{T}^l \in \mathbb{R}^{HW \times HW \times C}$ as follows:

$$\mathbf{T}^0(iW + j, k, c) = \mathbf{L}^0(i, j, c), \tag{4}$$

$$\mathbf{T}^1(k, iW + j, c) = \mathbf{L}^1(i, j, c), \tag{5}$$

with $k = \{0, \ldots, HW - 1\}$, $i = \{0, \ldots, H - 1\}$ and $j = \{0, \ldots, W - 1\}$.

Note that $\mathbf{T}^1$ is transposed with respect to $\mathbf{T}^0$, as depicted in the middle of Fig. 4. After that, all differences are calculated by element-wise absolute differences between $\mathbf{T}^0$ and $\mathbf{T}^1$.

By using the delta layer, we can obtain a representation of the latent difference information, which can be later exploited by the convolutional and fully-connected layers to estimate the overlap. Different overlaps induce different patterns in the output of the delta layer.

3.2.3 Correlation head

The correlation head (Nagashima et al. 2007) is designed to estimate the yaw angle between two scans using the feature volumes of the two legs. To perform the cross-correlation, we first pad horizontally one feature volume by copying the same values (as the range images are cyclic projections around the yaw angle). This doubles the size of the feature volume. We then use the other feature volume as a kernel that is shifted over the first feature volume generating a 1D output of size 360. The argmax of these correlation values serves as the estimate of the relative yaw angle of the two input scans with a 1° resolution.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 1 Layers of our network architecture

	Operator	Stride	Filters	Size	Output Shape
Legs	Conv2D	(2, 2)	16	(5, 15)	30 × 443 × 16
	Conv2D	(2, 1)	32	(3, 15)	14 × 429 × 32
	Conv2D	(2, 1)	64	(3, 15)	6 × 415 × 64
	Conv2D	(2, 1)	64	(3, 12)	2 × 404 × 64
	Conv2D	(2, 1)	128	(2, 9)	1 × 396 × 128
	Conv2D	(1, 1)	128	(1, 9)	1 × 388 × 128
	Conv2D	(1, 1)	128	(1, 9)	1 × 380 × 128
	Conv2D	(1, 1)	128	(1, 9)	1 × 372 × 128
	Conv2D	(1, 1)	128	(1, 7)	1 × 366 × 128
	Conv2D	(1, 1)	128	(1, 5)	1 × 362 × 128
	Conv2D	(1, 1)	128	(1, 3)	1 × 360 × 128
Delta Head	Conv2D	(1, 15)	64	(1, 15)	360 × 24 × 64
	Conv2D	(15, 1)	128	(15, 1)	24 × 24 × 128
	Conv2D	(1, 1)	256	(3, 3)	22 × 22 × 256
	Dense	–	–	–	1

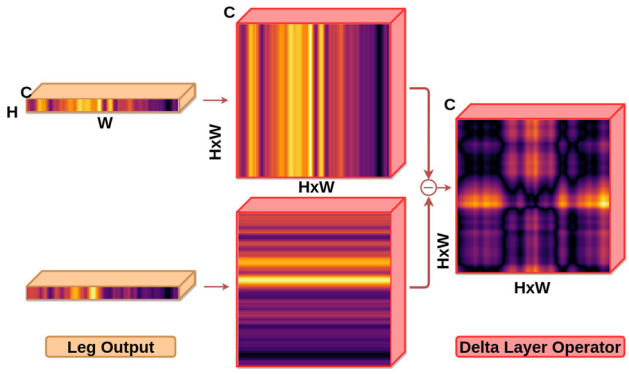


Fig. 4 The delta layer: Computation of pairwise differences is efficiently performed by concatenating the feature volumes and transposition of one concatenated feature volume

3.3 Loss function

We train our OverlapNet end-to-end to estimate the overlap and the relative yaw angle between two LiDAR scans at the same time. Typically, to train a neural network one needs a large amount of manually labeled ground truth data. In our case, this is (I_1, I_2, Y_O, Y_Y), where I_1, I_2 are two inputs and Y_O, Y_Y are the ground truth overlaps and the ground truth yaw angles respectively. We are, however, able to generate the input and the ground truth without any manual effort in a fully automated fashion given a dataset with pose information. From given poses (e.g., obtained using a GPS+IMU combination), we can calculate the ground truth overlap and relative yaw angles directly. We denote the legs part network with trainable weights as $f_L(\cdot)$, the delta head as $f_D(\cdot)$ and the correlation head as $f_C(\cdot)$.

For training, we combine the loss $L_O(\cdot)$ for the overlap and the loss $L_Y(\cdot)$ for the yaw angle using a weight α:

$$L\left(I_1, I_2, Y_O, Y_Y\right) = L_O\left(I_1, I_2, Y_O\right) + \alpha L_Y\left(I_1, I_2, Y_Y\right). \quad (6)$$

We treat the overlap estimation as a regression problem and use a weighted absolute difference of ground truth Y_O and network output $\hat{Y}_O = f_D\left(f_L\left(I_1\right), f_L\left(I_2\right)\right)$ as the loss function. For weighting, we use a scaled sigmoid function:

$$L_O\left(I_1, I_2, Y_O\right) = \text{sigmoid}\left(s\left(\left|\hat{Y}_O - Y_O\right| + a\right) - b\right), \quad (7)$$

with $\text{sigmoid}(v) = (1+\exp(-v))^{-1}$, the variables a, b being offsets, and s being a scaling factor.

For the yaw angle estimation, we use a lightweight representation of the correlation head output, which leads to a one-dimensional vector of size 360. We take the index of the maximum, the argmax, as the estimate of the relative angle in degrees. As the argmax is not differentiable, we cannot treat this as a simple regression problem. The yaw angle estimation, however, can be regarded as a binary classification problem that decides for every entry of the head output whether it is the correct angle or not. Therefore, we use the binary cross-entropy loss given by

$$L_Y\left(I_1, I_2, Y_Y\right) = \sum\nolimits_{i=\{1,\ldots,N\}} \mathrm{H}\left(Y_Y^i, \hat{Y}_Y^i\right), \quad (8)$$

where $\mathrm{H}(p, q) = -p\log(q) - (1-p)\log(1-q)$ is the binary cross entropy and N is the size of the output 1D vector. $\hat{Y}_Y = f_C\left(f_L\left(I_1\right), f_L\left(I_2\right)\right)$ is the relative yaw angle estimate. Note that we only train the network to estimate

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

the relative yaw angle of a pair of scans with overlap larger than 30%, since it is more uncertain and difficult to estimate the relative yaw angle if the pair of scans are less overlapping which is explained more detailed in Sect. 6.2.3. More uncertain estimations also decrease the accuracy of pose estimation building on top of OverlapNet and the corresponding experimental results can be found in Sect. 6.3.2.

4 OverlapNet for loop closing and SLAM

For loop closing, a threshold on the overlap percentage can be used to decide whether two LiDAR scans are taken from the same place. For finding loop closure candidates, this measure maybe even better than the commonly used distance between the recorded positions of a pair of scans, since the positions might be affected by drift and therefore unreliable. Furthermore, the overlap takes the scene into account, e.g., occlusions between the two scans, and is a direct measure for the number of corresponding points, which can be exploited by ICP that most SLAM systems are employing. The overlap predictions are independent of the relative poses and can be therefore used to find loop closures without knowing the correct relative pose between scans.

4.1 SLAM pipeline

We use the surfel-based mapping system called SuMa proposed by Behley and Stachniss (2018) as our SLAM pipeline and integrate OverlapNet in SuMa replacing its original heuristic loop closure detection method. We only summarize here the key steps of SuMa relevant to our approach and refer for more details to the original paper (Behley and Stachniss 2018).

SuMa uses the same vertex map $\mathcal{V}_D$ and normal map $\mathcal{N}_D$ as described in Sect. 3.1. Furthermore, SuMa uses projective ICP with respect to a rendered map view $\mathcal{V}_M$ and $\mathcal{N}_M$ at timestep $t-1$, the pose update $\mathbf{T}_{C_{t-1}C_t}$ and consequently $\mathbf{T}_{WC_t}$ by chaining all pose increments. Therefore, each vertex $\mathbf{u} \in \mathcal{V}_D$ is projectively associated to a reference vertex $\mathbf{v_u} \in \mathcal{V}_M$. Given this association information, SuMa estimates the transformation between scans by incrementally minimizing the point-to-plane error given by

$$E(\mathcal{V}_D, \mathcal{V}_M, \mathcal{N}_M) = \sum_{\mathbf{u}\in\mathcal{V}_D} \left(\mathbf{n}_u^\top \left(\mathbf{T}_{C_{t-1}C_t}^{(k)}\mathbf{u} - \mathbf{v}_u\right)\right)^2. \tag{9}$$

Each vertex $\mathbf{u} \in \mathcal{V}_D$ is projectively associated to a reference vertex $\mathbf{v}_u \in \mathcal{V}_M$ and its normal $\mathbf{n}_u \in \mathcal{N}_M$ via

$$\mathbf{v}_u = \mathcal{V}_M\left(\Pi\left(\mathbf{T}_{C_{t-1}C_t}^{(k)}\mathbf{u}\right)\right) \tag{10}$$

$$\mathbf{n}_u = \mathcal{N}_M\left(\Pi\left(\mathbf{T}_{C_{t-1}C_t}^{(k)}\mathbf{u}\right)\right). \tag{11}$$

SuMa then minimizes the objective function given in Eq. (9) using Gauss-Newton and determines increments δ by iteratively solving

$$\delta = \left(\mathbf{J}_\delta^\top \mathbf{W}\mathbf{J}_\delta\right)^{-1}\mathbf{J}_\delta^\top \mathbf{W}\mathbf{r}, \tag{12}$$

where $\mathbf{W} \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing weights w_u, $\mathbf{r} \in \mathbb{R}^n$ is the stacked residual vector, and $\mathbf{J}_\delta \in \mathbb{R}^{n\times 6}$ the Jacobian of $\mathbf{r}$ with respect to the increment δ.

SuMa employs a loop closure detection module, which considers the nearest frame in the built map as the candidate for loop closure given the current pose estimate. Loop closure detection works well for small loops, but the heuristic fails in areas with only a few large loops. Furthermore, drifts in the odometry estimate can lead to large displacements, where a heuristic of taking the nearest frame in the already mapped areas into account does not yield correct candidates. This effect is shown in our experiments, see Sect. 6.3.

4.2 Covariance propagation for geometric verification

SuMa's loop closure detection uses a fixed search radius. In contrast, we additionally use the covariance of the pose estimate and error propagation to automatically adjust the search radius.

We assume a noisy pose $\mathbf{T}_{C_{t-1}C_t} = \{\bar{\mathbf{T}}_{C_{t-1}C_t}, \boldsymbol{6}_{C_{t-1}C_t}\}$ with mean $\bar{\mathbf{T}}_{C_{t-1}C_t}$ and covariance $\boldsymbol{6}_{C_{t-1}C_t}$. We can estimate the covariance matrix by

$$\boldsymbol{6}_{C_{t-1}C_t} = \frac{1}{K}\frac{E}{N-M}\left(\mathbf{J}_\delta^\top \mathbf{W}\mathbf{J}_\delta\right)^{-1}, \tag{13}$$

where K is the correction factor of the Huber robustized covariance estimation (Huber 1981), E is the sum of the squared weighted point-to-plane errors (the sum of squared weighted residuals) given the pose $\mathbf{T}_{C_{t-1}C_t}$, see Eq. (9), N is the number of correspondences, $M = 6$ is the dimension of the transformation between two 3D poses.

To estimate the propagated uncertainty during the incrementally pose estimation, we can update the mean and covariance as follows:

$$\bar{\mathbf{T}}_{WC_t} = \bar{\mathbf{T}}_{WC_{t-1}}\bar{\mathbf{T}}_{C_{t-1}C_t} \tag{14}$$

$$\boldsymbol{6}_{WC_t} \approx \boldsymbol{6}_{WC_{t-1}} + \mathbf{J}_{C_{t-1}C_t}\boldsymbol{6}_{C_{t-1}C_t}\mathbf{J}_{C_{t-1}C_t}^\top, \tag{15}$$

where $\mathbf{J}_{C_{t-1}C_t}$ is the Jacobian of Eq. (14).

Since we use the Mahalanobis distance D_M as a probabilistic distance measure between two poses, we make use of Lie algebra to express $\mathbf{T}$ as a 6D vector $\xi \in \mathfrak{se}(3)$ using

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

$\xi = \log \mathbf{T}$, yielding

$$D_M(\mathbf{T}_{C1}, \mathbf{T}_{C2}) = \sqrt{\Delta\xi_{C1C2}^{\top} \boldsymbol{6}_{C1C2}^{-1} \Delta\xi_{C1C2}}. \quad (16)$$

Using the scaled distance, we can now restrict the search space depending on the pose uncertainty to save computation time.

Once we find loop closure candidates, we try to align the current point cloud to the rendered view at the corresponding pose $\mathbf{T}_{WC_{j^*}}$ using the frame-to-model ICP. For the ICP initialization, we use the yaw angle offset estimated from OverlapNet while keeping other setups the same as those used in SuMa. If we have found a loop closure candidate at timestep t, we try to verify it in the subsequent timesteps $t+1, \ldots, t+\Delta_{verification}$, which ensures that we only add consistent loop closures to the pose graph.

5 OverlapNet for global localization

For global localization in a Monte-Carlo localization framework (Dellaert et al. 1999), one of the key challenges lies in the design of the observation model. For the observation model, we need to compare the sensor data and the map. As we want to exploit the overlap and yaw angle predictions of OverlapNet for that, our map consists of virtual scans at discretized 2D locations on a grid. We assume that a point cloud of the environment is available, which we use to extract the map information. Then, we render virtual scans as a preprocessing step. We can then train the network completely self-supervised on the map of virtual scans.

Finally, we integrate an observation model using the overlap and a separate observation model for the yaw angle estimates in a particle filter to perform localization.

5.1 Map of virtual scans

OverlapNet requires two LiDAR scans as input. One is the current scan and the second has to be generated from the map point cloud. Thus, we build a map of virtual LiDAR scans given an aggregated point cloud by using a grid of locations with grid resolution γ, where we generate virtual LiDAR scans for each location. The grid resolution is a trade-off between the accuracy and storage size of the map. Instead of storing these virtual scans, we just need to use one leg of the OverlapNet to obtain a feature volume $\mathbf{F}$ using the input tensor $\mathbf{I}$ of this virtual scan. Storing the feature volume instead of the complete scan has two key advantages: First, it uses more than an order of magnitude less space than the original point cloud (roughly ours: 100 MB/km, raw scans: 1.7 GB/km). Second, we do not need to compute the $\mathbf{F}$ during localization on the map. The features volumes of the virtual scans can directly be used to compute overlap and yaw angle estimates with a query scan that is the currently observed LiDAR point cloud in our localization framework.

5.2 Monte-Carlo localization

Monte-Carlo localization or MCL is a localization algorithm based on the particle filter proposed by Dellaert et al. (1999). Each particle represents a hypothesis for the robot's or autonomous vehicle's 2D pose $\mathbf{x}_t = (x, y, \theta)_t$ at time t. When the robot moves, the pose of each particle is updated with a prediction based on a motion model with the control input $\mathbf{u}_t$. The expected observation from the predicted pose of each particle is then compared to the actual observation $\mathbf{z}_t$ acquired by the robot to update the particle's weight based on the observation model. Particles are resampled according to their weight distribution and resampling is triggered whenever the effective number of particles drops below 50% of the sample size, see Grisetti et al. (2007) for details. After several iterations of this procedure, the particles are likely to converge around the true pose.

MCL realizes a recursive Bayesian filtering scheme. The key idea of this approach is to maintain a probability density $p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ of the pose $\mathbf{x}_t$ at time t given all observations $\mathbf{z}_{1:t}$ up to time t and motion control inputs $\mathbf{u}_{1:t}$ up to time t. This posterior is updated as follows:

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta\, p(\mathbf{z}_t \mid \mathbf{x}_t) \cdot \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})\, d\mathbf{x}_{t-1}, \quad (17)$$

where η is the normalization constant resulting from Bayes' rule, $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$ is the motion model, and $p(\mathbf{z}_t \mid \mathbf{x}_t)$ is the observation model. This paper focuses only on the observation model. For the motion model, we follow a standard odometry model for vehicles (Thrun et al. 2005).

We split the observation model into two parts:

$$p(\mathbf{z}_t \mid \mathbf{x}_t) = p_L(\mathbf{z}_t \mid \mathbf{x}_t)\; p_O(\mathbf{z}_t \mid \mathbf{x}_t), \quad (18)$$

where $\mathbf{z}_t$ is the observation at time t, $p_L(\mathbf{z}_t \mid \mathbf{x}_t)$ is the probability encoding the location (x, y) agreement between the current query LiDAR scan and the virtual scan at the nearest grid position and $p_O(\mathbf{z}_t \mid \mathbf{x}_t)$ is the probability encoding the yaw angle θ agreement between the same pairs of scans.

5.3 OverlapNet-based observation model

Given a particle i with the state estimate (x_i, y_i, θ_i), the overlap estimates encode the location agreement between the query LiDAR scan and virtual scans of the grid cells where

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

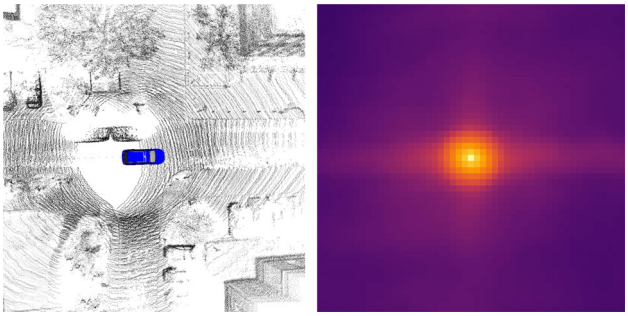


Fig. 5 Overlap observation model describing $p(\mathbf{z}_t \mid \mathbf{x}_t)$ for MCL. Local heatmap of the scan at the car's position with respect to the map. Lighter shades correspond to higher probabilities

particles are located. It can be directly used as the probability:

$$p_L(\mathbf{z}_t \mid \mathbf{x}_t) \propto f(\mathbf{z}_t, \mathbf{z}_i; \mathbf{w}), \tag{19}$$

where f corresponds to the neural network providing the overlap estimation between the input scans $\mathbf{z}_t$, $\mathbf{z}_i$ and $\mathbf{w}$ is the pre-trained weights of the network. $\mathbf{z}_t$ and $\mathbf{z}_i$ are the current query scan and a virtual scan of one (x, y) location respectively.

For illustration purposes, Fig. 5 shows the probabilities of all grid cells in a local area calculated by the overlap observation model. The blue car in the figure shows the current location of the car. The probabilities calculated by the overlap observation model can well represent the hypotheses of the current location of the car.

Typically, a large number of particles are used, especially when the environment is large. However, the computation time increases linearly with the number of particles. When applying the overlap observation model, particles can still obtain relatively large weights as long as they are close to the actual pose, even if not in the exact same position. This allows us to use fewer particles to achieve a high success rate of global localization.

Furthermore, the overlap estimation only encodes the location hypotheses. Therefore, if multiple particles locate in the same grid area, only a single inference using the nearest virtual scan of the map needs to be done, which can further reduce the computation time.

Given a particle i with the state vector (x_i, y_i, θ_i), the yaw angle estimates encode the orientation agreement between the query LiDAR scan and virtual scans of the corresponding grids where particles are located. We formulate the orientation probability as follows:

$$p_O(\mathbf{z}_t \mid \mathbf{x}_t) \propto \exp\left(-\frac{1}{2}\frac{\left(g(\mathbf{z}_t, \mathbf{z}_i; \mathbf{w}) - \theta_i\right)^2}{\sigma_\theta^2}\right), \tag{20}$$

where g corresponds to the neural network providing the yaw angle estimation between the input scans $\mathbf{z}_t$, $\mathbf{z}_i$ and $\mathbf{w}$ is the pre-trained weights of the network. $\mathbf{z}_t$ and $\mathbf{z}_i$ are the current query scan and a virtual scan of one particle respectively.

When generating the virtual scans of the grid map, all virtual scans will be set facing along the absolute 0° yaw angle direction. By doing this, the estimated relative yaw angle between the query scan and the virtual scan indicates the absolute yaw angle of the current query scan. Eq. (20) assumes a Gaussian measurement error in the heading.

By combining overlap and yaw angle estimation, the proposed observation model will correct the weights of particles considering agreements between the query scan and the map with the full pose (x, y, θ).

6 Experimental results

The experimental evaluation is designed to evaluate our approach and support the claims we made in the introduction of this article, which are that our approach is able to: (i) predict the overlap and relative yaw angle between pairs of LiDAR scans by exploiting multiple cues without using relative poses, (ii) combine odometry information with overlap predictions to detect correct loop closure candidates, (iii) improve the overall pose estimation results in a state-of-the-art SLAM system yielding more globally consistent maps, (iv) initialize ICP using the OverlapNet predictions yielding correct scan matching results, (v) build a novel observation model and achieve global localization.

6.1 Implementation, datasets, and experimental setups

We implemented OverlapNet based on Python and Tensorflow (Abadi et al. 2016). The open source implementation is available in form of a stand-alone library, released under the MIT license and can be obtained from GitHub.[1] The source code is well documented with multiple demos to show the functionalities including a loop closing example and can be easily integrated into any other framework. Besides the network, we also released the implementation of the proposed OverlapNet-based global localization method.[2]

For SLAM and loop closing, we train and evaluate our approach on the KITTI odometry benchmark (Geiger et al. 2012). It provides LiDAR scans recorded with a Velodyne HDL-64E showing urban areas around Karlsruhe in Germany. It provides 11 sequences (00–10) with ground truth poses covering different types of environment, e.g., urban, country, and highway. We follow the experimental setup

[1] https://github.com/PRBonn/OverlapNet.

[2] https://github.com/PRBonn/overlap_localization.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

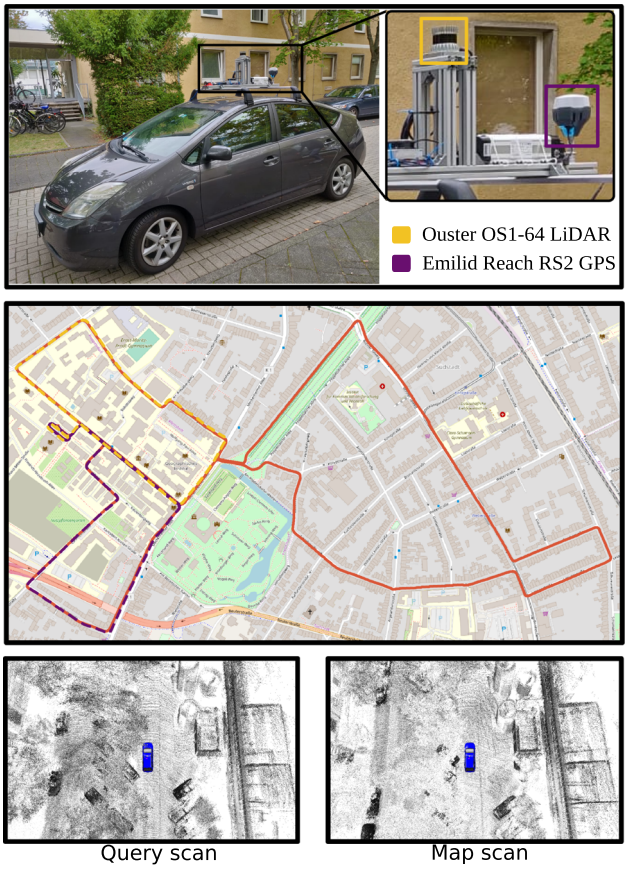


Fig. 6 Upper: sensor setup used for data recording. Middle: trajectories of the dataset used in this paper, overlayed on OpenStreetMap. The orange trajectory represents the sequence used to generate a map for localization. The yellow and purple trajectories represent two different test sequences. Bottom: LiDAR scans of the same place, once during mapping and once during localization. Since the LiDAR data was collected in different seasons, the appearance of the environment changed quite significantly due to changes in the vegetation but also due to parked vehicles at different places (Color figure online)

proposed by Schaupp et al. (2019) and use sequence 00 for evaluation. Sequences 03–10 are used for training and sequence 02 is used for validation.

To evaluate the generalization capabilities of our method, we also test it on the Ford campus dataset (Pandey et al. 2011), which was recorded on the Ford research campus in downtown Dearborn in Michigan using a different version of the Velodyne HDL-64E. In the case of the Ford campus dataset, we test our method on sequence 00, which contains several large loops. Note that we never trained our approach on the Ford campus dataset, only on the KITTI dataset.

For global localization, we use our own IPB-Car dataset, collected with our self-developed sensor platform illustrated in Fig. 6. The KITTI dataset and Ford Campus dataset do not perfectly fulfill our needs for evaluating a localization system, because there are no sequences of the same place but from different seasons available. We therefore collected a large-scale dataset in different seasons with multiple sequences repeatedly exploring the same crowded urban area of Bonn city in Germany using an Ouster OS1-64. For our car dataset, we performed a 3D LiDAR SLAM, SuMa (Behley and Stachniss 2018), combined with a high-precision and SAPOS correction GPS information to create near ground truth poses. During localization, we only use LiDAR scans for global localization *without* using any GPS.

The dataset has three sequences that were collected at different times of the year, sequence 00 in September 2019, sequence 01 in November 2019, and sequence 02 in February 2020. The whole dataset covers a distance of over 10 km. We use LiDAR scans from sequence 02 to build the virtual scans and use sequence 00 and 01 for localization. As can be seen from Fig. 6, the appearance of the environment changes significantly since the dataset was collected in different seasons and in a crowded urban environment, including changes in vegetation, but also parking cars at different locations, moving people, and other objects. The link to the dataset can be found in our open-source code repository.[2]

Besides the open source implementation, we also provide the parameters used in the proposed method in Tables 1 and 2 for the purpose of reproducibility to the experimental results. Table 1 shows the configuration of the network architecture and Table 2 shows hyper parameters used in the proposed method. As shown in Table 2, for generating the range image following SuMa setup (Behley and Stachniss 2018), we only use points within a distance of 75 m to the sensor and generate range images with height $h = 64$ and width $w = 900$. For overlap computation, see Eq. (3), we use $\epsilon = 1$ m. We use a learning rate of 10^{-3} with a decay of 0.99 every epoch and train at most 100 epochs. For the combined loss, Eq. (6), we set $\alpha = 5$. For the overlap loss, Eq. (7), we use $a = 0.25$, $b = 12$, and scale factor $s = 24$. For the calculation of yaw angle

Table 2 Hyper parameters of our approach

Parameter	Description	Value
h	Height of range image	64
w	Width of range image	900
$d_{\max}$	Maximum range	75 m
ϵ	Threshold to count overlap pixels	1 m
a	Constant offset of sigmoid	0.25
b	Constant offset of sigmoid	12
s	Scaling factor of sigmoid	24
σ_θ	Sigma for orientation probability	10 deg
d_{converge}	Threshold to success converge	5 m
Network training	Learning rate	10^{-3}
	Decay of every epoch	0.99
	Number of epochs	100
	Overlap loss weight α	5

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

probability, Eq. (20), we set $\sigma_\theta = 10$. We use the a threshold of $d_{\text{converge}} = 5\,m$ to decide whether the global localization converges or not.

6.2 Overlap and yaw angle evaluation

In this section, we show the experiments that support the claim that our approach is able to estimate the overlap and yaw angle offset between pairs of LiDAR scans, which is well-suited for solving the general similarity estimation. We also provide an ablation study of using different input modalities and the analysis of the relationship between the overlap and yaw angle estimations.

In the case of general LiDAR scans similarity estimation, we assume that we have no prior information about the robot pose. We compare our method with the state-of-the-art learning-based methods LocNet (Yin et al. 2019), LocNet++ and OREOS (Schaupp et al. 2019), and also the traditional hand-crafted feature-based method FPFH+RANSAC. We follow the experimental setup of OREOS, where the KITTI sequence 00 is used for the evaluation. The LiDAR scans from the first 170 s of sequence 00 are used to generate the database, as the vehicle starts to revisit previously traversed areas after 170 s. The remaining LiDAR scans are used for place recognition queries. The query point clouds are sampled to be at least 3 m apart. Two point clouds are considered in the same place, if their ground-truth poses are within 1.5 m. The baseline results are those produced by the authors of OREOS (Schaupp et al. 2019).

6.2.1 Overlap estimation for Similarity measurement

In a general place recognition application, multiple candidates may be retrieved according to the similarity measurements with respect to the current query LiDAR scan. The respective place recognition candidates recall results are shown in Fig. 7. Our method outperforms all baseline methods when using a small number of candidates and attains similar performance as baseline methods for higher values of numbers of candidates. However, OREOS and LocNet++ attain a slightly higher recall if more candidates are considered.

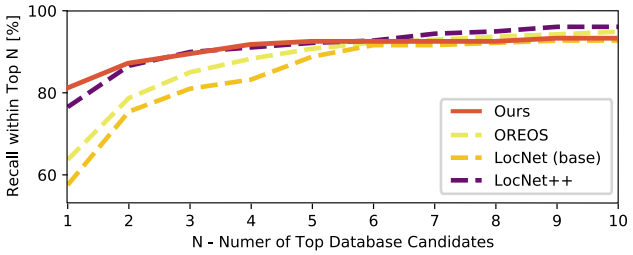


Fig. 7 Place recognition performance on KITTI sequence 00

Table 3 Yaw estimation errors

Approach	Mean (deg)	Std (deg)	Recall (%)
FPFH+RANSAC*	13.28	32.19	97
OREOS*	12.67	15.23	100
Ours	**1.13**	**3.34**	**100**

Bold numbers indicate best performance *The results are those produced by the authors of OREOS

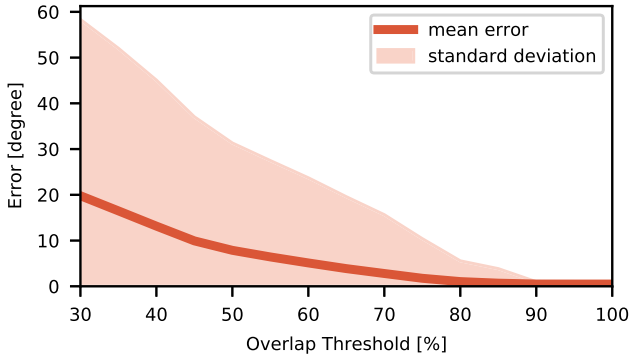


Fig. 8 Overlap and yaw estimation relationship

6.2.2 Yaw angle estimation

Table 3 summarizes the yaw angle errors on KITTI sequence 00. We can see that our method outperforms the other methods in terms of mean error and standard deviations. In terms of recall, OverlapNet and OREOS always provide a yaw angle estimate, since both approaches are designed to estimate the relative yaw angle for any pairs of scans in contrast to the RANSAC-based method that sometimes fails.

The superior performance can be mainly attributed to the correlation head exploiting the fact that the orientation in LiDAR scans can be well represented by the shift in the range projection. Therefore, it is easier to train the correlation head to accurately predict the relative yaw angles rather than a multilayer perceptron used in OREOS. Furthermore, there is also a strong relationship between overlap and yaw angle, which also improves the results when trained together.

6.2.3 Relationship between overlap and yaw estimations

Figure 8 shows the relationship between real overlap and yaw angle estimation error. As expected, the yaw angle estimate gets better with increasing overlap. Based on these plots, our method not only finds candidates but also measures the quality, i.e., when the overlap of two scans is larger than 90%, our method can accurately estimate the relative yaw angle with an average error of about only 1°.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 4 Ablation study on usage of input modalities

$\mathcal{R}$	$\mathcal{N}$	$\mathcal{I}$	$\mathcal{S}$	Overlap AUC	Overlap F1	Yaw (degree) Mean	Yaw (degree) Std
✓				0.86	0.87	11.67	25.32
✓	✓			0.86	0.85	2.97	14.28
✓	✓	✓		0.87	0.87	2.53	14.56
✓	✓	✓	✓	0.87	0.88	1.13	3.34

$\mathcal{R}$, range depth; $\mathcal{N}$, normal; $\mathcal{I}$, intensity; $\mathcal{S}$, semantic information. AUC, area under the curve, F1, balanced F-score

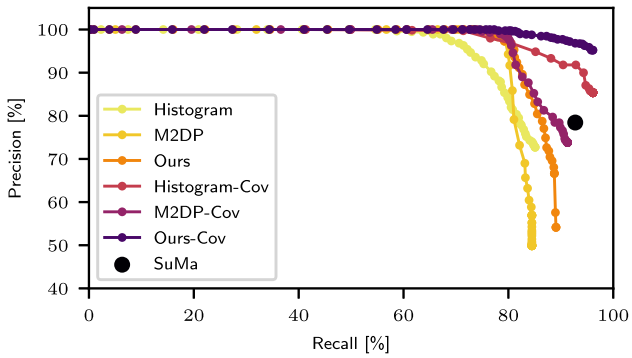
6.2.4 Ablation study on input modalities

An ablation study on the usage of different inputs is shown in Table 4. As can be seen, when employing more input modalities, the proposed method gets more robust. We notice that exploiting only depth information with OverlapNet can already perform reasonably in terms of overlap prediction, while it does not perform well in yaw angle estimation. When combining this with normal information, the OverlapNet can perform well on both tasks. Another interesting finding is the drastic reduction of yaw angle mean error and standard deviation when using semantic information. One reason could be that adding semantic information will make the input data more distinguishable when the car drives in symmetrical environments.

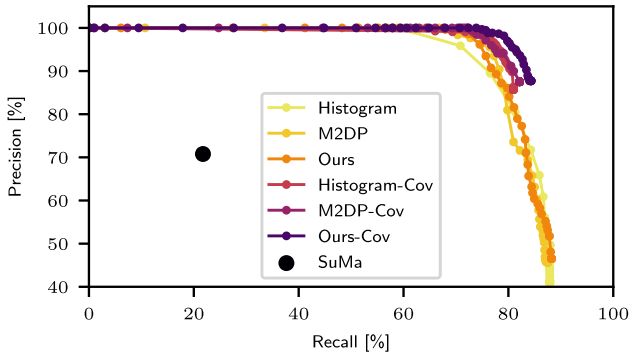
6.3 OverlapNet-based loop closing

In the following experiments, we investigate the loop closing performance of our approach and compare it to existing methods. Different from the general place recognition, loop closure detection typically assumes that robots revisit places during the mapping while moving with uncertain odometry. Therefore, the prior information about robot poses extracted from the pose graph is available for the loop closure detection. The following criteria are used in these experiments:

- To avoid detecting a loop closure in the most recent scans, we do not search candidates in the latest 100 scans.
- For each query scan, only the best candidate is considered throughout this evaluation.
- Most SLAM systems search for potential loop closures only within the 3σ area around the current pose estimate. We do the same, either using the Euclidean or the Mahalanobis distance, depending on the approach.
- We aim to find more loops even in some challenging situations with low overlaps, e.g., when the car drives back to an intersection from the opposite direction as shown in Fig. 1. We use the overlap value to decide if a candidate is a true positive rather than distance. Furthermore, ICP can find correct poses if the overlap between pairs of scans is around 30%, as illustrated in Sect. 6.3.2.



(a) KITTI sequence 00



(b) Ford campus sequence 00

Fig. 9 Precision–Recall curves of different approaches

We evaluate OverlapNet on both the KITTI dataset and the Ford campus dataset to showcase the generalization capabilities of the approach.

6.3.1 Quantitative analysis

Figure 9 shows the precision–recall curves of different loop closure detection methods. We compare our method, trained with two heads and all cues labeled as *Ours (AllChannel, TwoHeads)*, with three state-of-the-art approaches: a histogram-based approach (*Histogram*) by Röhling et al. (2015), *M2DP* by He et al. (2016), and the original heuristic approach of SuMa by Behley and Stachniss (2018). Since SuMa always uses the nearest frame as the candidate for loop closure detection, we can only get one pair of precision and recall value resulting in a single point. We also show the result of our method using prior information, *Ours-Cov*, which uses covariance propagation (Sect. 4.2) to define the search space with the Mahalanobis distance and use the nearest in Mahalanobis distance of the top 10 predictions of OverlapNet as the loop closure candidates. For a fair comparison, we also show the results that can be obtained by enhancing the base-

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

lines with the proposed covariance-based method. These are labeled as *Histogram-Cov* and *M2DP-Cov* respectively.

Table 5 shows the comparison between our approach and the state of the art using the F1 score and the area under the curve (AUC) on both, KITTI and Ford campus dataset. For the KITTI dataset, our approach uses the model trained with all cues, including depth, normals, intensity, and a probability distribution over semantic classes. For the Ford campus dataset, our approach uses the model trained on KITTI with geometric information only, namely *Ours (GO)*, since other cues are not available in this dataset. We can see that our method outperforms the other methods on the KITTI dataset and attains a similar performance on the Ford campus dataset. There are two reasons to explain the worse performance on the Ford campus dataset. First, we never trained our network on the Ford campus dataset or even US roads, and secondly, there is only geometric information available on the Ford campus dataset. However, our method outperforms all baseline methods in both, KITTI and Ford campus dataset, if we integrate prior information.

We also show the performance in comparison to variants of our method in Table 6. We compare our best model *AC-TH* using all available cues and two heads to a variant which only uses a basic multilayer perceptron as the head named *MLPOnly* which consists of two hidden fully connected layers and a final fully connected layer with two neurons (one for overlap, one for yaw angle). The substantial difference of the AUC and F1 scores shows that such a simple network structure is not sufficient to get a good result. Training the network with only one head (only the delta head for overlap estimation, named *DeltaOnly*), has not a significant influence on the performance. A huge gain can be observed when regarding the nearest frame in Mahalanobis distance of the top 10 candidates in overlap percentage (*Ours-Cov*).

Table 5 Comparison with state of the art

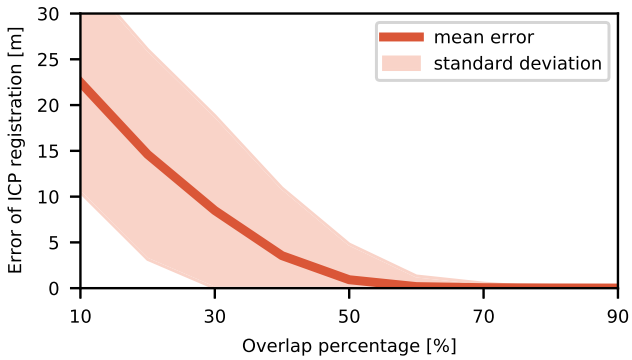
Dataset	Approach	AUC	F1 score
KITTI	SuMa	–	0.85
	Histogram	0.83	0.83
	Histogram-Cov	0.95	0.92
	M2DP	0.83	0.87
	M2DP-Cov	0.89	0.88
	Ours (AC-TH)	0.87	0.88
	Ours-Cov (AC-TH)	**0.96**	**0.96**
Ford Campus	SuMa	–	0.33
	Histogram	0.84	0.83
	Histogram-Cov	0.85	0.85
	M2DP	0.84	0.85
	M2DP-Cov	0.85	0.86
	Ours (GO)	0.85	0.84
	Ours-Cov (GO)	**0.85**	**0.88**

Table 6 Comparison with our variants

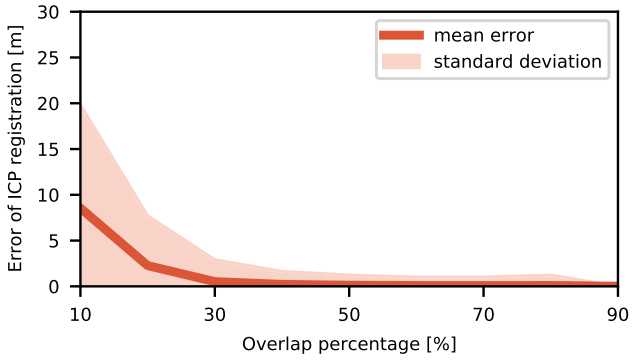
Dataset	Variant	AUC	F1 score
KITTI	MLPOnly	0.58	0.65
	DeltaOnly	0.85	0.88
	Ours (AC-TH)	0.87	0.88
	Ours-Cov (AC-TH)	**0.96**	**0.96**
Ford Campus	Ours (GO)	0.85	0.84
	Ours-Cov (GO)	**0.85**	**0.88**

6.3.2 Using OverlapNet predictions as an initial guess for ICP registration

We aim at supporting the claim that our network provides good initializations for 3D LiDAR-based ICP registration in the context of autonomous driving. Figure 10 shows the relations between the overlap and ICP registration error with and without using OverlapNet predictions as initial guesses. The error of the ICP registration is here the Euclidean distance between the estimated relative translation and the ground



(a) With identity as initial guess



(b) With yaw angle as initial guess

Fig. 10 ICP using OverlapNet predictions as the initial guess. The error of ICP registration here is the Euclidean distance between the estimated translation and the ground-truth translation

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

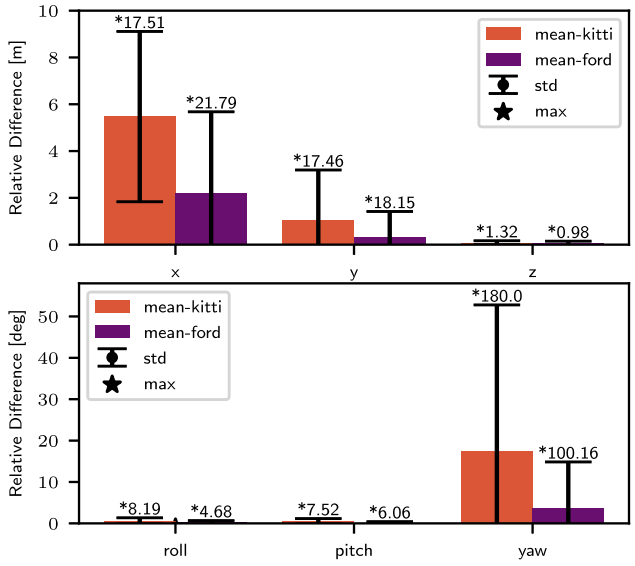


Fig. 11 Statistics on relative 6 degree of freedom pose $(x, y, z, roll, pitch, yaw)$ between pairs of scans with overlap larger than 30%. Results on KITTI and Ford campus datasets are shown

truth translation. As can be seen, the yaw angle prediction of the OverlapNet increases the chance to get a good result from the ICP even when two frames are relatively far away from each other, i.e., have only a low overlap. Therefore in some challenging cases, e.g., the car drives back into an intersection from a different street, our approach can still find loop closures. The results also show that the overlap estimates measure the quality of the found loop closure: larger overlap values result in better registration results of the involved ICP.

To analyze the correlation between the overlap and the 6 degree of freedom pose $(x, y, z, roll, pitch, yaw)$, we show the statistics with all sequences of both KITTI and Ford Campus datasets in Fig. 11. As can be seen, with overlap values larger than 30%, the relative differences in roll, pitch are much smaller than that in yaw. This means that our method is likely to filter out such cases where there are large differences in roll and pitch corresponding to low overlaps between pairs of scans and therefore are not good to be used for loop closing. Furthermore, these experiments also show the motivation that it is more important to estimate the yaw angle rather than roll and pitch, since even when two scans have a large overlap, the yaw angle offset could be very large. For example, when the car drives in both directions on a road or a slope, the yaw angle offset is around 180 degrees. Note that, our method is not influenced by such cases, because it can estimate the yaw angle offset between pairs of scans. After rotating in yaw, the offset of other orientation angles are small and can typically be handled by the following ICP that is used in general loop closing.

6.3.3 Improving SLAM results

This experiment supports our claim that our method is able to improve the overall SLAM result. Figure 12 shows the odometry results on KITTI sequence 02. The color in Fig. 12 shows the 3D translation error (including height). The top figure shows the SuMa method and the bottom figure shows *Ours-Cov* using the proposed OverlapNet to detect loop closures. We can see that after integrating our method, the overall odometry is much more accurate since we can provide more loop closure candidates with higher accuracy in terms of overlap. The colors represent the translation error of the estimated poses with respect to the ground truth. Furthermore, after integrating the proposed OverlapNet, the SLAM system can find more loops even in some challenging situations, e.g., when the car drives back to an intersection from the opposite direction as shown in Fig. 1.

6.4 OverlapNet-based global localization

In the following experiments, we test our proposed localization method by integrating the OverlapNet based sensor model into the MCL framework. The MCL frameworks are the same for all baselines and we only exchange the observation models. Note that in general maps of the environment are built using previously recorded data. Thus the task is more difficult than loop closing because of larger environmental changes between the map and the new scans. We will show in the following experiments that our method is nevertheless able to perform global localization.

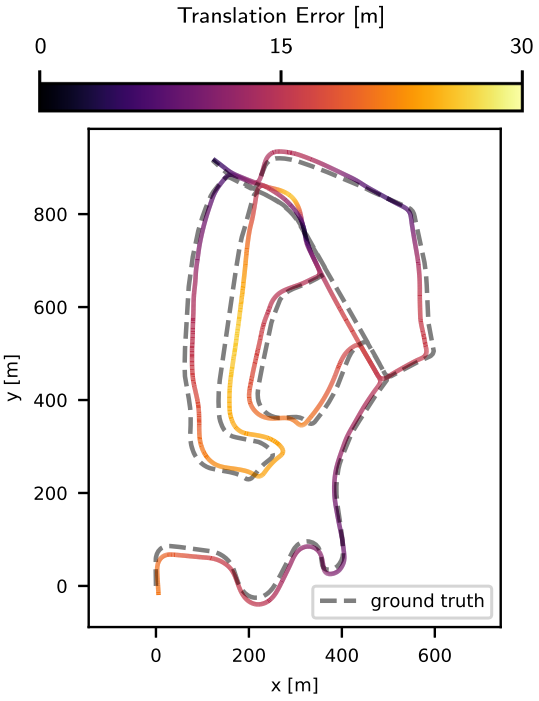
6.4.1 Baselines

We compare our observation model with two baseline observation models: the typical beam-end model as described by Thrun et al. (2005) and a histogram-based model derived from the work of Röhling et al. (2015).

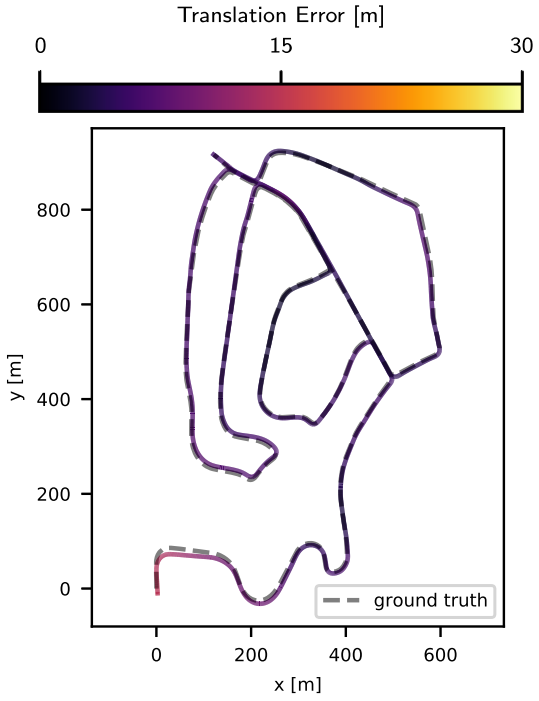
The beam-end observation model is often used for 2D LiDAR data. For 3D LiDAR scans, it needs much more particles to make sure that it converges to the correct pose, which causes the computation time to increase substantially. In this paper, we implement the beam-end model with a downsampled point cloud map using voxelization with a resolution of 10 cm.

Our second baseline for comparison uses the model by Röhling et al. (2015), which proposed a fast method to detect loop closures through the use of similarity measures on histograms extracted from 3D LiDAR data. The histogram contains the range information, and use it in the MCL framework as a baseline observation model. We employ the same grid map and virtual frames as used for our method with the histogram-based observation model. When updating the weights of particles, we first generate the histograms of the

Content courtesy of Springer Nature, terms of use apply. Rights reserved.



(a) SuMa



(b) Ours-Cov

Fig. 12 Qualitative result on KITTI sequence 02 comparing SuMa to our approach

current query scan and the virtual scans of grids where the particles locate. Then, we use the same Wasserstein distance (Dobrushin 1970) to measure the similarity between them and update the weights of particles as follows:

$$p(\mathbf{z}_t|\mathbf{x}_t) \propto d\left(h(\mathbf{z}_t), h(\mathbf{z}_i)\right), \tag{21}$$

where d represents the Wasserstein distance between histograms $h(\mathbf{z}_t)$, $h(\mathbf{z}_i)$ of LiDAR scan $\mathbf{z}_t$, $\mathbf{z}_i$.

6.4.2 Localization performance

The experiment presented in this section is designed to show the performance of our approach and to support the claim that it is well suited for global localization.

First of all, we show the general localization results tested with two sequences of the IPB-Car dataset in Fig. 13. For the overlap network, we used only geometric information (range and normal images) as input. The qualitative results show that, after applying our sensor-model, the proposed method can well localize in the map with only LiDAR data collected in highly dynamic environments at different times.

For quantitative results, we calculate the success rate for different methods with different numbers of particles, as shown in Fig. 14. The x-axis represents the number of particles used during localization, while the y-axis is the success rate of different setups. The success rate for a specific setup of one method is calculated using the number of success cases divided by the total number of the tests. To decide whether one test is successful or not, we check the location error by every 100 frames after converging. If the location error is smaller than a certain threshold, we count this run as a success case.

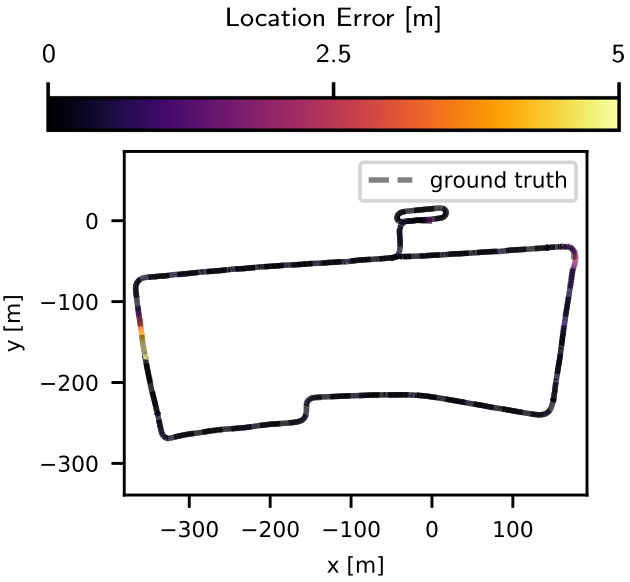
We test our method together with two baselines using five different numbers of particles $N = \{1000; 5000; 10{,}000; 50{,}000; 100{,}000\}$. For each setup, we sample 10 trajectories and perform global localization.

Quantitative results of localization accuracy are shown in Table 7. The upper part shows the location error of all methods tested with both sequences. The location error is defined as the root mean square error (RMSE) of each test in terms of the Euclidean error computed in $(x,\ y)$ with respect to the ground truth poses. It shows the mean and the standard deviation of the error for each observation model. Note that the location error is only calculated for success cases.

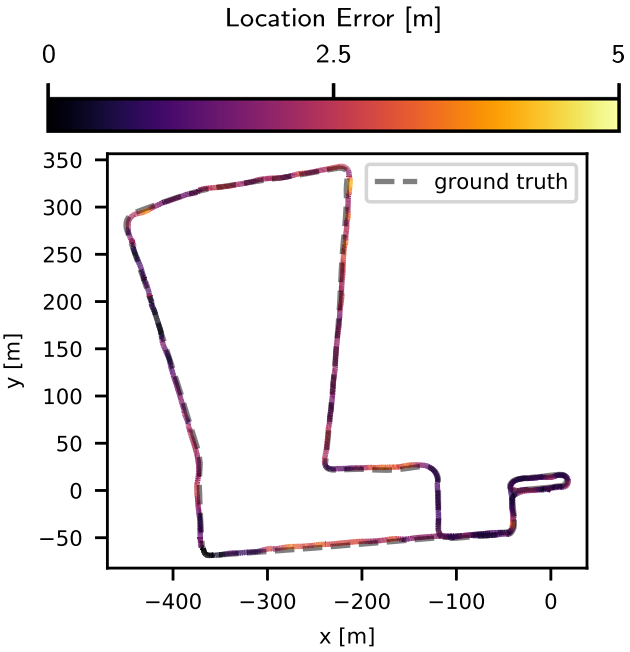
The lower part shows the yaw angle error. It is the RMSE of each test in terms of yaw angle error with respect to the ground truth poses. The table shows the mean and the standard deviation of the error for each observation model. As before, the yaw angle error is also only calculated for cases in which the global localization converged.

As can be seen from the results, our method achieves higher success rates with a smaller number of particles compared to the baseline methods, which also makes the proposed method faster than baseline methods. Furthermore, our method converges already with 100,000 particles in all cases, whereas the other observation models still need more particles to sufficiently cover the state space. Moreover, the

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

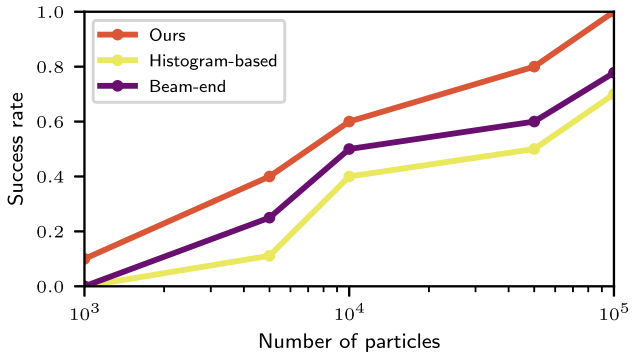


(a) Localization trajectory from sequence 00

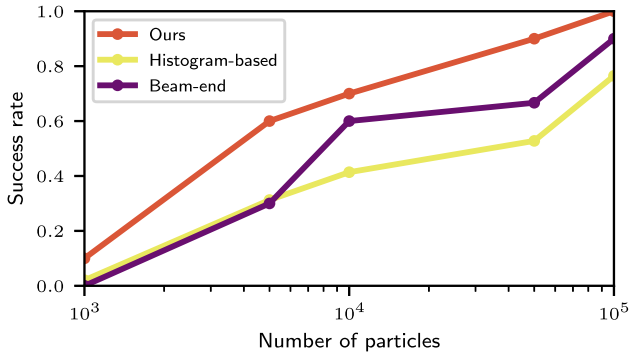


(b) Localization trajectory from sequence 01

Fig. 13 Localization results of our method with 10,000 particles on two sequences recorded with the setup depicted in Fig. 6. Shown are the ground truth trajectory (black) and our estimated trajectory using our observation model (red) (Color figure online)



(a) sequence 00



(b) sequence 01

Fig. 14 Success rate of the different observation models for 10 global-ization runs. Here, we use sequence 00 and sequence 01 to localize in the map of the IPB-Car dataset. We count runs as success if converge to the ground truth location within 5 m

Table 7 Localization results

Sequence	Location error (meter)		
	Beam-end	Histogram	Ours
0	0.92 ± 0.27	1.85 ± 0.34	$\mathbf{0.81 \pm 0.13}$
1	$\mathbf{0.67 \pm 0.11}$	1.86 ± 0.34	0.88 ± 0.07
Sequence	Yaw angle error (degree)		
	Beam-end	Histogram	Ours
0	1.87 ± 0.47	3.10 ± 3.07	$\mathbf{1.74 \pm 0.11}$
1	2.10 ± 0.59	3.11 ± 3.08	$\mathbf{1.88 \pm 0.09}$

proposed method gets similar performance in location error compared to the baseline methods but it achieves better results in yaw angle estimation. This is because the proposed method decouples the location and yaw angle estimation and, therefore, can exploit more constraints in yaw angle corrections.

To sum up, for global localization the proposed method outperforms the baseline methods in terms of success rate, while getting similar results in terms of location error. Moreover, our method outperforms baseline methods in yaw angle estimation, because of the proposed de-coupled observation model. Furthermore, our method is faster than the baseline methods.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

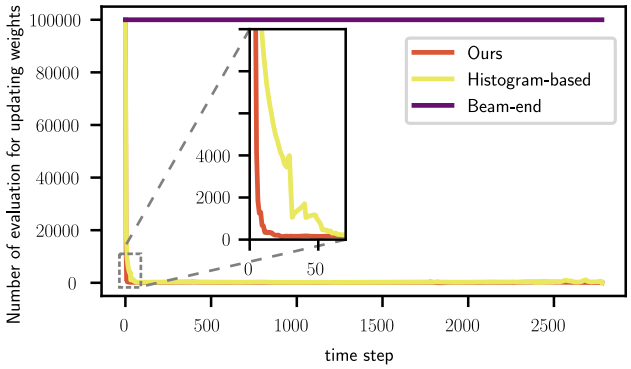


Fig. 15 Number of observation model evaluations for updating the weights at each timestep with 100,000 particles. The beam end model needs to be evaluated for each and every particle individually. The histogram-based method is more computationally efficient, while our proposed method still needs the fewest evaluations

6.5 Runtime

We tested our method on a system equipped with an Intel i7-8700 with 3.2 GHz and an Nvidia GeForce GTX 1080 Ti with 11 GB memory. When we use only geometric information, our method takes on average per frame 2 ms for feature extraction and 5 ms for head comparison. It takes additional 75 ms to perform semantic segmentation using RangeNet++.

In a real SLAM implementation, most systems only search loop closure candidates inside a certain search space given by pose uncertainty using the Mahalanobis distance (see Sect. 4.2). Once we generated a feature volume for a scan, it will be stored in memory. During the search process, we need only to generate the feature volume for the current scan and compare it to the feature volumes in memory. Therefore, our method can achieve online operation in long-term tasks, since we usually have to compare only a small number of candidate poses.

For global localization, we show the *number* of observation model evaluations necessary for updating the weights at each time step in Fig. 15. This is a fairer way to compare the computational cost of different methods, since our neural network-based method uses a GPU to concurrently updating the weights of particles, while the other methods only use a CPU. As can be seen, our method needs a smaller number of observation model evaluations to update the weights for all particles. This is because we only need to perform the network inference for all particles which are localized in the same grid cell once. For an incoming frame and the virtual frame of that grid cell, the inputs of the network and thus the outputs are the same for all particles in that grid cell.

For *initializing in the large-scale map*, the worst case will take around 43 s to process one frame. After convergence, the proposed method takes only 1 s on average to process one frame with 10,000 particles.

7 Conclusion

This paper addressed loop closing and localization for a vehicle using 3D LiDAR data. We proposed a modified Siamese network to estimate the similarity between pairs of scans using an image overlap generalized to range images and that provides a relative yaw angle estimate. For loop closure detection, we integrated our approach into an existing SLAM system to improve its mapping and odometry results. For global localization, we proposed a novel observation model using the concept of overlap and integrated this observation model into a Monte-Carlo localization framework to achieve good localization results. The experiments suggest that our method is able to estimate the overlap and yaw angle estimates between pairs of LiDAR scans, which can be successfully used for both, loop closing and global localization. Moreover, our method generalizes well to different environments at different times.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265–283).

Barsan, I., Andrei, I., Wang, S., Pokrovsky, A., & Urtasun, R. (2018). Learning to localize using a LiDAR intensity map. In *Proceedings of the second conference on robot learning (CoRL)* (pp. 605–616).

Behley, J., & Stachniss, C. (2018). Efficient surfel-based SLAM using 3D laser range data in urban environments. In *Proceedings of robotics: science and systems (RSS)*.

Besl, P., & McKay, N. (1992). A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 14*(2), 239–256.

Bogoslavskyi, I., & Stachniss, C. (2016). Fast range image-based segmentation of sparse 3d laser scans for online operation. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a "Siamese" time delayed neural network.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 07(04), 669–688. https://doi.org/10.1142/S0218001493000339.

Chen, X., Läbe, T., Milioto, A., Röhling, T., Vysotska, O., Haag, A., Behley, J., & Stachniss, C. (2020a) OverlapNet: Loop closing for LiDAR-based SLAM. In *Proceedings of robotics: science and systems (RSS)*.

Chen, X., Läbe, T., Nardi, L., Behley, J., & Stachniss, C. (2020b). Learning an overlap-based observation model for 3D LiDAR localization. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Chen, X., Milioto, A., Palazzolo, E., GiguÈre, P., Behley, J., & Stachniss, C. (2019). SuMa++: Efficient LiDAR-based semantic SLAM. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Chen, X., Vizzo, I., Läbe, T., Behley, J., & Stachniss, C. (2021). Range image-based LiDAR localization for autonomous vehicles. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.

Cop, K., Borges, P., & Dub, R. (2018). Delight: An efficient descriptor for global localisation using lidar intensities. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.

Cramariuc, A., Dube, R., Sommer, H., Siegwart, R., & Gilitschenski, I., (2018). Learning 3D segment descriptors for place recognition. *arXiv preprint*.

Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo localization for mobile robots. In *IEEE international conference on robotics and automation (ICRA)*.

Dobrushin, R. (1970). Prescribing a system of random variables by conditional distributions. *Theory of Probability & its Applications, 15*(3), 458–486.

Dubé, R., Dugas, D., Stumm, E., Nieto, J., Siegwart, R., & Lerma, C. (2017). SegMatch: Segment based place recognition in 3D point clouds. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.

Fischler, M., & Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM, 24*(6), 381–395.

Fox, D., Burgard, W., Dellaert, F., & Thrun, S. (1999). Monte Carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the conference on advancements of artificial intelligence (AAAI)* (pp. 343–349).

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3354–3361).

Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with Rao–Blackwellized particle filters. *IEEE Transactions on Robotics, 23*(1), 34–46.

Guo, J., Borges, P., Park, C., & Gawel, A. (2019). Local descriptor for robust place recognition using LiDAR intensity. *IEEE Robotics and Automation Letters, 4*(2), 1470–1477.

He, L., Wang, X., & Zhang, H. (2016). M2DP: A novel 3D point cloud descriptor and its application in loop closure detection. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Huber, P. J. (1981). *Robust statistics*. Wiley.

Hussain, M., & Bethel, J. (2004). Project and mission planing. In: C. McGlone, E. Mikhail, J. Bethel, & R. Mullen (Eds.), *Manual of Photogrammetry, chapter 15.1.2.6* (pp. 1109–1111) American Society for Photogrammetry and Remote Sensing.

Kim, G., Park, B., & Kim, A. (2019). 1-day learning, 1-year localization: Long-term LiDAR localization using scan context image. *IEEE Robotics and Automation Letters, 4*(2), 1948–1955.

Kong, X., Yang, X., Zhai, G., Zhao, X., Zeng, X., Wang, M., Liu, Y., Li, W., & Wen, F. (2020). Semantic graph based place recognition for point clouds. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Levinson, J., Montemerlo, M., & Thrun, S. (2007). Map-based precision vehicle localization in urban environments. In *Proceedings of robotics: science and systems (RSS)*.

Ma, W., Tartavull, I., Bârsan, I.A., Wang, S., Bai, M., Mattyus, G., Homayounfar, N., Lakshmikanth, S. K., Pokrovsky, A., & Urtasun, R. (2019). Exploiting sparse semantic HD maps for self-driving vehicle localization. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Merfels, C., & Stachniss, C. (2016). Pose fusion with chain pose graphs for automated driving. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Milioto, A., & Stachniss, C. (2019). Bonnet: An open-source training and deployment framework for semantic segmentation in robotics using CNNs. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.

Milioto, A., Vizzo, I., Behley, J., & Stachniss, C. (2019). RangeNet++: Fast and accurate LiDAR semantic segmentation. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Nagashima, S., Ito, K., Aoki, T., Ishii, H., & Kobayashi, K. (2007). A high-accuracy rotation estimation algorithm based on 1D phase-only correlation. In *Proceedings of the international conference on image analysis and recognition* (pp. 210–221).

Pandey, G., McBride, J., & Eustice, R. (2011). Ford campus vision and LiDAR data set. *International Journal of Robotics Research, 30*(13), 1543–1552.

Röhling, T., Mack, J., & Schulz, D. (2015). A fast histogram-based similarity measure for detecting loop closures in 3-D LiDAR data. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 736–741).

Schaefer, A., Büscher, D., Vertens, J., Luft, L., & Burgard, W. (2019) Long-term urban vehicle localization using pole landmarks extracted from 3-D LiDAR scans. In *Proceedings of the European conference on mobile robotics (ECMR)* (pp. 1–7).

Schaupp, L., Bürki M, Dubé, R., Siegwart, R., & Cadena, C. (2019). OREOS: Oriented recognition of 3D point clouds in outdoor scenarios. *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Silver, D., & Stentz, A. (2011). Monte Carlo localization and registration to prior data for outdoor navigation. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Stachniss, C., Leonard, J., & Thrun, S. (2016). *Springer Handbook of Robotics*, chapter 46: Simultaneous Localization and Mapping (2nd ed.) Springer.

Steder, B., Ruhnke, M., Grzonka, S., & Burgard, W. (2011). Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Steder, B., Rusu, R., Konolige, K., & Burgard, W. (2010). NARF: 3D range image features for object recognition. In *Workshop on defining and solving realistic perception problems in personal robotics at the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Sun, L., Adolfsson, D., Magnusson, M., Andreasson, H., Posner, I., & Duckett, T. (2020). Localising faster: Efficient and precise lidar-based robot localisation in large-scale environments. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT Press.

Tinchev, G., Penate-Sanchez, A., Fallon, M. (2019) Learning to see the wood for the trees: Deep laser localization in urban and natural

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

environments on a CPU. *IEEE Robotics and Automation Letters (RA-L)* 4(2): 1327–1334.
Uy, A., & Lee, G. (2018). PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 4470–4479).
Vizzo, I., Chen, X., Chebrolu, N., Behley, J., & Stachniss, C. (2021). Poisson surface reconstruction for LiDAR odometry and mapping. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.
Vysotska, O., & Stachniss, C. (2017). Improving slam by exploiting building information from publicly available maps and localization priors. *Photogrammetrie - Fernerkundung - Geoinformation (PFG), 85*(1), 53–65.
Wei, X., Bârsan, I.A., Wang, S., Martinez, J., & Urtasun, R. (2019). Learning to localize through compressed binary maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 10316–10324).
Wiesmann, L., Milioto, A., Chen, X., Stachniss, C., & Behley, J. (2021). Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters, 6,* 2060–2067. https://doi.org/10.1109/LRA.2021.3059633.
Wilbers, D., Merfels, C., & Stachniss, C. (2019). Localization with sliding window factor graphs on third-party maps for automated driving. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.
Wolcott, R., & Eustice, R. (2015). Fast lidar localization using multiresolution gaussian mixture maps. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)* (pp. 2814–2821).
Yan, F., Vysotska, O., & Stachniss., C. (2019). Global localization on OpenStreetMap using 4-bit semantic descriptors. In *Proceedings of the European conference on mobile robotics (ECMR)*.
Yin, H., Wang, Y., Ding, X., Tang, L., Huang, S., & Xiong, R. (2019). 3D LiDAR-based global localization using Siamese neural network. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2019.2905046.
Zaganidis, A., Zerntev, A., Duckett, T., & Cielniak, G. (2019). Semantically assisted loop closure in SLAM using NDT histograms. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
Zhang, C., Ang, M.H., & Rus, D. (2018). Robust LiDAR localization for autonomous driving in rain. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 3409–3415).

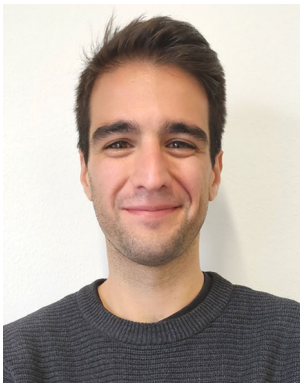
Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Xieyuanli Chen is a Ph.D. student at Cyrill Stachniss' Photogrammetry and Robotics Laboratory, the University of Bonn. He is now also a member of the Technical Committee of RoboCup Rescue Robot League (RRL). He received his Master degree in Robotics in 2017 at the National University of Defense Technology, China. During that time, he was a member of the Organizing Committee of RoboCup Rescue Robot League. He received his Bachelor degree in Electrical Engineering and Automation in 2015 at Hunan University, China.



Thomas Läbe is a technical associate at the professorship of Photogrammetry of the Institute of Geodesy and Geoinformation. He finished his diploma in computer sciene ("Dipl.-Inform. (FH)") in 1994 at the Cologne University of Applied Sciences. From 1994 to 2000 he was employed at the Institute of Photogrammetry in the joint project "Passunktmodelldatenbank für die automatische Orientierung von Luftbildern" (automatic orientation of images) which was a cooperation with the land survey department ("Landesvermessungsamt") of North-Rhine-Westfalia. Since 2001 he is a software developer and administrator at the professorship of Photogrammetry.



Andres Milioto is a Senior Computer Vision Engineer at Scythe Robotics, in Boulder, Colorado, USA. Before this, he was a doctoral student and research assistant at Cyrill Stachniss' Photogrammetry and Robotics Laboratory, in Bonn, Germany. His main topics of research are semantic scene understanding for camera images and LiDAR point clouds, and efficiency techniques to make deep learning run online on embedded devices. Before his Ph.D., he got a Masters Degree in Electronics Engineering from Universidad Nacional de Rosario, in Rosario, Argentina, where he graduated top of his class in 2016.



Timo Röhling got his Diploma degree in computer science from the University of Bonn and is currently working towards his Ph.D. He is employed at the Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE). His research focuses on SLAM related problems and LiDAR sensor data processing with GPU acceleration.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.



Jens Behley is a postdoctoral researcher at the Department for Photogrammetry since February 2016. From September 2008 to July 2015, Jens worked at the Department for Computer Science III, University of Bonn and he successfully defended his Ph.D. thesis on "Three-dimensional Laser-based Classification in Outdoor Environments" supervised by Prof. Dr. Armin B. Cremers in January 2014.



Cyrill Stachniss is a full professor at the University of Bonn and heads the Photogrammetry and Robotics Lab since 2014. Before his appointment in Bonn, he was a lecturer at the University of Freiburg and the Swiss Federal Institute of Technology. The research activities of the Photogrammetry and Robotics Lab focus on probabilistic techniques in the context of mobile robotics, navigation problems, and perception. The lab has made several contributions in the areas of SLAM, localization, place recognition, semantic scene interpretation, autonomous exploration, and planning. It covers ground vehicles including mobile robots and autonomous cars as well as UAVs. The main application areas of the research are autonomous service robots, agricultural robotics, and self-driving cars. Cyrill Stachniss has coauthored over 220 scientific publications, he was senior editor of the IEEE Robotics and Automation Letters as well as associate editor of the IEEE Transactions on Robotics. He is a Microsoft Research Faculty Fellow and received the IEEE RAS Early Career Award in 2013. He is actively participating in German and European research projects. He is a spokesperson of the DFG Cluster of Excellence "PhenoRob - Robotics and Phenotyping for Sustainable Crop Production" at the University of Bonn. Furthermore, he acted as coordinator of the EC-funded project "ROVINA" for digitizing the Roman catacombs with robots and as the spokesperson of the DFG research unit "Mapping on Demand".

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com

[F] X. Chen, S. Li, B. Mersch, *et al.*, "Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 4, pp. 6529–6536, 2021. DOI: 10.1109/LRA.2021.3093567

IEEE Robotics and Automation Letters (RAL) paper presented at the
2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data

Xieyuanli Chen Shijie Li Benedikt Mersch Louis Wiesmann Jürgen Gall Jens Behley Cyrill Stachniss

***Abstract*—The ability to detect and segment moving objects in a scene is essential for building consistent maps, making future state predictions, avoiding collisions, and planning. In this paper, we address the problem of moving object segmentation from 3D LiDAR scans. We propose a novel approach that pushes the current state of the art in LiDAR-only moving object segmentation forward to provide relevant information for autonomous robots and other vehicles. Instead of segmenting the point cloud semantically, i.e., predicting the semantic classes such as vehicles, pedestrians, roads, etc., our approach accurately segments the scene into moving and static objects, i.e., also distinguishing between moving cars vs. parked cars. Our proposed approach exploits sequential range images from a rotating 3D LiDAR sensor as an intermediate representation combined with a convolutional neural network and runs faster than the frame rate of the sensor. We compare our approach to several other state-of-the-art methods showing superior segmentation quality in urban environments. Additionally, we created a new benchmark for LiDAR-based moving object segmentation based on SemanticKITTI. We published it to allow other researchers to compare their approaches transparently and we furthermore published our code.**

***Index Terms*—SLAM, Deep Learning Methods**

I. Introduction

THE ability to identify which parts of the environment are static and which ones are moving is key to safe and reliable autonomous navigation. It supports the task of predicting the future state of the surroundings, collision avoidance, and planning. This knowledge can also improve and robustify pose estimation, sensor data registration, and simultaneous localization and mapping (SLAM). Thus, accurate and reliable moving object segmentation (MOS) in sensor data at frame rate is a crucial capability supporting most autonomous mobile systems. Depending on the application domain and chosen sensor setup, moving object segmentation can be a challenging task.

In this work, we address the problem of moving object segmentation in 3D LiDAR data at sensor frame rate in urban environments. Instead of detecting all *theoretically movable* objects such as vehicles or humans, we aim at separating the *actually moving* objects such as driving cars from static or non-moving objects such as buildings, parked cars, etc. See Fig. 1 for an example scene and our segmentation. Moving objects are colored in red. We propose a novel approach based on convolutional neural networks (CNNs) to explicitly address the MOS problem for 3D LiDAR scans. We exploit range images as an intermediate representation, which is a natural representation of the scan from a rotating 3D LiDAR such as a Velodyne or Ouster sensors. Based on this comparably light-weight representation, we can directly exploit the existing range-image-based semantic segmentation networks as proposed by Milioto et al. [19], Cortinhal et al. [8], and Li et al. [16] to tackle the MOS problem. Most of such existing LiDAR-based semantic segmentation networks predict the semantic labels of a point cloud, e.g. vehicle, building, road, etc. They do not distinguish between actually moving objects, like moving cars, and static objects, like parked cars and also buildings, etc. We are making this distinction and are exploiting sequences of range images, allowing for an effective moving object segmentation targeted to autonomous vehicles. Our main application focus is perception for self-driving cars in outdoor scenes, but the method itself is not restricted to this domain.

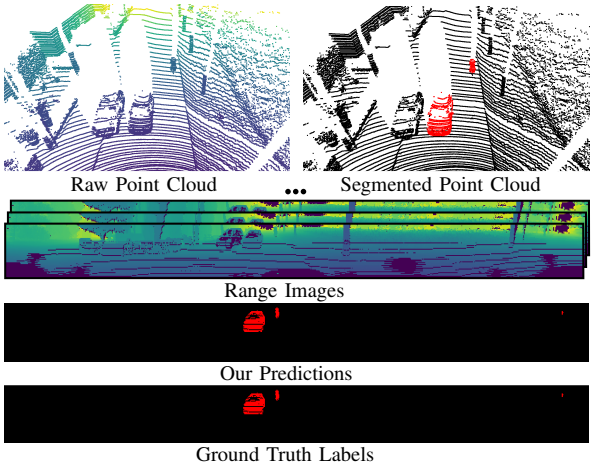


Fig. 1: Moving object segmentation using our approach. Our method can detect and segment the currently moving objects given sequential point cloud data exploiting its range projection. Instead of detecting all *potentially movable* objects such as vehicles or humans, our approach distinguishes between *actually moving* objects (labeled in red) and static or non-moving objects (black) in the upper row. At the bottom, we show the range image and our predictions in comparison to the ground truth labels.

This paper's main contribution is a novel method based on CNNs using range images generated from 3D LiDAR scans together with the residual images generated from past scans as inputs and outputs for each range measurement in the current frame a label indicating if it belongs to a moving object or not. By combining range images and residual images our

Manuscript received: February 24, 2021; Revised May 19, 2021; Accepted June 17, 2021. This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments.

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob, under GA 1927/5-2 (FOR 2535), by the EC within Horizon Europe, grant agreement no. 101017008 (Harmony), and by the Chinese Scholarship Committee. All authors are with the University of Bonn, Germany. xieyuanli.chen@igg.uni-bonn.de

Digital Object Identifier (DOI): 10.1109/LRA.2021.3093567.

Copyright ©2021 IEEE

network exploits the temporal information and can differentiate between moving and static objects as shown in Fig. 1. For training, we reorganize the SemanticKITTI [3] dataset and merge the original labels into two classes, moving and static, by exploiting the existing annotations of moving traffic participants. Furthermore, our approach runs faster than the sensor frame rate, i.e., 10 Hz for a typical rotating 3D LiDAR sensor. Comparisons with multiple existing methods suggest that the proposed approach leads to more accurate moving object segmentation. In sum, we make two key claims: First, our approach is able to achieve moving object segmentation using only 3D LiDAR scans and runs faster than the sensor frame rate of 10 Hz. Second, it improves the moving object segmentation performance by incorporating residual images in addition to the current scan and outperforms several state-of-the-art networks. To allow for as easy as possible comparisons and support future research, we propose and release a moving object segmentation benchmark [1], including a hidden test set, based on the SemanticKITTI dataset and we release the source code of our approach [2]. We also provide a short video[3] illustrating the capabilitiy of our method.

II. Related Work

While there has been a large interest in vision-based [18], [2], [21], radar-based [1] and vision and LiDAR combined [32], [23] moving object segmentation approaches, we concentrate here on approaches using only LiDAR sensors. Below, we distinguish between map-based and map-free approaches.

Map-based approaches. Most of the existing LiDAR-based approaches target the cleaning of a point cloud map. These methods mostly run offline and rely on a prebuilt map. Some methods use time-consuming voxel ray casting and require accurately aligned poses to clean the dense terrestrial laser scans [13], [25]. To alleviate the computational burden, visibility-based methods have been proposed [22], [31]. These types of methods associate a query point cloud to a map point within a narrow field of view, e.g. cone-shaped used by Pomerleau et al. [22]. Recently, Pagad et al. [20] propose an occupancy map-based method to remove dynamic points in LiDAR scans. They first build occupancy maps using object detection and then use the voxel traversal method to remove the moving objects. Kim et al. [15] propose a range-image-based method, which exploits the consistency check between the query scan and the pre-built map to remove dynamic points and uses a multi-resolution false prediction reverting algorithm to refine the map. Even though such map-based methods can separate moving objects from the background, they need a pre-built and cleaned map and therefore usually can not achieve online operation.

Map-free approaches. Recently, LiDAR-based semantic segmentation methods operating only on the sensor data have achieved great success [19], [8], [16]. Wang et al. [29] tackle the problem of segmenting things that could move from 3D laser scans of urban scenes, e.g. cars, pedestrians, and bicyclists. Ruchti and Burgard et al. [24] also propose a learning-based method to predict the probabilities of potentially movable objects. Dewan et al. [10] propose a LiDAR-based scene flow method that estimates motion vectors for rigid bodies. Based on that, they developed both semantic classification and segmentation methods [11], [9], which exploit the temporally consistent information from the sequential LiDAR scans. Bogoslavskyi and Stachniss [5] propose a class-agnostic segmentation method for 3D LiDAR scans that exploits range images to enable online operation and leads to more coherent segments, but does not distinguish between moving and non-moving objects.

Semantic segmentation can be seen as a relevant step towards moving object segmentation. Most existing semantic segmentation methods, however, only find *movable* objects, e.g. vehicles and humans, but do not distinguish between actually moving objects, like driving cars or walking pedestrians, and non-moving/static objects, like parked cars or building structures. The most similar work to ours is the one by Yoon et al. [33], which also detects dynamic objects in LiDAR scans without using a pre-built map. It exploits heuristics, e.g. the residual between LiDAR scans, free space checking, and region growing to find moving objects. There are also multiple 3D point cloud-based semantic segmentation approaches [28], [27], [26], which also perform well in semantic segmentation tasks. Among them, Shi et al. [26] exploit sequential point clouds and predict moving objects. However, based on networks operating directly on point clouds, these methods are usually heavy and difficult to train. Furthermore, most of them are both time-consuming and resource-intensive, which might not be applicable for autonomous driving.

Our method is also based on neural networks and we investigate the usage of three recent range projection-based semantic segmentation methods proposed by Milioto et al. [19], Cortinhal et al. [8], and Li et al. [16] to tackle MOS with the prospect of real-time capability and operation beyond the frame rate of the LiDAR sensor. Our method does not rely on a pre-built map and operates online, i.e., uses only LiDAR scans from the past. We exploit residuals between the current frame and the previous frames as an additional input to the investigated semantic segmentation networks to enable class-agnostic moving object segmentation. Note that the proposed architecture does not depend on a specific range projection-based semantic segmentation architecture. By training the network with proposed new binary masks, our method distinguishes between moving cars and parked cars in an end-to-end fashion.

III. Our Approach

The goal of our approach is to achieve accurate and fast moving object segmentation (MOS) for LiDAR scans to enable autonomous mobile systems to make decisions in a timely manner. Fig. 2 shows a conceptual overview of our proposed method. To achieve online MOS, we first project the point clouds into range image representation (see Sec. III-A). To separate moving and non-moving objects, we then exploit

[1] http://bit.ly/mos-benchmark
[2] https://github.com/PRBonn/LiDAR-MOS
[3] https://youtu.be/NHvsYhk4dhw

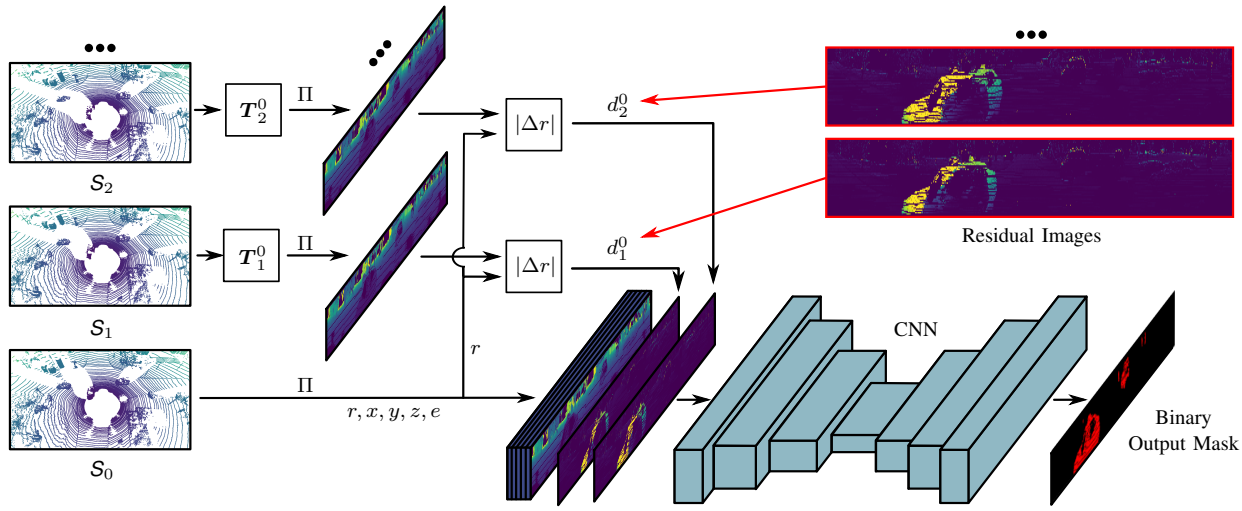


Fig. 2: Overview of our method. We use the range projection-based representation of LiDAR scans to achieve online moving object segmentation. Given the current scan S_0, we generate residual images from previous scans $\{S_i\}_{i=1}^{N}$ to explore the sequential information. This is by transforming them to the current viewpoint with a homogeneous transformation matrix T_i^0 estimated from a SLAM or sensor-based odometry, projecting them to the range representation with a mapping Π and subtracting them from the current scan's range image. The residual images are then concatenated with the current range image and used as input to a fully convolutional neural network. Trained with the binary labels, the proposed method can separate moving and static objects.

sequential information (see Sec. III-B) computing residuals between the current and the previous scans (see Sec. III-C). We finally concatenate them together with the range information as the input for a segmentation CNN (see Sec. III-D). In addition, we propose a novel MOS benchmark based on SemanticKITTI (see Sec. III-E) to train and evaluate MOS methods.

A. Range Image Representation

In line with prior work [19], [16], [7], we use a range projection of a point cloud to generate an image representation. Specifically, we convert each LiDAR point $\boldsymbol{p} = (x, y, z)$ via a mapping $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates, and finally to image coordinates, as defined by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y,x)\,\pi^{-1}\right] & w \\ \left[1 - \left(\arcsin(z\,r^{-1}) + \mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right] & h \end{pmatrix}, \quad (1)$$

where (u, v) are image coordinates, (h, w) are the height and width of the desired range image representation, $\mathrm{f} = \mathrm{f}_{\mathrm{up}} + \mathrm{f}_{\mathrm{down}}$ is the vertical field-of-view of the sensor, and $r = ||\boldsymbol{p}_i||_2$ is the range of each point. This procedure results in a list of (u, v) tuples containing a pair of image coordinates for each $\boldsymbol{p}_i$. Using these indices, we extract for each $\boldsymbol{p}_i$, its range r, its x, y, and z coordinates, and its remission e, and store them in the image. Thus, each pixel can store more than only a range. Consequently, we can easily exploit extra information and add this as extra channels. Therefore, we can directly feed this information to existing networks without changing the architectures, which makes our method easily transferred to other new architectures. To show this capability, we test our method with three different segmentation networks in this paper.

B. Sequence Information

We aim at segmenting moving objects online, i.e., only using the current and recent LiDAR scans, such that one can exploit the information for odometry estimation in a SLAM pipeline and potentially remove dynamics from a map representation. We assume that we are given a time series of N LiDAR scans in the SLAM history, denoted by $S_j = \{\boldsymbol{p}_i \in \mathbb{R}^4\}$ with M points represented as homogeneous coordinates, i.e., $\boldsymbol{p}_i = (x, y, z, 1)$. We denote the current LiDAR scan by S_0 and the sequence of N previously scans by S_j with $1 < j < N$. The *estimated* N consecutive relative transformations from the SLAM / odometry approach, $T_N^{N-1}, \ldots, T_1^0$, between the $N+1$ scanning poses, represented as transformation matrices, i.e., $T_k^l \in \mathbb{R}^{4\times4}$, are also assumed to be available. Given the estimated relative poses between consecutive scans, we can transform points from one viewpoint to another. We denote the k^{th} scan transformed into the l^{th} scan's coordinate frame by

$$S^{k\to l} = \{T_k^l \boldsymbol{p}_i | \boldsymbol{p}_i \in S_k\}, \quad (2)$$

with $T_k^l = \prod_{j=k}^{l+1} T_j^{j-1}$.

C. Residual Images

Inspired by Wang et al. [30], who exploit the difference between RGB video frames for action recognition, we propose to use LiDAR-based residual images together with pixel-wise binary labels on the range image to segment moving objects. Combining the current sensor reading and residual images, we can employ existing segmentation networks to distinguish between pixels on moving objects and background by leveraging the temporal information inside the residual images.

To generate the residual images and later fuse them into the current range image, transformation, and re-projection are required. To realize this, we propose a three-step procedure: First, we compensate for the ego-motion by transforming the previous scans into the current local coordinate system given

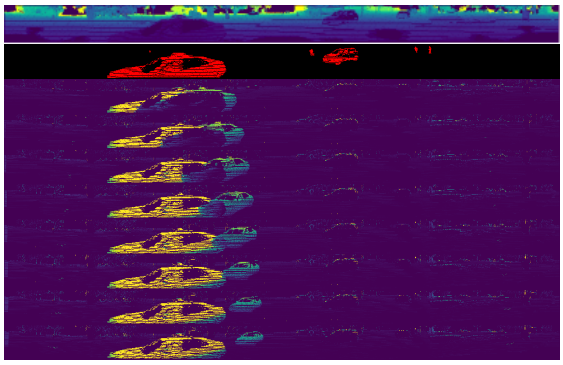


Fig. 3: Residual images, where j means the residual image generated between the current frame and the last j-th frame. We can see the continuous discrepancy in the residual images due to the motion of the moving car.

the transformation estimates as defined in Eq. (2). Next, the transformed past scans $\mathcal{S}^{k\to l}$ are re-projected into the current range image view using Eq. (1). We compute the residual $d^l_{k,i}$ for each pixel i by computing the normalized absolute difference between the ranges of the current frame and the transformed frame by

$$d^l_{k,i} = \frac{|r_i - r_i^{k\to l}|}{r_i}, \tag{3}$$

where r_i is the range value of $\boldsymbol{p}_i$ from the current frame located at image coordinates (u_i, v_i) and $r_i^{k\to l}$ is the corresponding range value from the transformed scan located at the same image pixel. We only calculate the residual for the valid pixels that contain measurements and set the residual to zero for the invalid pixels. Examples of such residual images are depicted in Fig. 3. We can see that due to the motion of objects in the scene, e.g. the moving car, the displacement between these points in the common viewpoint is relatively large compared to the static background. However, there are ambiguities, since the large residual patterns appear twice for one moving object, while for the slowly moving objects the residual patterns are not obvious. Therefore, directly using residual images for moving object segmentation does not lead to a great performance. It, however, provides a valuable cue for moving objects and can guide the network to separate moving and non-moving objects.

In the end, the residual images are concatenated with the current range image as extra channels where range image provides spatial information and residual image encodes temporal information. Each pixel (u_i, v_i) in the fused range image then contains a vector of different types of information $(x_i, y_i, z_i, r_i, e_i, d^0_{1,i}, ..., d^0_{j,i}, ..., d^0_{N-1,i})$, where d^0_j is the residual image calculated between the last j^{th} frame and the current frame.

D. Range Projection-based Segmentation CNNs

In this paper, we do not design a new network architecture but reuse networks that have been successfully applied to LiDAR-based semantic segmentation in the past. We adopt and evaluate three popular networks, namely SalsaNext [8], RangeNet++ [19], and MINet [16], for MOS. SalsaNext and RangeNet++ are encoder-decoder architectures with a solid performance and MINet uses a light-weight and efficient multi-path architecture. After the segmentation, a fast GPU-based k-Nearest-Neighbor search over the point cloud is used to remove artifacts produced by the range projection [19]. All methods are state-of-the-art range projection-based LiDAR semantic segmentation networks, comparably light-weight, and can achieve real-time operation, i.e., run faster than the frame rate of the employed LiDAR sensor, which is 10 Hz for common Ouster and Velodyne scanners. For more detailed information about each network, we refer to the original papers [19], [8], [16].

Instead of changing the architecture of these segmentation networks, we directly feed them with the fused range images plus the residual information, retrain the network and evaluate their performance with our MOS benchmark proposed in Sec. III-E. Using our proposed residual image approach, all segmentation networks show a large improvement in moving object segmentation as shown in Sec. IV-A. For training, we use the same loss functions as used in the original segmentation methods, while mapping all classes into two per-point classes, moving and non-moving.

E. Moving Object Segmentation Benchmark

Large datasets for LiDAR-based odometry, object detection, and tracking, like the KITTI Vision Benchmark [14], and semantic segmentation, panoptic segmentation, and scene completion like SemanticKITTI [3] are available and widely used. There are, however, not many datasets and benchmarks available for 3D LiDAR-based moving object segmentation. With this work, we also aim at covering this gap with a novel benchmark task for MOS.

Our proposed MOS benchmark is based on SemanticKITTI. It uses the same split for training and test set as used in the original odometry dataset, where sequences 00 to 10 are used for training and sequences 11 to 21 are used as a test set. SemanticKITTI contains in total 28 semantic classes such as vehicles, pedestrians, buildings, roads, etc. and distinguishes between moving and non-moving vehicles and humans. In the proposed MOS benchmark, we manually reorganize all the classes into only two types: moving and non-moving/static objects. The actually moving vehicles and humans belong to moving objects and all other classes belong to the non-moving/static objects.

For quantifying the MOS performance, we use the commonly applied Jaccard Index or intersection-over-union (IoU) metric [12] over moving objects, which is given by

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}, \tag{4}$$

where TP, FP, and FN correspond to the number of true positive, false positive, and false negative predictions for the moving class.

IV. Experimental Evaluation

This paper focuses on moving object segmentation from 3D LiDAR scan sequences. We present our experiments to

show the capabilities of our method and to support our key claims, that our approach: (i) achieves moving object segmentation using only 3D LiDAR scans and runs faster than the sensor frame rate of 10 Hz and (ii) improves the moving object segmentation performance by using residual images, and outperforms several state-of-the-art networks.

We evaluate all the methods on our proposed MOS benchmark. We use the odometry information provided by SemanticKITTI, which are estimated with a LiDAR-based SLAM system, SuMa [4]. Aiming at an easy-to-integrate algorithm, we stick to the original setup while only changing the input and the output of the classification head into the proposed binary labels. We train each network using their specific training hyperparameters over 150 epochs on sequences 00-07 and 09-10 and keep sequence 08 as the validation set. For more details on the training regime for each network, we refer to the original papers [19], [8], [16].

A. Ablation Study on Input and Architecture

The first ablation study presented in this section is designed to support our claim that our approach is able to achieve moving object segmentation using only 3D LiDAR scans. All the experiments in this section are evaluated on the validation set, i.e., sequence 08.

We test three different setups with three different networks, RangeNet++, SalsaNext, and MINet, for moving object segmentation as shown in Tab. I. The first setup is to train the three range projection-based networks directly with the labels for moving and non-moving classes. The second setup is to attach the previous frames to the current frame as the input of the network resulting in 2×5 input channels, as each image contains the coordinates (x, y, z), the range r, and the remission e for each pixel. The third setup is to concatenate the proposed residual images to the current frame as the input of the network and therefore the input size is $5{+}N$, as detailed in Sec. III-C.

As can be seen in Tab. I, RangeNet++ and SalsaNext show a basic performance while MINet fails when training the network together with the binary labels and no additional inputs. Overall, the performance has space for improvements. This is probably due to the fact, that from one frame, the semantic segmentation networks cannot distinguish well between the moving and static objects from the same general class, but may learn some heuristics, e.g. that cars on the road are usually moving while those on parking lots are static, which can also be seen in the qualitative results Fig. 5. A reason why MINet fails may be due to the lightweight architecture that is not capable of learning such heuristics.

In the second setup, we directly combine two frames. Here, the networks can already gain some improvements in MOS, since they can obtain the temporal information from two scans. In this setting, MINet is also capable of predicting moving objects. In the third setup, the best MOS performance is achieved. We hypothesize that it is advantageous to give direct access to the residual information instead of the full range views. Given that most of the points are redundant in two successive frames and the input is large due to the concatenation, the networks need less time to extract the temporal information if the residuals are provided directly. While a large enough network should be able to learn concepts as the difference between frames given enough time, it is generally advantageous to directly provide this information as also shown by Milioto et al. [19].

TABLE I: Evaluating our method with three different networks

Input	RangeNet++	MINet	SalsaNext
One frame	38.9	9.1	51.9
Two frames	40.6	35.0	56.0
Residual frames (N=1)	40.9	38.9	59.9

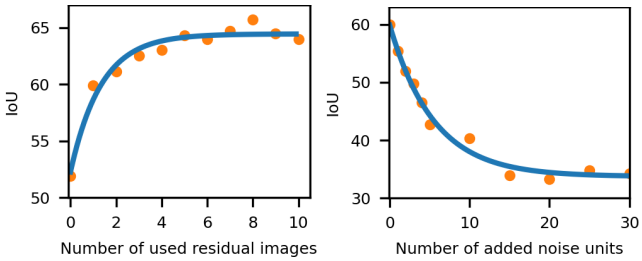


Fig. 4: Ablation studies. The left figure shows the ablation study on the MOS performance vs. the number of residual images N. The right figure shows the ablation study on the MOS performance vs. the number of added noise units to the poses during the inferring.

As shown in Fig. 4, we provide two further ablation studies using SalsaNext as the segmentation network. The left figure shows an ablation study on the number of residual images used for MOS Both ablation studies use SalsaNext as the segmentation network. We can see that $N = 1$ residual image attains the biggest improvement in terms of MOS performance, while adding more residual images improves the MOS performance further with diminishing returns for $N > 8$ residual images. The figure on the right shows an ablation study on the MOS performance vs. the amount of noise added to the relative odometry poses used to generate the residual images. We manually add noise to the the poses estimated by SLAM in (x, y, yaw) with a unit of $(0.1\,m, 0.1\,m, 1^\circ)$ to see how the pose estimations influence our method during inferring. As can be seen, the MOS performance will drop due to the noisy poses. However, when the added noises are larger than 20 units, $(2\,m, 2\,m, 20^\circ)$, the network may ignore the noisy residual images and the MOS performance will not become worse.

B. MOS Performance and Comparisons

The experiment presented in this section investigates the MOS performance of our approach. It supports the claim that our approach improves the MOS performance by using residual images and outperforms several state-of-the-art networks. Since there are not many existing implementations for LiDAR-based MOS available, we choose several methods that have been used in similar tasks, e.g., semantic segmentation and scene flow, and modify them to achieve LiDAR-based MOS. All the methods are evaluated on the test data of the proposed benchmark, i.e., sequences 11-21.

TABLE II: MOS performance compared to the state of the art.

	IoU
SalsaNext (moveable classes)	4.4
SalsaNext (retrained)	46.6
Residual	1.9
Residual + RG	14.1
Residual + RG + Semantics	20.6
SceneFlow	4.8
SceneFlow + Semantics	28.7
SqSequence	43.2
KPConv	60.9
Ours (based on SalsaNext/N = 1)	52.0
Ours (based on SalsaNext/N = 8 + Semantics)	**62.5**

We analyze multiple alternative approaches. We start using an existing semantic segmentation network, e.g. SalsaNext [8], directly and label all the movable objects, e.g. vehicles and humans, as moving objects while labeling other objects as static. We name this method as *SalsaNext (movable classes)*. Here, we also show the results generated by the retrained SalsaNext with the proposed binary labels, named as *SalsaNext (retrained)*. Since the residual images can already point out rough positions of moving objects, here we also take it as a simple heuristic-based baseline, named as *Residual*. Inspired by Yoon et al. [33], we also re-implement the pure geometric heuristic-based method using residual information together with free space checking and region growing, named as *Residual+RG*.

We furthermore compare our method also to the state-of-the-art scene flow method, FlowNet3D [17], referred to as *SceneFlow*, which is a network estimating the translational flow vector for every LiDAR point given two consecutive scans as input. We set a threshold on the estimated translation of each point to decide the label for each point, i.e., points with translations larger than the threshold are labeled as moving. We fix the threshold based on the best MOS performance on the validation set. We also compare our method to the state-of-the-art multiple point cloud-based semantic segmentation methods [28], [26], since they can also distinguish between moving and non-moving classes.

For the non-semantic-based methods, we additionally add semantic information by checking if the predicted moving objects are movable or not, and only label a point as moving if it is both predicted as moving by the original method and at the same time assigned to a movable object, e.g. vehicles and humans. The semantic information is generated using SalsaNext with the pre-trained weights provided by the original paper. We identify the semantic-enhanced methods by adding *+Semantics*.

We compare two setups of our method to all the above-mentioned methods. For our methods, we choose SalsaNext as the base network as it shows the best performance in our ablation study. In the first setup, we use only one residual image, $N = 1$, to obtain the temporal information, and in the other setup, we use our best setup fixed on the validation sequence with $N = 8$ residual images and semantic information to see the best performance of our method.

As shown in Tab. II, our residual image-based method with $N = 1$ already outperforms most baselines, while being worse than KPConv, which is a dense multiple point clouds-based semantic segmentation method. Due to heavy computation burden, it cannot achieve real-time performance. When our method uses multiple residual images ($N = 8$) together with semantic information, our method outperforms all other methods.

TABLE III: KITTI Odometry Benchmark Results

Split	Approach		
	SuMa	SuMa++	SuMa+MOS
Train (Seq. 00-10)	0.36/0.83	0.29/0.70	**0.29/0.66**
Test (Seq. 11-21)	0.34/1.39	0.34/1.06	**0.33/0.99**

Relative errors averaged over trajectories of 100 to 800 m length: relative rotational error in degrees per 100 m / relative translational error in %.

Fig. 5 and Fig. 6 show the qualitative results on range images and LiDAR scans respectively in a very challenging situation, where the car is at the intersection and there are both a lot of moving objects and static objects. Our method can distinguish moving and static points even when some of the moving objects are moving slowly and other methods fail to detect this.

C. Applications

Two obvious applications of our proposed method are LiDAR-based odometry/SLAM as well as 3D mapping. Here, we show the effectiveness of our method by using the MOS predictions as masks for the input LiDAR scans, which removes effectively all points belonging to moving objects. No further tweaks have been employed. We use our best setup for the MOS, i.e., our approach extending SalsaNext with $N = 8$ residual images and semantics.

1) Odometry/SLAM: For the LiDAR-based odometry experiments, we use an off-the-shelf SLAM approach [4] and apply our MOS method before feeding the point cloud into the SLAM pipeline. We compare the improved odometry results to both the original approach, called SuMa, and our semantic-enhanced approach, SuMa++ [6]. We evaluate these odometry methods, SuMa, SuMa++, and SuMa+MOS on the KITTI odometry benchmark [14].

The quantitative results are shown in Tab. III. We can see that, by simply applying our MOS predictions as a pre-processing mask, the odometry results are improved in both the KITTI training and test data and even slightly better than the well-designed semantic-enhanced SuMa.

2) 3D Mapping: As shown in Fig. 7, we compare the aggregated point cloud maps (a) directly with the raw LiDAR scans, (b) with the cleaned LiDAR scans by applying our MOS predictions as masks. We use the Open3D library [34] to visualize the mapping results. As can be seen, there are moving objects present that pollute the map, which might have adversarial effects, when used for localization or path planning. By using our MOS predictions as masks, we can effectively remove these artifacts and get a clean map. Note that, here we show two direct use cases of our MOS approach without any further optimizations employed.

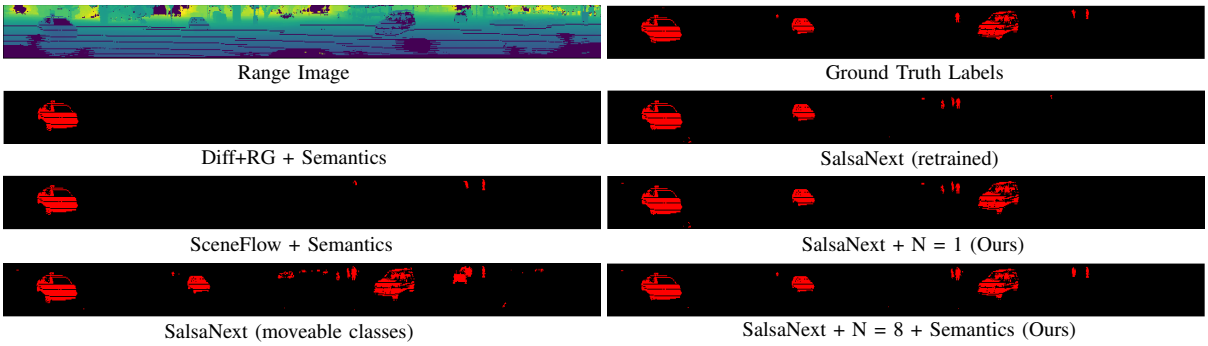
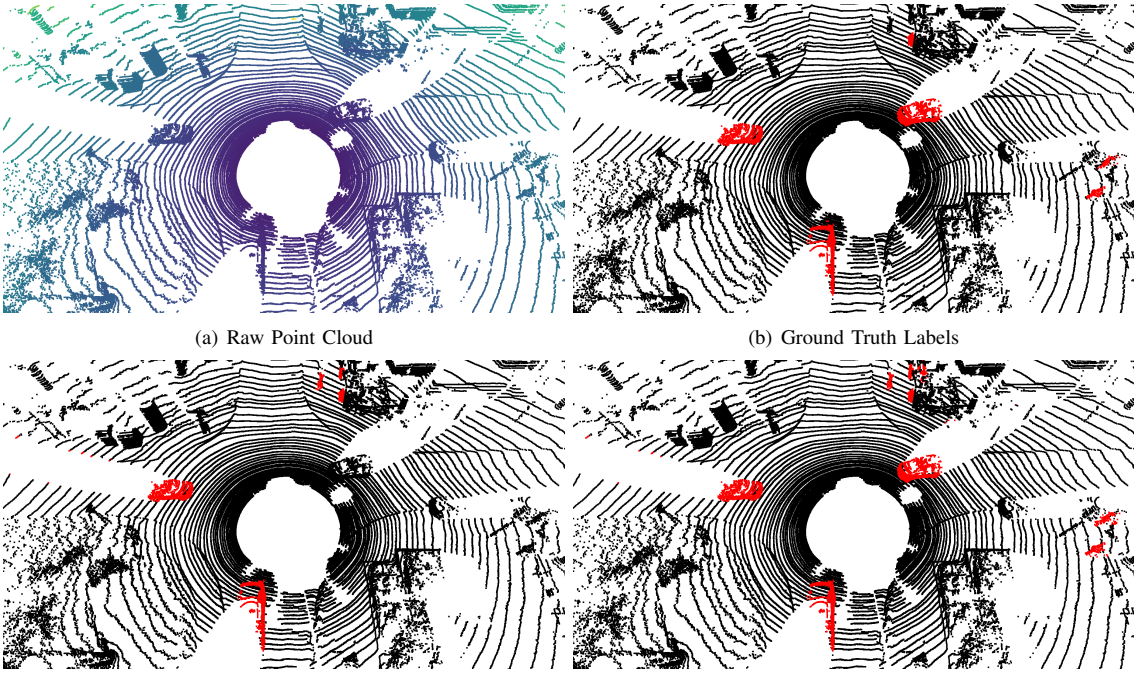


Fig. 5: Qualitative results with range projections, where red pixels correspond to moving objects.



(a) Raw Point Cloud (b) Ground Truth Labels

(c) SalsaNext (retrained) (d) SalsaNext+ N = 8 + Semantics (Ours)

Fig. 6: Qualitative results shown as point clouds. (a) shows the raw point cloud with points colored depending on the range from purple (near) to yellow (far). (b) shows the ground truth, and (c,d) prediction results, where red points correspond to the class moving.

D. Runtime

The runtime is evaluated on sequence 08 with an Intel i7-8700 with 3.2 GHz and a single Nvidia Quadro P6000 graphic card. It takes around 10 ms on average to estimate the odometry and generate the residual image. Since we only change the input of each network while keeping the architecture the same, the inference time is nearly the same as before, specifically 75 ms for RangeNet++, 42 ms for SalsaNext, and 21 ms for MINet. In case of using semantics for the MOS, we can run a second full semantics network in parallel.

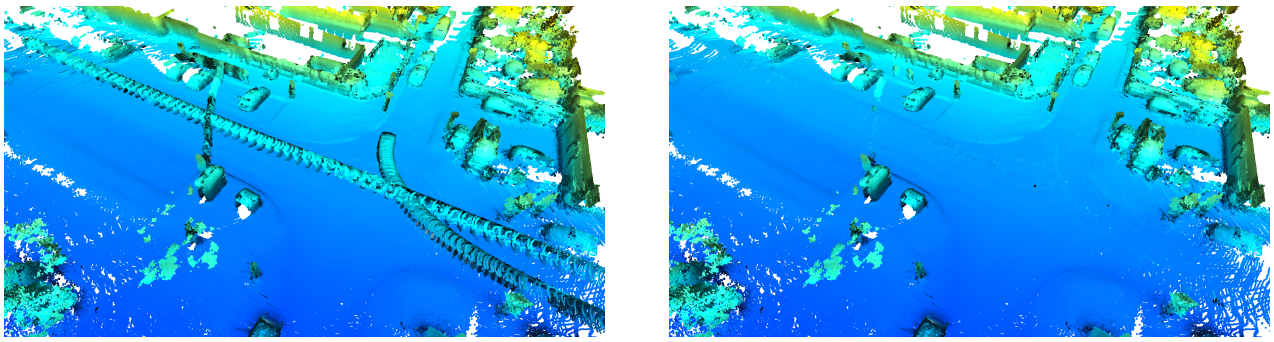
As the odometry history for the SLAM is available, we need to estimate the pose and generate the residual images only once for every incoming frame. In sum, using our method for LiDAR-based odometry takes approx. 51 ms per scan (=20 Hz) using SalsaNext, which is faster than the frame rate of a typical LiDAR sensor, i.e., 10 Hz.

V. Conclusion

In this paper, we presented a novel and effective approach to achieve LiDAR-based moving object segmentation in an end-to-end online fashion. Our method exploits neural networks and sequential information, which allows our approach to successfully distinguish between moving and static objects. Our method is based on range projections and thus fast and can directly improve existing SLAM and mapping systems. The experiments suggest that our method achieves good performance on MOS and outperformed several state-of-the-art approaches. We also propose a new benchmark for LiDAR-based MOS and used it to evaluate our approach, also allowing further comparisons with future MOS systems.

Acknowledgments

Thanks to Thomas Läbe, Ignacio Vizzo, Nived Chebrolu, Tiziano Guadagnino, and Haofei Kuang for fruitful discussions and early support.



(a) Raw point clouds (b) Point clouds with moving segments removed

Fig. 7: Mapping Results on Sequence 08, Frame 3960-4070, where we show the accumulated point cloud (a) without removing segments and (b) when we remove the segments predicted as moving.

REFERENCES

[1] R. Aldera, D. De Martini, M. Gadd, and P. Newman. Fast radar motion estimation with a learnt focus of attention using weak supervision. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, 2019.

[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 *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, 2018.

[3] 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 Intl. Conf. on Computer Vision*, 2019.

[4] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems*, 2018.

[5] I. Bogoslavskyi and C. Stachniss. Efficient online segmentation for sparse 3d laser scans. *Photogrammetrie – Fernerkundung – Geoinformation (PFG)*, pages 1–12, 2017.

[6] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2019.

[7] X. Chen, I. Vizzo, T. Läbe, J. Behley, and C. Stachniss. Range Image-based LiDAR Localization for Autonomous Vehicles. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, 2021.

[8] T. Cortinhal, G. Tzelepis, and E.E. Aksoy. SalsaNext: Fast, Uncertainty-Aware Semantic Segmentation of LiDAR Point Clouds. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2020.

[9] A. Dewan and W. Burgard. DeepTemporalSeg: Temporally Consistent Semantic Segmentation of 3D LiDAR Scans. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, pages 2624–2630, 2020.

[10] A. Dewan, T. Caselitz, G.D. Tipaldi, and W. Burgard. Rigid scene flow for 3d lidar scans. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 1765–1770, 2016.

[11] A. Dewan, G.L. Oliveira, and W. Burgard. Deep semantic classification for 3d lidar data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2017.

[12] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Intl. Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.

[13] J. Gehrung, M. Hebel, M. Arens, and U. Stilla. An approach to extract moving objects from mls data using a volumetric background representation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4, 2017.

[14] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. pages 3354–3361, 2012.

[15] G. Kim and A. Kim. Remove, then revert: Static point cloud map construction using multiresolution range images. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2020.

[16] S. Li, X. Chen, Y. Liu, D. Dai, C. Stachniss, and J. Gall. Multi-scale interaction for real-time lidar data segmentation on an embedded platform. *arXiv preprint arXiv:2008.09162*, 2020.

[17] X. Liu, C.R. Qi, and L.J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2019.

[18] C. McManus, W. Churchill, A. Napier, B. Davis, and P. Newman. Distraction suppression for vision-based pose estimation at city scales. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, 2013.

[19] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2019.

[20] S. Pagad, D. Agarwal, S. Narayanan, K. Rangan, H. Kim, and G. Yalla. Robust Method for Removing Dynamic Objects from Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, 2020.

[21] P.W. Patil, K.M. Biradar, A. Dudhane, and S. Murala. An end-to-end edge aggregation network for moving object segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020.

[22] F. Pomerleau, P. Krüsiand, F. Colas, P. Furgale, and R. Siegwart. Long-term 3d map maintenance in dynamic environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, 2014.

[23] G. Postica, A. Romanoni, and M. Matteucci. Robust moving objects detection in lidar data exploiting visual cues. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2016.

[24] P. Ruchti and W. Burgard. Mapping with dynamic-object probabilities calculated from single 3d range scans. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, 2018.

[25] 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*, 3(3):1679–1686, 2018.

[26] H. Shi, G. Lin, H. Wang, T.Y. Hung, and Z. Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020.

[27] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han. Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In *Proc. of the Europ. Conf. on Computer Vision*, 2020.

[28] H. Thomas, C. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, 2019.

[29] D.Z. Wang, I. Posner, and P. Newman. What could move? finding cars, pedestrians and bicyclists in 3d laser data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, 2012.

[30] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks for action recognition in videos. *IEEE Trans. on Pattern Analalysis and Machine Intelligence*, 41(11):2740–2755, 2018.

[31] W. Xiao, B. Vallet, M. Brédif, and N. Paparoditis. Street Environment Change Detection from Mobile Laser Scanning Point Clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 107:38–49, 2015.

[32] J. Yan, D. Chen, H. Myeong, T. Shiratori, and Y. Ma. Automatic Extraction of Moving Objects from Image and LIDAR Sequences. In *Proc. of the Intl. Conf. on 3D Vision*, 2014.

[33] D. Yoon, T. Tang, and T. Barfoot. Mapless online detection of dynamic objects in 3d lidar. In *Proc. of the Conf. on Computer and Robot Vision*, pages 113–120, 2019.

[34] Q. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[G] L. Wiesmann, A. Milioto, X. Chen, *et al.*, "Deep Compression for Dense Point Cloud Maps," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2060–2067, 2021. DOI: 10.1109/LRA.2021.3059633

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

Deep Compression for Dense Point Cloud Maps

Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, Jens Behley

***Abstract*— Many modern robotics applications rely on 3D maps of the environment. Due to the large memory requirements of dense 3D maps, compression techniques are often necessary to store or transmit 3D maps efficiently. In this work, we investigate the problem of compressing dense 3D point cloud maps such as those obtained from an autonomous vehicle in large outdoor environments. We tackle the problem by learning a set of local feature descriptors from which the point cloud can be reconstructed efficiently and effectively. We propose a novel deep convolutional autoencoder architecture that directly operates on the points themselves so that we avoid voxelization. Additionally, we propose a deconvolution operator to upsample point clouds, which allows us to decompress to an arbitrary density. Our experiments show that our learned compression achieves better reconstructions at the same bit rate compared to other state-of-the-art compression algorithms. We furthermore demonstrate that our approach generalizes well to different LiDAR sensors. For example, networks learned on maps generated from KITTI point clouds still achieve state-of-the-art compression results for maps generated from nuScences point clouds.**

I. Introduction

Maps are a key ingredient for robot navigation, involving tasks such as localization, planning, or scene understanding. Over the past fifteen years, a lot of robot navigation systems have moved from 2D to 3D representations. Moving from 2D to 3D substantially increases the required amount of memory. Particularly, if large-scale environments shall be represented, storing and especially transmitting it over network links becomes a challenge due to their large size.

Modern 3D LiDAR sensors, which are typically used on autonomous cars or mobile robots, generate millions of points in just a couple of seconds. Point clouds are naturally unordered and quite memory inefficient, but they represent the raw data that such sensors generate and are the main representation used, for example, for scan registration. Although upcoming 5G networks will enable higher data transfer rates, representing 3D point cloud data in a compact and structured way is key to handle the huge amount of data. Our approach focuses on effectively compressing 3D point cloud data collected with modern 3D LiDAR sensors in large, outdoor environments.

Representing 3D data in a memory-efficient way is a common problem in robotics and computer graphics and several approaches exist that tackle this challenge. Binary space partition approaches, like Octrees [20] or kD-trees [4], utilize a hierarchical data representation that allows for efficient storage and fast neighbor queries. Voxel grids [1] discretize the world enabling constant time access, however, at the cost of incurring a large memory footprint if a fine resolution is desired. This makes it inadequate for large-scale environments. To overcome this problem, Octomaps [12] store voxels in a sparse octree structure, thereby reducing the memory requirements. Other representations [18], [39] make strong assumptions that often do not hold in the real world. Another technique is to use triangular meshes to approximate the environment or use multiple primitives, like planes, cylinders, or spheres [35], but such predefined geometries may not fully exploit the available potential of recurring structures. Therefore, learning-based representations often try to capture common characteristics

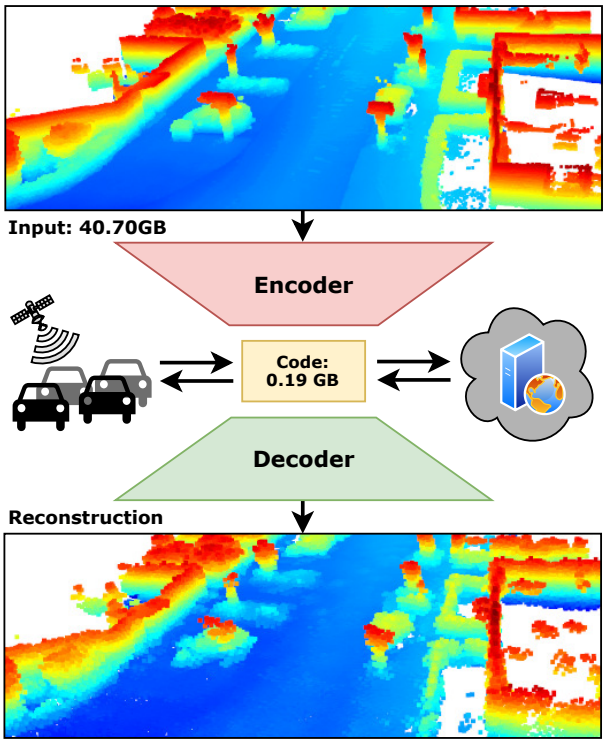


Fig. 1: 3D point clouds obtained by an autonomous car or robot require a large amount of memory. Transferring this data to a fleet of cars or sending it to some cloud service requires a compression of the data. Our approach can reconstruct dense point clouds from a highly compressed representation even when targeting low bit rates. The point cloud color indicates the height above ground ranging from blue (low) to red (high).

Manuscript received: October, 15, 2020; Revised January, 8, 2021; Accepted January, 25, 2021.

This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments. This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob.

All authors are with the University of Bonn, Germany. louis.wiesmann@igg.uni-bonn.de

of the scene for further compression [32]. Recent deep neural autoencoder networks [13], [15], [29] provide data compression for different domains and they have also been successfully used to compress 3D point cloud data. This work belongs to this class of approaches and proposes a novel architecture exploiting state-of-the-art representation learning for 3D point clouds to achieve high compression rates.

The main contribution of this paper is a novel deep learning-based approach for compressing 3D LiDAR point cloud data in a lossy fashion for large-scale outdoor environments (see Fig. 1 for an illustration).

Our approach exploits the occurrence of common structures through local feature descriptors. It learns a small and compact set of local feature descriptors that allows for compressing and reconstructing point cloud data. Our approach is end-to-end learnable and provides dense point clouds even when targeting low bit rate compression. Additionally, we propose a novel deconvolution for point clouds with feature propagation and integration. Our deconvolutional kernel operates directly on a set of points, which makes discretization unnecessary. We compare our approach with state-of-the-art compression techniques such as Draco [8] and the octree-based approach from Mekuria et al. [21]. In brief, we are able to provide higher quality maps after decompression than the state-of-the-art compression approaches at the same bit rate. Together with this paper, we plan to release the source code along with the trained models at `https://github.com/PRBonn/deep-point-map-compression`.

II. Related Work

In recent years, we witnessed an increasing interest in deep learning using point clouds from the computer vision and robotics community. Guo et al. [10] survey the field and here we concentrate on representation learning and point cloud compression using neural networks.

Deep Learning on Point Clouds. The seminal PointNet approach [26] uses multi-layer perceptrons (MLPs) followed by a pooling operation to extract an embedding for a whole unordered point cloud. The follow-up work PointNet++ [27] extends this idea to a hierarchical approach by applying the PointNet structure repeatedly to local neighborhoods for different scaling factors. Each layer downsamples the point cloud by iteratively choosing the furthest point from the already collected subset to keep a good coverage of the point cloud. Yongcheng et al. [43] extend this idea by a multi-level feature aggregation to increase the contextual information.

Grid-based approaches [19], [42] compute local features using 3D convolutions on a voxel grid. Octree-based approaches [30], [41] alleviate the large memory footprint of high-resolution voxel grids by using a hierarchical representation to efficiently represent free space. Our approach operates directly on the point cloud to avoid discretization errors and excessive memory usage.

For the extraction of local features, we use the convolutions proposed by Thomas et al. [38]. They define a continuous convolution for unordered point clouds using kernel points. A convolutional kernel learns a weight for each position in a set of kernel points. The 3D continuous weight space is an interpolation of these discrete positions. They design the network architecture for segmentation and classification which allows them to use skip connections to recover the points which get lost in the downsampling process. This is not feasible for compression, since this would require storing intermediate results of the encoder, which we want to avoid to efficiently compress the point clouds. Therefore, we define a continuous 3D deconvolution to recover the points based on their learned feature.

Yu et al. [44] propose a network to upsample point clouds using local feature matrices. In contrast to their hierarchical purely point-based approach, our deconvolution integrates and propagates features. We use multiple deconvolution layers which allows for more flexible upsampling rates at inference time.

Point Cloud Compression. Many works in the field of robotics and computer vision also focus on compressing point cloud data. Octrees [20] can efficiently store three-dimensional data and can be used as an efficient map representation which is commonly used in robotics [12], [6]. This already compact representation can be further compressed by using additional geometric heuristics [16], [34]. Huang et al. [14] use neural networks to learn a conditional probability model, followed by entropy coding to compress frequent symbols with fewer bits than rarer symbols. Octrees efficiently store three-dimensional coordinates but require additional methods to compress attributes [45]. Other approaches focus on an iterative prediction of neighboring points using spanning trees [9], [22]. Tree structures are very memory efficient in most real-world scenarios but do not exploit the full potential of recurring common objects.

The correlation between point clouds acquired from LiDAR or depth camera streams offers a further possibility to save memory. A range-image-based representation allows for the usage of image or video compression algorithms [24], [37], [40]. However, such a projective representation is not suitable for large dense maps captured from many locations, since it only represents the environment of the actual viewpoint well, but is not very generalizable.

Golla et al. [7] exploit standard image compression for static point clouds by storing oriented compressed height and occupancy images for local patches. Deep convolutional autoencoders can use a rate-distortion loss [28], [29] to ensure a good trade-off between quality and memory consumption. These approaches use grids and differentiable quantization instead of a combination of points and features. Similar to Huang et al. [15], we also propose an end-to-end learnable point cloud compression autoencoder network. In contrast to their approach to embed the information of the whole point cloud into a single feature, we store a set of local feature descriptors together with positions from which we can then restore the point cloud.

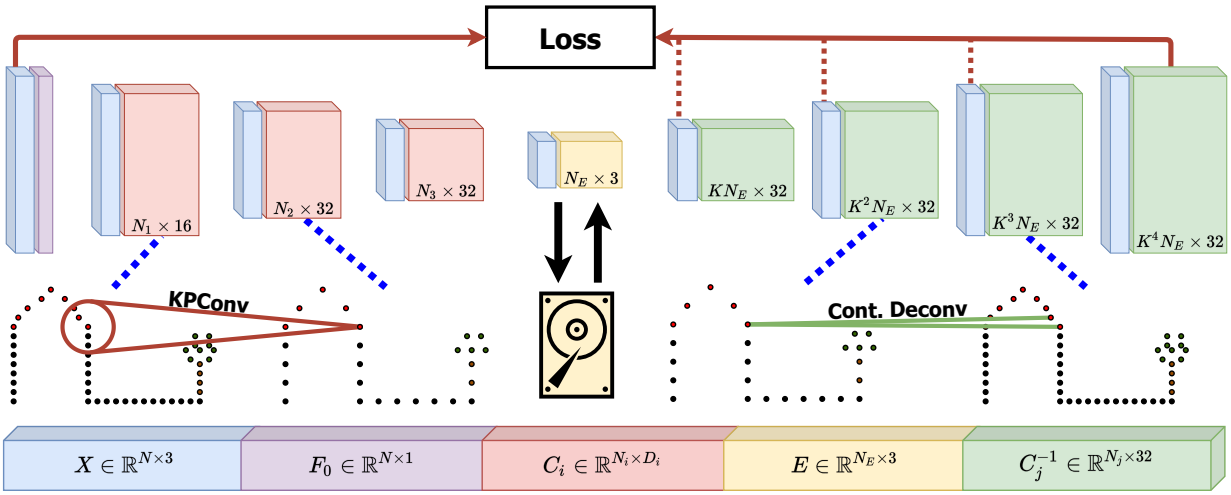


Fig. 2: Schematic overview of our proposed network. The encoder takes as input a point cloud $\mathcal{P} = \{\mathcal{X}, \mathcal{F}\}$, $|\mathcal{P}| = N_0$ and computes subsequentially for each subset a feature descriptor $C_i, i \in \{0, 1, 2\}$ (red) using kernel point convolutions. The last feature descriptor will be mapped (using an MLP) to the compact embedding space (yellow) to enable a memory-efficient representation $E \in \mathbb{R}^{N_E\times 3}$. This embedding E with its associated points can be used for storage or transmission and will later be used by the decoder to decompress the point cloud. The decoder consists of four deconvolutional layers which subsequentially upsample the point cloud (green). We use the feature of the last layer as a refinement of the last coordinates. The loss enforces a similar appearance of input and output. Additionally, we have a regularization term for each upsampled cloud (denoted by the dotted line) to be lower-resolution versions of the input.

III. Our Approach

Our goal is to learn for a small subset of points useful features from which we can recover the original point cloud. To this end, we propose an autoencoder structure for point cloud compression comprised of two parts. First, the encoder learns, from a given input data, a reduced and often more abstract representation – an embedding or code. The embedding is the input for the second part, the decoder that tries to reconstruct the original data using this compressed representation. By comparing the reconstruction with the original point cloud, the network can learn self-supervised parameters via backpropagation [33]. Compression is achieved as long as the embedding is smaller than the original point cloud.

Encoder Blocks. In our case, the input of the encoder is a dense 3D point cloud $\mathcal{P} = \{(\boldsymbol{x}_i, \boldsymbol{f}_i)\}$, $|\mathcal{P}| = N$, which consists of coordinates $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3\}$ and point features $\mathcal{F} = \{\boldsymbol{f}_i \in \mathbb{R}^D\}$. The encoder uses multiple convolutional layers to learn local geometric features for each point. In Fig. 2 the coordinates $\mathcal{X}$ are denoted in blue and the learned descriptors $\mathcal{F}$ in red. Each layer reduces the number of points while increasing the receptive field. We use kernel point convolutions (KPConvs) [38], which directly operate on the features of the points themselves to avoid discretization effects caused by using grid-based representations such as voxel grids or octrees. The convolution of F at a point $\boldsymbol{x}$ with a convolutional kernel g is defined as a weighted sum of its neighboring features as

$$(F * g)(\boldsymbol{x}) = \sum_{(\boldsymbol{x}_i, \boldsymbol{f}_i)\in\mathcal{N}(\boldsymbol{x})} g(\boldsymbol{x}_i - \boldsymbol{x})\boldsymbol{f}_i, \qquad (1)$$

where $\mathcal{N}(\boldsymbol{x}) = \{(\boldsymbol{x}_i, \boldsymbol{f}_i) \in \mathcal{P} \mid \|\boldsymbol{x}_i - \boldsymbol{x}\| < r\}$ are the neighbors of $\boldsymbol{x}$ within radius r. The convolutional weights $\boldsymbol{W}$ are defined on M kernel points from which the weight of the neighbors $\mathcal{N}(\boldsymbol{x})$ are interpolated using first-order splines of size σ

$$g(\boldsymbol{y}_i) = \sum_{m<M} \max\left(0, 1 - \frac{\|\boldsymbol{y}_i - \hat{\boldsymbol{x}}_m\|}{\sigma}\right) \boldsymbol{W}_m, \qquad (2)$$

with the relative coordinate $\boldsymbol{y}_i = \boldsymbol{x}_i - \boldsymbol{x}$ and the weight $\boldsymbol{W}_m \in \mathbb{R}^{D_{\text{in}}\times D_{\text{out}}}$ of the m-th kernel point $\hat{\boldsymbol{x}}_m$. In each encoding block, we downsample the previous point cloud using grid-based subsampling. This returns for each occupied voxel the point which is the nearest to its center. It provides us with a homogenous point distribution without the risk of losing all points in a certain sparser area (as random sampling) or incur long sampling times (as for furthest point sampling). We use the same ResNet-like blocks as Hugues et al. [38]. The identity shortcuts in ResNet blocks enable a more direct gradient flow to earlier layers to reduce the risk of encountering instabilities due to vanishing gradients [11]. In contrast to skip connections between encoder and decoder, no additional information needs to be stored for the decompression. The last layer consists of an MLP to compress the features of size $\mathbb{R}^{N\times D_{\text{out}}}$ to the desired dimension $\mathbb{R}^{N\times D_{\text{emb}}}$.

Decoder Blocks. The task of the decoder is to reconstruct the original data from the embedding. Most encoder-decoder networks use skip connections [31], [23] from the encoder to the decoder to keep the high-frequency information. For compression, skip connections cannot be used, since it would require storing additional data from the encoder blocks for usage in the decoder. Hence, the whole signal must be encoded in the embedding to achieve effective compression. Therefore, we present a decoder block that does not depend on any skip connections but rather estimates the lost coordinates themselves. For a given point $(\boldsymbol{x}_i, \boldsymbol{f}_i)$, we

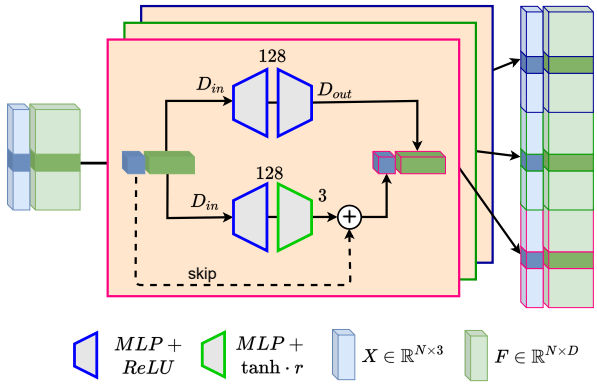


Fig. 3: Overview of our proposed deconvolution. Our deconvolutional kernel consists of two small MLP networks which serve as 1D convolutions. The first MLP (upper part) transforms the old feature $\boldsymbol{F}$ into a new feature space. The second MLP (lower part) predicts the offset to the new position $\boldsymbol{X}$ within the radius r. Depending on the number of deconvolutional kernels, in this example $K = 3$, we upsample the point cloud by a factor of K.

define the deconvolution C^{-1} by a set of K MLP layers, as depicted in Fig. 3. For each point, we obtain K new points $\{(\boldsymbol{x}_i^k, \boldsymbol{f}_i^k)\}, k \in \{0, \dots K-1\}$ within a given radius r by

$$\boldsymbol{x}_i^k = \boldsymbol{x}_i + r \cdot \Delta_k(\boldsymbol{f}_i), \tag{3}$$

$$\boldsymbol{f}_i^k = \Phi_k(\boldsymbol{f}_i). \tag{4}$$

Let us call Δ_k an offset block, which consists of an MLP with one hidden layer which uses a ReLU activation after the first layer and $\tanh$ after the second. The offset block Δ_k determines a coordinate increment by a nonlinear mapping of the feature space into the coordinate frame of the kernel $\Delta_k : \mathbb{R}^{D_{\text{in}}} \mapsto [-1, 1]^3$. The feature block $\Phi_k : \mathbb{R}^{D_{\text{in}}} \mapsto \mathbb{R}^{D_{\text{out}}}$ computes new features based on the old descriptor. Similar to the offset block Δ_k, the feature block Φ_k is also an MLP with one hidden layer but utilizes two ReLU activations instead. K offset and feature blocks form the proposed deconvolution $C^{-1} : \{\mathbb{R}^{N\times 3}, \mathbb{R}^{N\times D_{\text{in}}}\} \mapsto \{\mathbb{R}^{KN\times 3}, \mathbb{R}^{KN\times D_{\text{out}}}\}$ which will be applied to each point $\{(\boldsymbol{x}_i, \boldsymbol{f}_i)\}$ of the current layer. This upsamples the point cloud by a factor of K.

Architecture. Fig. 2 provides an overview of our proposed network architecture. We use 3 encoding blocks (KPConv) with grid-based subsampling followed by one MLP layer for compressing the embedding. Experiments show that a deeper encoder architecture leads to a lower decompression quality. We explain this behavior by vanishing gradients due to the absence of skip connections between the encoder and decoder. The radius of the convolution is equal to the resolution r_s of the sampling grid. This ensures that all points are part of at least one convolutional operation.

We estimate the relation between the grid resolution r_s and the sub-sampling rate $A(r_s)$ to upsample the point clouds to their original size

$$A(r_s) = \frac{a}{r_s^b}. \tag{5}$$

Further derivation and reasoning of the chosen power function (with the parameters a and b) are provided in Sec. IV.

We use the grid resolution r_s to control the compression rate. The sparser we sample, the higher the compression will be and the more points we need to upsample. Each KPConv has $3^3 = 27$ kernel points arranged in a grid with an influence radius of $\sigma = r/2$. The feature dimension (the red blocks in Fig. 2) for the KPConvs is $16, 32, 32$ respectively, and 3 for the embedding (yellow block). We saw that these relative compact feature descriptors are sufficient to store the geometric information about the neighborhood. If the original point cloud has no feature, we then use the occupancy value ($\boldsymbol{f}_i = 1$) as advocated by Thomas et al. [38]. The decoder consists of 4 deconvolutions to upsample the embedding to its original size. We use 32-dimensional point features and 128-dimensional hidden layer spaces in the deconvolution (expressed through the green blocks in Fig. 2). The upsampling factor K_i of the deconvolutional kernels is adapted to the sampling rate (see Sec. IV-E). Since the feature of the last layer is unused otherwise, we feed it to an MLP ($\mathbb{R}^{32} \mapsto \mathbb{R}^3$) to refine the coordinates.

Loss Function. We want to restore a point cloud, which is as similar as possible to the input. We use the Chamfer distance D_{CD} as a measure of similarity in our loss function $\mathcal{L}$. It is the average symmetric squared distance $\bar{d}^2$ of each point to its nearest neighbor in the other point cloud

$$D_{\text{CD}}(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}}) = \frac{\bar{d}^2(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}})}{2} + \frac{\bar{d}^2(\mathcal{P}_{\text{out}}, \mathcal{P}_{\text{in}})}{2}, \tag{6}$$

$$\bar{d}^2(\mathcal{P}_i, \mathcal{P}_j) = \frac{1}{|\mathcal{P}_i|} \sum_{\boldsymbol{x}_i \in \mathcal{P}_i} \min_{\boldsymbol{x}_j \in \mathcal{P}_j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \tag{7}$$

where $\mathcal{P}_{\text{in}}$ denotes the input point cloud before compression and $\mathcal{P}_{\text{out}}$ the decompressed point cloud. Using the symmetric distance prevents the network from flooding the whole space with points but also from leaving out parts that have been present in the original point cloud. We add the Chamfer distances between the input point cloud and the output points $\hat{\mathcal{P}}$ of all deconvolutions as a regularization term to ensure valid intermediate results. The loss $\mathcal{L}$ is then given by:

$$\mathcal{L} = D_{\text{CD}}(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}}) + \beta \sum_j D_{\text{CD}}(\mathcal{P}_{\text{in}}, \hat{\mathcal{P}}_j), \tag{8}$$

where β is a weight to control the impact of the regularization term and we use $\beta = 0.2$ in all of our experiments. For a more detailed analysis of the regularization, see Sec. IV-E.

IV. Experimental Evaluation

In this section, we validate that our proposed algorithm is able to compress point cloud data efficiently. We compare our method to Draco [8] and the octree-based compression algorithm by Mekuria et al. [21], which we will refer to as "MPEG anchor". Additionally, we will show the results for a plain Octree [20] using binary flags to indicate if a leaf octant is occupied.

A. Experimental Setup

Our network is designed for compressing large-scale dense point clouds. We evaluate our method on the SemanticKITTI [2] dataset. For obtaining high accurate and

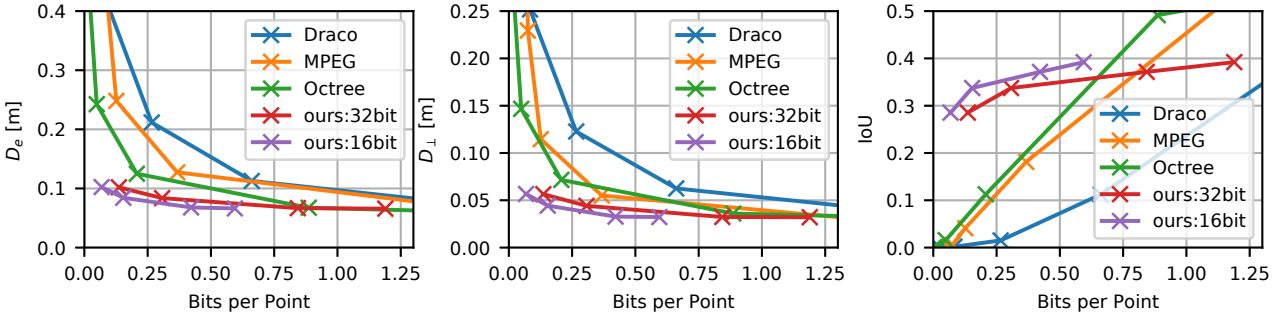


Fig. 4: Compression results on the test sequence 08 of the KITTI Vision Benchmark dataset. We use Draco [8], the MPEG Anchor from Mekuria et al. [21] and an own binary Octree implementation as baselines. Our approach can reconstruct the point cloud for the same amount of memory at a higher quality level.

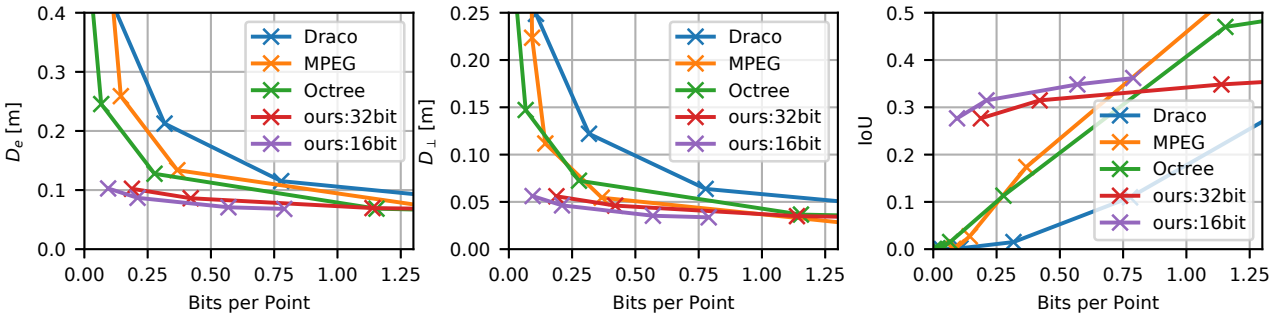


Fig. 5: Compression results on the nuScenes dataset. The model was still trained on SemanticKITTI to show the generalizability of our approach. The errors are slightly higher compared to the validation set of KITTI, but we are still able to outperform the baselines.

dense point clouds, we aggregate all scans using the ground truth poses and divide the map into patches of size $40 \times 40 \times 15\,\mathrm{m}^3$. The labels have been used to remove the dynamic objects which otherwise will lead to artifacts in the map. To remove redundant points in the original point cloud, we filter it using a voxel grid with a resolution of $10\,\mathrm{cm}^3$. We use sequence 00 to 10 (except 08) for training (see Fig. 6 first column). A small subset of the training data serves as a validation set and the complete sequence 08 is used for testing and comparison with the baselines.

Evaluation metrics. The quality of a compression algorithm is a trade-off between compression ratio and reconstruction error. For the compression ratio, we use the average bits per point (BPP) required for storing the encoding of the point cloud. We use 3 metrics for measuring the reconstruction error. Symmetric point cloud distances D_d are widely used for measuring the quality of the point cloud reconstruction. These metrics consist of two parts: the distance $\bar{D}_d$ from the ground truth point cloud $\mathcal{P}_{\text{in}}$ to the reconstruction $\mathcal{P}_{\text{out}}$ and vise versa

$$D_d(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}}) = \frac{\bar{D}_d(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}})}{2} + \frac{\bar{D}_d(\mathcal{P}_{\text{out}}, \mathcal{P}_{\text{in}})}{2}, \quad (9)$$

$$\bar{D}_d(\mathcal{P}_i, \mathcal{P}_j) = \frac{1}{|\mathcal{P}_i|} \sum_{\boldsymbol{x}_i \in \mathcal{P}_i} \min_{\boldsymbol{x}_j \in \mathcal{P}_j} d(\boldsymbol{x}_i - \boldsymbol{x}_j). \quad (10)$$

Thereby, the metric is sensitive to false positives (reconstructing points in unoccupied areas) and false negatives (leaving out occupied areas). The euclidean distance $D_e = D_d(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}})$ with $d = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$ is used as a metric for reconstructing the points itself. However, for some robotics applications (e.g. point-to-plane ICP), it is less important to reconstruct the exact same point than that it lies on the same surface. Therefore, we also show the symmetric plane distance $D_\perp = D_d(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}})$ with $d = |\boldsymbol{n}^T(\boldsymbol{x}_i - \boldsymbol{x}_j)|$. Where $\boldsymbol{n} \in \mathbb{R}^3$ denotes the ground truth normal of that point. The normals have been precomputed using the Eigenvalue decomposition of the covariance matrix from all points within a $50\,\mathrm{cm}$ radius around the query point.

The last metric is the intersection-over-union (IoU) between occupancy grids G for both point clouds. The IoU is here defined as

$$IoU = \frac{|G_{\text{in}} \cap G_{\text{out}}|}{|G_{\text{in}} \cup G_{\text{out}}|}. \quad (11)$$

The occupancy grids G_{in} and G_{out} have a resolution of $20 \times 20 \times 10\,\mathrm{cm}^3$ as used by Huang et al. [14].

Implementation details. We use the octree implementation by Behley et al. [3] for an efficient radius neighbor search in the KPConv-blocks. Our model is implemented in PyTorch [25] and trained on a GeForce RTX 2080 SUPER and with an Intel CPU with 3.5 GHz. We use the Adam optimizer [17] and the one cycle learning rate schedule proposed by Smith et al. [36] with a start learning rate of 10^{-6}, which will increase to 10^{-4} in the first 20 epochs and afterward decrease to 10^{-5}. We use the cosine annealing strategy and train for 200 epochs with a batch size of 3. We limit the number of input points to 30,000 points to speed-up the training and reduce the memory footprint while training. Nevertheless, we compare at test time our reconstructed point cloud to all available points. We train four

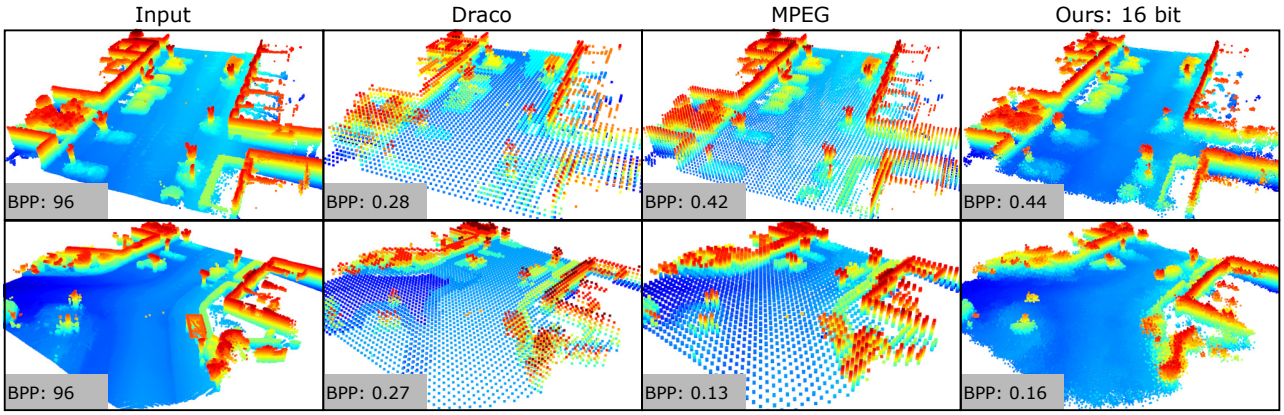


Fig. 6: Qualitative results of our proposed method and the baselines for different bits per point (BPP) on two example point clouds. The points are colored according to their height. Our approach is able to recover dense point clouds also when targeting low bit rates.

network architectures with varying sub-sampling resolutions $r_s \in \{3.0\,\text{m}, 2.0\,\text{m}, 1.2\,\text{m}, 1.0\,\text{m}\}$ for different compression levels. The encoder needs approximately 0.27 MB and the decoder between 0.30 MB and 0.62 MB memory storage (depending on the upsampling rates), which is a one-time investment and not depending on the number of compressed maps or points. We use Open3D [46] for visualization.

B. Compression Results

In this first experiment, we compare our compression results to the baselines to quantify the compression performance of our proposed method. Our approach stores the output of the encoder, namely a set of features and points, as the compressed representation. We will show the results for storing the embedding as 32 and 16 bit floating-point values. The compression results of our approach and the baselines on sequence 08 are presented in Fig. 4. Our proposed method outperforms the baselines in the distance-based metrics D_e and $D_\perp$, as well as in the IoU for the 16 bit representation. The small quality gain ($< 2\%$) of the 32 bit representation is disproportional concerning doubling the memory demand. The reduced density of the baselines leads to substantially higher errors than our approach which can reconstruct for each bit rate the same number of points. Our approach achieves over 2.4 times lower reconstruction errors for bit rates under 0.1 BPP compared to the baselines. We think that the worse performance of the MPEG Anchor [21] compared to the plain occupancy octree is due to some overhead for attribute compression and the ability to further compress tele-immersive data streams.

C. Generalization Capability

Learning-based methods often degrade when applied in a different environment due to over-fitting to the specific characteristics of the training set. We claim that our method generalizes well by learning the local geometries rather than remembering the global shape. To support this claim, we test our approach on a completely different dataset without retraining the model. Here, we use the nuScenes dataset [5], which not only has a different sensor setup (different height, field-of-view, and the usage of 32 beams instead of a 64-beam LiDAR), it is also recorded on a different continent which usually changes the appearance of the scenes quite substantially. Note that there are no labels for the nuScenes dataset available so that the dynamics will not be removed from the map, which makes it more challenging due to new artifacts it has never seen before. Fig. 5 shows that we still outperform the baselines for the 16 bit representation, even though the margin to the baselines got smaller.

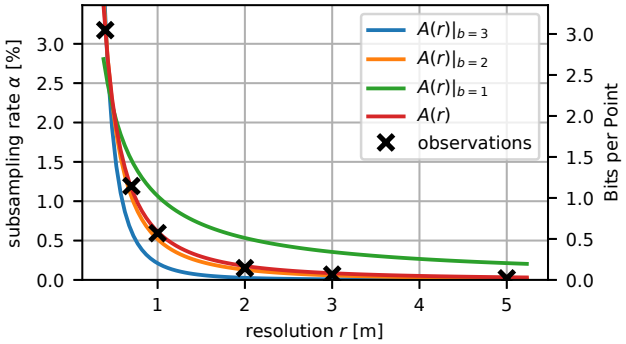


Fig. 7: We estimate the dependency between the subsampling rate α and the subsampling resolution r to predict the bit rate and to estimate the necessary upsampling rate for the deconvolution. We use least squares fitting to estimate the parameters of a power function $A(r) = a \cdot r^{-b}$ with parameters a and b.

D. Qualitative Analysis

In this part, we will analyze the decompression results qualitatively. In Fig. 6, we show the decompressed maps of the baselines and our approach. For the baselines, we have chosen the quality levels with the closest bit rates to our approach. The points are colored according to their height. As we can see, our approach is able to recover comparably dense point clouds even for varying compression rates. The point clouds of the baselines are sparser, especially when targeting low bit rates (see Fig. 6 second row). When reducing the bit rate, the decompressed maps of our approach get noisier while the baselines show larger quantization errors. Structures like the curbs of the streets are only visible in our denser maps.

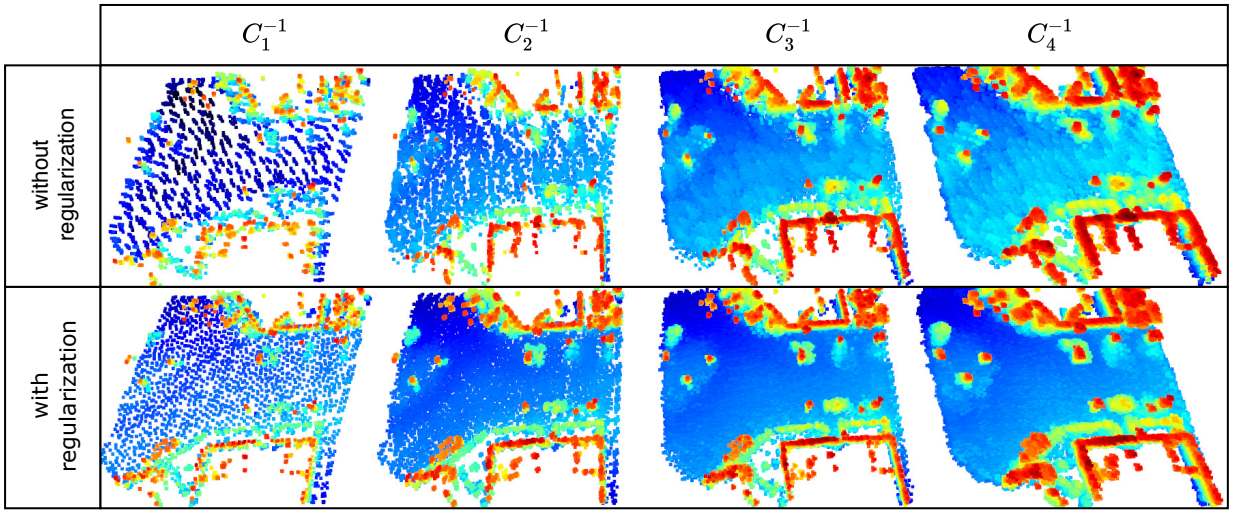


Fig. 8: Intermediate point clouds after each deconvolutional layer $C_j^{-1}, j \in \{1, 2, 3, 4\}$ for the same networks but with different loss functions. The network of the top is trained without the regularization term, whereas the bottom row is learned with the regularization term. The lack of the regularization leads to less homogeneous and more noisy point clouds after each deconvolution. When trained with the regularization, the intermediate point clouds can be used as lower resolution model.

E. Ablation Studies

In this part, we provide ablation studies to validate the choices made and to give a deeper understanding of the behavior of the network. We first investigate the adaptive sampling strategy and then the regularization.

Adaptive Sampling. Each encoding block reduces the number of points by grid-based subsampling and thus the number of embedding points depends on the resolution of the sampling grid. In this experiment, we investigate the influence of varying grid resolutions r_s and the subsampling rate α of the points. This enables us to adapt the up-sampling rate to ensure that the input and output point clouds have similar sizes. Additionally, it enables us to predict the bit rate BPP based on the resolution r_s or vise versa. Therefore, we subsample the point clouds of the training set with different grid resolutions r_s and compute the mean subsampling rate α. We fit a power function $A(r_s)$ which is proportional to the density distribution $\rho(r_s)$ of the point cloud

$$A(r_s) = \frac{a}{r_s^b} \propto \rho(r_s). \tag{12}$$

The parameter a denotes the magnitude of the density while the parameter b describes the dimensionality of the point distribution. A fix parameter $b \in \{1, 2, 3\}$ would correspond to a voluminous, planar, or linear distribution, respectively. The result of $A(r_s)$ as well as for fix $b \in \{1, 2, 3\}$ are shown in Fig. 7. The best fitted function is given by $a = 0.006$ and $b = 1.80$. The parameter $b = 1.80$ reflects the huge amount of planar surfaces (streets, walls, meadows...) in outdoor environments. Each deconvolutional block i up-samples the point cloud by the factor $K_i = \lceil A(r_e)^{-1/I} \rceil$, where I is the number of deconvolutional blocks and r_e is the sub-sampling resolution of the last encoding layer.

Impact of regularization. In this experiment, we show the importance of the regularization term. We train the same network two times, first with and second without regularization. In Fig. 8, we can see the qualitative impact on the point clouds after each upsampling step. The distribution of the points is more uniform and less noisy for the point clouds of the regularized network. The regularization term penalizes big point distances between the clouds. This leads implicitly to more homogeneous regions so that we do not see the necessity of an additional repulsion loss as in the PU-Net [44]. The quantitative results in Tab. I support our assumption that more meaningful intermediate point clouds help the network to reconstruct the data. Additionally, the intermediate point clouds can, with regularization, be used as lower-resolution versions of the point cloud.

TABLE I: Quantitative regularization impact

experiment	D_e	$D_\perp$	IoU
without regularization	0.110	0.062	0.327
with regularization	0.077	0.039	0.332

V. Conclusion

In this work, we presented a novel approach for lossy point cloud compression using a neural network. Our approach operates on dense large-scale maps of aggregated point clouds used for autonomous driving. Our method exploits recurring structures to learn a small set of feature descriptors using a deep convolutional autoencoder. The descriptor set serves as the compressed representation, which can be used for efficient storage and transmission, or later by the decoder to reconstruct the point cloud. Furthermore, we propose a 3D deconvolution that directly operates on the points themselves to avoid discretization effects from using voxel grids or memory size problems from skip connections. This allows us to successfully reduce the memory footprint of dense point clouds. We implemented and evaluated our approach on different datasets and provided comparisons to other existing techniques and supported all claims made in this paper.

REFERENCES

[1] T. Akenine-Möller, E. Haines, and N. Hoffmann. *Real-time Rendering*. A K Peters/CRC Press, 4th edition, 2018.

[2] 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 Intl. Conf. on Computer Vision (ICCV)*, 2019.

[3] J. Behley, V. Steinhage, and A.B. Cremers. Efficient radius neighbor search in three-dimensional point clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3625–3630, 2015.

[4] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] 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. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] J. Elseberg, D. Borrmann, and A. Nüchter. One billion points in the cloud – an octree for efficient processing of 3D laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 76:76–88, 2013.

[7] T. Golla and R. Klein. Real-time point cloud compression. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5087–5092, 2015.

[8] Google. Draco 3d data compression. *https://github.com/google/draco*, 2017. (26.08.2020).

[9] S. Gumhold, Z. Kami, M .Isenburg, and H-P. Seidel. Predictive point-cloud compression. In *ACM SIGGRAPH Sketches*, page 137. 2005.

[10] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 2020.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 34:189–206, 2013.

[13] Y. Hu, W. Yang, Z. Ma, and J. Liu. Learning End-to-End Lossy Image Compression: A Benchmark. *arXiv preprint arXiv:2002.03711*, 2020.

[14] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun. OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1313–1323, 2020.

[15] T. Huang and Y. Liu. 3D point cloud geometry compression on deep learning. In *Proc. of the ACM Intl. Conf. on Multimedia*, pages 890–898, 2019.

[16] Y. Huang, J. Peng, C-C. Kuo, and M. Gopi. Octree-Based Progressive Geometry Coding of Point Clouds. *In Proc. of the Symp. on Point-Based Graphics (PBG)*, 6:103–110, 2006.

[17] D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2016.

[18] I. Kweono, M. Hebert, E. Krotkov, and T. Kanade. Terrain mapping for a roving planetary explorer. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 997–1002, 1989.

[19] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[20] D. Meagher. Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer. *Rensselaer Polytechnic Institute Image Processing Laboratory, Technical Report*, (IPL-TR-80-111), 1980.

[21] R. Mekuria, K. Blom, and P. Cesar. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Trans. on Circuits and Systems for Video Technology (TCSVT)*, pages 828–842, 2016.

[22] B. Merry, P. Marais, and J. Gain. Compression of dense and regular point clouds. In *Proc. of the Intl. Conf. on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 15–20, 2006.

[23] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[24] F. Nenci, L. Spinello, and C. Stachniss. Effective Compression of Range Data Streams for Remote Robot Operations using H.264. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, Natalia N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 8026–8037. 2019.

[26] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] C.R. Qi, K. Yi, H. Su, and L.J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[28] M. Quach, G. Valenzise, and F. Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *Proc. of the IEEE Intl. Conf. on Image Processing (ICIP)*, pages 4320–4324, 2019.

[29] M. Quach, G. Valenzise, and F. Dufaux. Improved Deep Point Cloud Geometry Compression. *Proc. of the IEEE Intl. Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2020.

[30] G. Riegler, A. Ulusoy, and A. Geiger. OctNet: Learning Deep 3D Representations at High Resolutions. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.

[32] M. Ruhnke, B. Steder, G. Grisetti, and W. Burgard. Unsupervised learning of compact 3d models based on the detection of recurrent structures. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2137–2142, 2010.

[33] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. *Learning Representations by Back-Propagating Errors*, pages 696–699. 1988.

[34] R. Schnabel and R. Klein. Octree-based Point-Cloud Compression. In *Proc. of the Symposium on Point-Based Graphics*, 2006.

[35] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. In *Computer Graphics Forum*, pages 214–226, 2007.

[36] L. Smith and N. Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019.

[37] X. Sun, H. Ma, Y. Sun, and M. Liu. A novel point cloud compression algorithm based on clustering. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):2132–2139, 2019.

[38] H. Thomas, C R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L.J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[39] R. Triebel, P. Pfaff, and W. Burgard. Multi-level Surface Maps for Outdoor Terrain Mapping and Loop Closing. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[40] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda. Point cloud compression for 3D LiDAR sensor using recurrent neural network with residual blocks. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3274–3280, 2019.

[41] P. Wang, Y. Liu, Y. Guo, C. Sun, and X. Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Trans. on Graphics (TOG)*, 36(4):1–11, 2017.

[42] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[43] L. Yongcheng, F. Bin, M. Gaofeng, L. Jiwen, X. Shiming, and P. Chunhong. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 5239–5248, 2019.

[44] L. Yu, X. Li, C. Fu, D. Cohen-Or, and P. Heng. Pu-net: Point cloud upsampling network. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2799, 2018.

[45] C. Zhang, D. Florencio, and C. Loop. Point cloud attribute compression with graph transform. In *Proc. of the IEEE Intl. Conf. on Image Processing (ICIP)*, pages 2066–2070, 2014.

[46] Q. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[H] M. Aygun, A. Osep, M. Weber, *et al.*, "4D Panoptic LiDAR Segmentation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. DOI: 10.1109/CVPR46437.2021.00548

This CVPR 2021 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

4D Panoptic LiDAR Segmentation

Mehmet Aygün[1*] Aljoša Ošep[1*] Mark Weber[1] Maxim Maximov[1]
Cyrill Stachniss[2] Jens Behley[2] Laura Leal-Taixé[1]
[1]Technical University of Munich, Germany [2]University of Bonn, Germany
[1]{mehmet.ayguen, aljosa.osep, leal.taixe, mark-cs.weber, maxim.maximov}@tum.de
[2]{firstname.lastname}@igg.uni-bonn.de

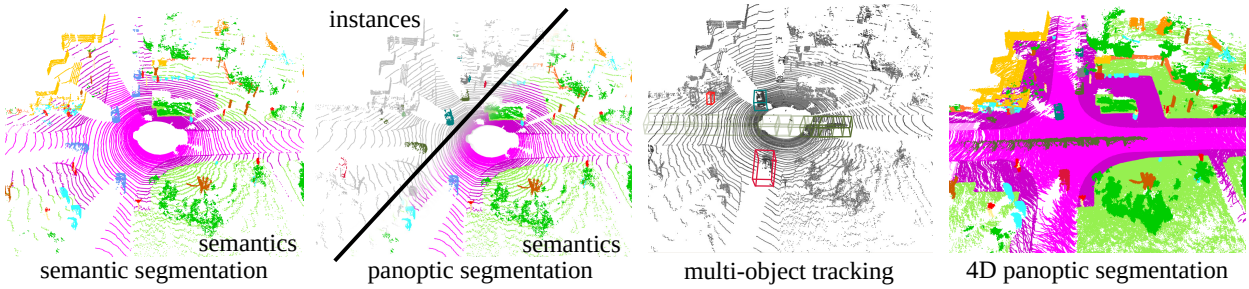


Figure 1: Types of LiDAR-based scene understanding. Semantic and panoptic segmentation assign semantic classes and determine instances in 3D space. Multi-object tracking encompasses 3D object detection in space, followed by association over time. 4D panoptic LiDAR segmentation jointly tackles semantic and instance segmentation in 3D space over time.

Abstract

Temporal semantic scene understanding is critical for self-driving cars or robots operating in dynamic environments. In this paper, we propose 4D panoptic LiDAR segmentation to assign a semantic class and a temporally-consistent instance ID to a sequence of 3D points. To this end, we present an approach and a point-centric evaluation metric. Our approach determines a semantic class for every point while modeling object instances as probability distributions in the 4D spatio-temporal domain. We process multiple point clouds in parallel and resolve point-to-instance associations, effectively alleviating the need for explicit temporal data association. Inspired by recent advances in benchmarking of multi-object tracking, we propose to adopt a new evaluation metric that separates the semantic and point-to-instance association aspects of the task. With this work, we aim at paving the road for future developments of temporal LiDAR panoptic perception.

1. Introduction

Spatio-temporal interpretation of raw sensory data is important for autonomous vehicles to understand how to interact with the environment and perceive how trajectories of moving agents evolve in 3D space and time.

In the past, different aspects of dynamic scene understanding such as semantic segmentation [22, 18, 48, 72, 85, 69], object detection [23, 62, 40, 65, 64, 66], instance segmentation [28], and multi-object tracking [43, 8, 53, 10, 76, 61, 58] have been tackled independently. The developments in these fields were largely fueled by the rapid progress in deep learning-based image [37] and point-set representation learning [59, 60, 72], together with contributions of large-scale datasets, benchmarks, and unified evaluation metrics [44, 22, 25, 19, 17, 74, 26, 5, 18, 13, 68]. In the pursuit of image-based holistic scene understanding, recent community efforts have been moving towards convergence of tasks, such as multi-object tracking (MOT) and segmentation [74, 83], and semantic and instance segmentation, *i.e.*, panoptic segmentation [35]. Recently, panoptic segmentation was extended to the video domain [34]. Here, the dataset, task formalization, and evaluation metrics focused on interpreting short and sparsely labeled video snippets in 3D (2D image+time) in an offline setting. Autonomous vehicles, however, need to continuously interpret sensory data and localize objects in a 4D continuum.

Tackling sequence-level LiDAR panoptic segmentation is a challenging problem, since state-of-the-art methods [72] usually need to downsample even single-scan point clouds to satisfy the memory constraints. Therefore, the common approach in (3D) multi-object tracking is detect-

* Authors contributed equally.

ing objects in individual scans, followed by temporal association [24, 76, 77], often guided by a hand-crafted motion model. In this paper, we take a substantially different approach, inspired by the unified space-time treatment philosophy. We form overlapping 4D volumes of scans (see Fig. 1) and, in parallel, assign to 4D points a semantic interpretation while grouping object instances jointly in 4D space-time.

Importantly, these 4D volumes can be processed in a single network pass, and the temporal association is resolved implicitly via clustering. This way, we retain inference efficiency while resolving long-term association between overlapping volumes based on the point overlap, alleviating the need for explicit data association.

For the evaluation, we introduce a point-centric higher-order tracking metric, inspired by recent metrics for multi-object tracking [45] and concurrent work on video panoptic segmentation [75] which differ from the available metrics [35, 9] that overemphasize the recognition part of the tasks. Our metric consist of two intuitive terms, one measuring the semantic aspect and second the spatio-temporal association of the task. Together with the recently proposed SemanticKITTI [5, 6] dataset, this gives us a test bed to analyze our method and compare it with existing LiDAR semantic/instance segmentation [40, 72, 76, 47] approaches, adapted to the sequence-level domain.

In summary, our **contributions** are: (i) we propose a unified space-time perspective to the task of 4D LiDAR panoptic segmentation, and pose detection/segmentation/tracking jointly as point clustering which can effectively leverage the sequential nature of the data and process several LiDAR scans while maintaining memory efficiency; (ii) we adopt a point-centric evaluation protocol that fairly weights semantic and association aspects of this task and summarizes the final performance with a single number; (iii) we establish a test bed for this task, which we use to thoroughly analyze our model's performance and the existing LiDAR panoptic segmentation methods used in conjunction with a tracking-by-detection mechanism. Our code, experimental data[1] and benchmark[2] are publicly available.

2. Related Work

Our work is related to tasks covering different aspects of scene perception, such as semantic segmentation, object detection/segmentation, and tracking. In the following, we review related methods and tasks.

Datasets and Metrics. The growing interest in autonomous vehicles has sparked interest in scene perception using LiDAR sensors. Here the progress has been fueled by advances in deep learning on point sets [59, 60, 31, 36, 38, 79, 80, 70, 72, 47] and datasets with standardized benchmarks for 3D semantic/instance segmentation [25, 5] and 3D object detection and multi-object tracking [13, 68]. This confirms the importance of advancing both spatial and temporal aspects of mobile robot perception. Our proposed task formulation and evaluation metric is the first that unifies both aspects to the best of our knowledge.

Recent community efforts in the field of image-based perception have been moving towards the convergence of different tasks. For instance, Kirillov *et al*. [35] proposed to unify semantic and instance segmentation, which they termed panoptic segmentation, together with an evaluation metric, the panoptic quality (PQ). Others proposed to tackle multi-object tracking and instance segmentation (MOTS) in videos jointly [74, 83]. Moreover, [34] recently extended panoptic segmentation to videos – however, the dataset and the evaluation metrics focus on interpreting short and sparse video snippets offline. This is reflected in the evaluation metric, which is essentially PQ evaluated based on the 3D IoU [83] and averaged over temporal windows of varying sizes to compensate that the difficulty of the task depends on the sequence length. This setting is not suitable for autonomous vehicles that need to interpret raw sensor data continuously. Hurtado *et al*. [32] proposes to combine ideas from MOTA [9] and PQ [35] by adding a penalty related to ID switches to the PQ. Nonetheless, both PQ and MOTA were criticized [57, 45], and the proposed evaluation inherits all of their well-known issues.

In this paper, we propose a different approach and bring ideas recently introduced in the context of benchmarking vision-based multi-object tracking [45] to the domain of sequential LiDAR semantic and instance segmentation. Together with the metric, we also propose an approach that operates directly on spatio-temporal point clouds providing object instances in space and time.

Point Cloud Segmentation. Semantic segmentation or point-wise classification of point clouds is a well-known research topic [2]. Traditionally, it was solved using feature extractors in combination with traditional classifiers [1] and conditional random fields to enforce label consistency of neighboring points [73, 50, 81]. Availability of large-scale datasets, such as S3DIS [3], Semantic3D [27], and recently SemanticKITTI [5], made it possible to also investigate end-to-end pipelines [39, 48, 72, 30, 85, 63, 60, 59, 47]. Similar to recent trends in RGB-D [33, 20] and LiDAR segmentation [78], our method performs bottom-up point grouping in a data-driven fashion. However, different from the aforementioned, we perform grouping in 3D space and time. We use the backbone by [72] that applies deformable point convolutions directly on the point clouds. In our case, this empirically performed better compared to backbones, specifically designed for point sequences [15, 63].

[1] https://github.com/mehmetaygun/4d-pls
[2] http://bit.ly/4d-panoptic-benchmark

Multi-Object Tracking and Segmentation. The majority of vision-based MOT methods follow tracking-by-detection [52]. Here the idea is to first run a pre-trained object detector independently in each video frame and then associate detections across time. In the past, there was a strong focus on developing robust and, preferably, globally optimal methods for data association [84, 42, 56, 11, 12]. Recent data-driven trends mainly focus on learning to associate detections [41, 67] or to regress targets [8], often in combination with end-to-end learning [16, 82, 10].

In the realm of robot vision, it is critical to localize object trajectories in 3D space and time. Early methods localized monocular detections in 3D *e.g.*, using stereo [43, 53, 21], or performed tracking in a category-agnostic manner by first performing bottom-up segmentation based on spatial proximity followed by point-segment association [71, 29]. Recently, LiDAR-based MOT became popular, thanks to the emergence of reliable 3D object detectors [65, 40] and LiDAR-centric datasets [13, 68]. Weng *et al*. [76] demonstrated that simple methods based on linear assignment and constant-velocity motion models can perform surprisingly well when 3D detections are localized reliably. Our method departs from 3D object detection in the spatial domain, followed by the detection association in the temporal domain. Instead, we follow recent advances in image [51, 14] and video instance segmentation [4]. We localize possible object instance centers within a 4D volume and associate points to estimated centers in a bottom-up manner, while a semantic branch assigns semantic classes to points.

3. Method

In this paper, we propose a method and a metric for 4D Panoptic LiDAR Segmentation task that tackles LiDAR semantic segmentation and instance segmentation jointly in the spatial and temporal domain. Given a sequence of LiDAR scans, the goal of this task is to predict for each 3D point (i) a semantic label for both *stuff* and *thing* classes, and (ii) a unique, identity-preserving object instance ID that should persist over the whole sequence.

3.1. 4D Panoptic LiDAR Segmentation: 4D-PLS

In this work, we take a different path compared to the tracking-by-detection paradigm to video-instance and video-panoptic segmentation [74, 83, 34, 32]. We pose 4D panoptic segmentation as two joint processes. The first one is responsible for point grouping in the 4D continuum using clustering, while the second assigns a semantic interpretation to each point.

We provide an overview of our method in Fig. 2. In a nutshell, we first form 4D point clouds from several consecutive LiDAR scans. In parallel, within a single network pass, we localize the most likely object centers (inspired by point-based tracking methods by [86, 14]) in the sequence

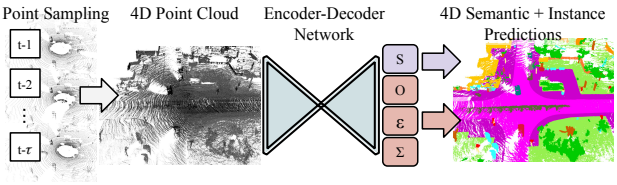


Figure 2: Visualization of our method. We sample points from past scans to form a 4D point cloud. Our encoder-decoder network estimates point objectness map (O), point variance map (Σ), and point embeddings (ε). We use these maps to assign points to their respective instances via density-based clustering in a 4D continuum. We obtain semantic interpretation from the semantic decoder (S).

(objectness map O), assign semantic classes to points (semantic map S), and compute per-point embeddings (embedding map ε) and variances (variance map Σ).

The clustering can be performed efficiently by evaluating the probability of each 4D point belonging to a certain "seed" point, which is similarly performed in the context of images and video segmentation [51, 4]. Finally, to associate 4D sub-volumes, we examine point intersections between overlapping point volumes.

4D Volume Formation. During inference and training, we form overlapping 4D point cloud volumes in an online setting. In particular, for scan t and temporal window size τ, we align together point clouds within temporal window $\{\max(0, t-\tau), ..., t\}$ using ego-motion estimates provided by a SLAM approach [7]. Our experiments in Sec. 4.1 reveal that processing multiple point clouds significantly improves spatial and temporal point association performance. However, due to the linear growth in memory requirements, stacking point clouds along the temporal dimension is prohibitively expensive. To overcome this issue, we build on the intuition that *thing* classes are most critical for a stable temporal association, since these classes correspond to potentially moving objects. As we operate in an online setting, where the past scans have already been processed, we can sample points that belong to *thing* classes or lie near to an object centers from earlier scans.

Density-based Clustering. We model object instances via Gaussian probability distributions. Given an estimate of the object center, *i.e.*, clustering "seed" point, we can assign points to their respective instance by evaluating each point under the Gaussian pdf based on the point's embedding vectors. The estimated centers do not need to correspond to exact object centers but are merely used to initiate the clustering. Thus, our approach is in practice fairly robust to occlusions and cross-time view changes. We note that the Gaussian assumption is only valid for shorter temporal windows. In particular, given a point $\mathbf{p}_i$ representing the instance center and its embedding vector e_i, and a query point p_j with its embedding vector e_j, we can evaluate the

probability of point p_j belonging to its center "seed" point p_i as:

$$\hat{p}_{ij} = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(e_i - e_j)^\top \Sigma_i^{-1}(e_i - e_j)\right) \quad (1)$$

where Σ_i is a diagonal matrix constructed using variance prediction σ_i of point p_i. We concatenate coordinate values (x, y, z, t) with the learned point embedding vectors to combine spatial and temporal coordinates with learned embeddings. We account for these additional dimensions during the training of the variance map.

Network and Training. To perform such clustering, we need to identify most likely instance centers, *i.e.*, "seed" points, in a 4D point cloud. We also need variance predictions for each point to evaluate probability scores during clustering, and a posterior over all semantic classes.

We estimate all these quantities using an encoder-decoder architecture that operates directly on the 4D point cloud $P \in \mathbb{R}^{N\times 4}$. The encoder network is based on the KPConv [72] backbone that uses deformable point convolutions. The decoder predicts point-wise feature embeddings $\varepsilon \in \mathbb{R}^{N\times D}$ using consecutive point convolutions. On top of the encoder, we add an object centerness decoder in $\mathbb{R}^{N\times 1}$, point variance decoder in $\mathbb{R}^{N\times D}$, and semantic decoder in $\mathbb{R}^{N\times C}$. We train our network in an end-to-end manner and in online fashion.

To train the semantic decoder, we use cross-entropy classification loss L_{class}. As the semantic classes are highly imbalanced, we sample points to ensure that the probability of sampling a point from a certain class is roughly uniform.

To learn the point centerness and point variance, we use three different losses. First, we impose the mean squared error (MSE) loss to train the object centerness decoder. However, there will generally be no points near the actual object centers, unlike the image and video domain [51, 4]. Therefore, instead of predicting per-point centerness, we predict the proximity of the point to its instance center. We compute for each point $\mathbf{p}_i$ its objectness o_i as Euclidean distance between the point and its instance center, *i.e.*, mean point of all instance points, normalized to $[0, 1]$. This objectness o_i is then compared to the regressed objectness $\hat{o}_i$:

$$L_{\text{obj}} = \sum_{i=1}^{N} (\hat{o}_i - o_i)^2, \quad \hat{o}_i, o_i \in [0, 1] \quad (2)$$

Since we want the embeddings of instances to form clusters in the spatio-temporal domain, we introduce our instance loss. Given a 4D point cloud of N points and K instances, it is defined as:

$$L_{\text{ins}} = \sum_{j=1}^{K} \sum_{i=1}^{N} (\hat{p}_{ij} - p_{ij})^2, p_{ij} = \begin{cases} 1, & \text{if } \mathbf{p}_i \in I_j \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $\hat{p}_{ij}$ is evaluated under the Gaussian pdf (Eq. 1) with points embedding e_i as well as instance embedding and variance e_j and σ_j. In addition, we employ variance smoothness loss L_{var}, similar to [4, 51] for training the variance decoder. In summary, we use four different losses to train our network in end-to-end manner: $L = L_{\text{class}} + L_{\text{obj}} + L_{\text{ins}} + L_{\text{var}}$.

Inference. We resolve point-to-instance associations in two stages, first within a processed 4D volume, and then across volumes. First, based on the point cloud centerness map, we select the point $\mathbf{p}_i$, which has the highest objectness score. Then, we evaluate probabilities p_{ij} for all candidate points and assign them to the cluster in case $p_{ij} > 0.5$. The assigned points are then removed from the candidate pool. We repeat these steps until the next highest objectness score is below a certain threshold. To transfer identities across processed 4D volumes, we perform cross-volume association greedily based on the overlap score, taking all scans into account. When the overlap is below a threshold, we assign a new id.

3.2. Measuring Performance

The central question when proposing a novel task and benchmark is how to evaluate and compare different methods. Preferably, we would like to summarize performance with a single number to rank the methods while retaining the capability of looking at different aspects of the task.

3.2.1 Existing Evaluation Measures

To motivate our approach to evaluation, we first briefly discuss established metrics for image-based panoptic segmentation (PQ [35]) and multi-object tracking and segmentation (MOTSA/MOTSP [9, 74]). Then, we discuss two recently proposed extensions of PQ to the temporal domain and argue why we do not promote their adaptation for the task of 4D LiDAR panoptic segmentation.

Segment-centric Evaluation. PQ and MOTSA/MOTSP are instance-centric evaluation metrics. Both first determine a unique matching between sets of ground-truth objects and model predictions for each frame individually to determine true positives (TPs), false positives (FPs), and false negatives (FNs). Both metrics provide measures for the segmentation and recognition aspects of the task. The segmentation quality (SQ) term of PQ and MOTSP integrates IoUs over the set TPs and normalizes it by the size of the TP set. The recognition quality (RQ) term of PQ is expressed as the F_1 score. Similarly, MOTSA combines detection errors (FNs and FPs) with ID switch (IDSW) penalty in a single term. IDSW occurs when a track is lost, and the tracker assigns a new identity to a tracked object. This is the only term that takes the temporal aspect of the task into account.

A criticism of PQ is that it over-emphasizes the importance of very small segments and *stuff* classes can be difficult to match [57]. MOTSA overemphasizes the detection compared to the association aspect and it is nonintuitive, since the score can be negative and is unbounded, as can be seen in Sec. 4. Furthermore, the influence of the ID switches on the final score depends on the frame rate, and MOTSA does not reward trackers that recover from incorrect associations. Importantly, both metrics are sensitive to the selection of the matching threshold. Thus, instances that slightly miss this threshold will cause both a FN and a FP. This is not the case for pixel or point centric metrics used for evaluating semantic segmentation. The standard mean IoU (mIoU) metric [22] computes sets of TPs, FPs and FNs pixel (or point) basis, effectively bypassing the segment matching.

PQ Extensions. Recent work [34] proposes video panoptic quality, a variant of PQ for the sequential domain. Different from PQ, gt-to-prediction mapping is established based on the sequential IoU matching criterion [83].As objects are not present throughout the clip and the difficulty of the task critically depends on the length of the temporal window, the final metric is averaged over varying temporal window sizes. This is suitable for the setting defined in [34], where the task is to evaluate short, sparsely labeled video snippets. However, this approach does not scale to real-world sequences of arbitrary length. Another extension to PQ, panoptic tracking quality (PTQ) [32] combines MOTA and PQ by adding an ID penalty to the PQ measure. This approach inherits issues from both PQ and MOTSA metrics.

3.2.2 LiDAR Segmentation and Tracking Quality

In the following, we assume a sequence of 3D point clouds of length l, sampled at discrete time-steps: $\Omega = \{(\mathbf{p}, n) \in \mathbb{R}^3 \times \mathbb{N} | n < l\}$. We define the ground-truth assignment function as $gt(\mathbf{p}, n) \rightarrow (c, id)$ and a prediction function as $pr(\mathbf{p}, n) \rightarrow (c, id)$, that map each 4D tuple, consisting of a point $\mathbf{p}$ and a timestamp n, to a certain class c and identity id. In the following, we devise an evaluation metric that, for each pair $(\mathbf{p}, n)$, evaluates whether (i) it was assigned to a correct class, and (ii) for the *thing* classes, whether it was assigned to the correct object instance. Inspired by the recently introduced Higher Order Tracking Accuracy (HOTA) [45], proposed in the context of MOT, and concurrent work on video panoptic segmentation proposing the Segmentation and Tracking Quality (STQ) [75], our *LSTQ* (LiDAR Segmentation and Tracking Quality) consists of two terms, the classification score S_{cls} and the association score S_{assoc}.

We adopt a fundamentally different evaluation philosophy compared to other metrics [45, 35, 34, 32]. In particular, we drop the concept of the frame-level "detection" and do not match segments between ground-truth and prediction. Our association score measures point-to-instance association quality in a unified way – in space *and* time at point level, which is more natural for segmentation tasks.

Classification Score. For the classification score, we first define instance-agnostic ground-truth and predictions sets:

$$gt_{\text{agn}}(c) = \{(\mathbf{p}, n) \mid \text{gt}(\mathbf{p}, n) = (c, *)\},$$
$$pr_{\text{agn}}(c) = \{(\mathbf{p}, n) \mid \text{pr}(\mathbf{p}, n) = (c, *)\},$$

representing the ground truth and predicted points that belong to class c irrespective of their assigned ids. Then, the TP, FP, FN sets are computed as in semantic segmentation evaluation with respect to gt class c and predicted class c':

$$\text{TP}_c = |pr_{\text{agn}}(c) \cap gt_{\text{agn}}(c)|,$$
$$\text{FP}_c = |pr_{\text{agn}}(c) - pr_{\text{agn}}(c) \cap gt_{\text{agn}}(c)|,$$
$$\text{FN}_c = |gt_{\text{agn}}(c) - pr_{\text{agn}}(c) \cap gt_{\text{agn}}(c)|.$$

The classification score then simply boils down to intersection-over-union (IoU) over these sets, which is the standard approach for evaluating semantic segmentation (however, this is different from segment-centric PQ, where points contribute to the TP_c term only if the segment that they belong is matched). We follow the standard procedure and average over the classes:

$$S_{cls} = \frac{1}{|\mathcal{C}|} \sum_{c=1}^{\mathcal{C}} \frac{|\text{TP}_c|}{|\text{TP}_c| + |\text{FN}_c| + |\text{FP}_c|} = \frac{1}{C} \sum_{c=1}^{\mathcal{C}} IoU(c).$$

Association Score. To evaluate the association score, we introduce the following class-agnostic predictions and ground-truth for the *thing* classes:

$$gt_{id}(id) = \{(\mathbf{p}, n) \mid gt(\mathbf{p}, n) = (c, id), c \in things\},$$
$$pr_{id}(id) = \{(\mathbf{p}, n) \mid pr(\mathbf{p}, n) = (c, id), c \in things\}.$$

We define the true positive association (*TPA*) set between a ground-truth object t with identity id and prediction s, that was assigned identity id'. This gives us a set of points with mutually consistent identities id and id':

$$TPA(id, id') = |pr_{id}(id') \cap gt_{id}(id)|. \quad (4)$$

Similarly, we define the set of false positive associations:

$$FPA(id, id') = |pr_{id}(id') - pr_{id}(id') \cap gt_{id}(id)|. \quad (5)$$

Intuitively, this set contains predicted point assignments with identity id', that were assigned a different ground-truth identity ($\neq id$), or were not assigned to a valid object instance. Finally, the set of false negative assignments:

$$FNA(id, id') = |gt_{id}(id) - pr_{id}(id') \cap gt_{id}(id)| \quad (6)$$

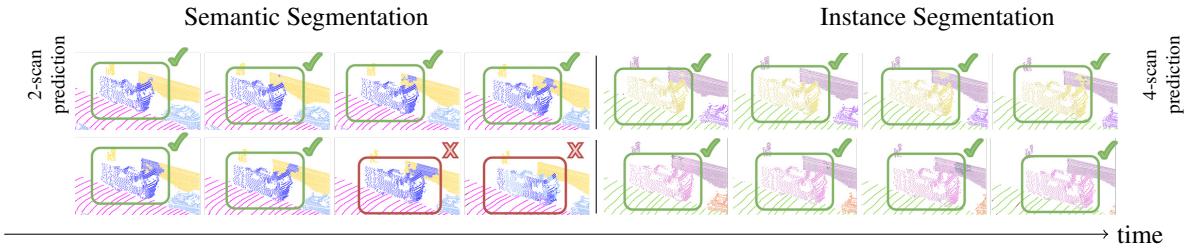


Figure 3: Predictions from 2 and 4 scan versions, MOTSA=1.0/0.0, S_{assoc}=0.96/0.93, S_{cls}=0.86/0.75. While both models track the instance correctly, due to slight difference in semantic segmentation predictions, MOTSA scores differ drastically.

contains ground-truth points with identity id that were assigned an identity, different to id', or were missed. We note that the concept of TPA, FPA and FNA was first introduced in the context of MOT evaluation for measuring the quality of temporal detection association. Therefore, to establish these sets, a bijective mapping between gt and pred needs to be established (as in the case of [9]). However, in *LSTQ*, these sets are established with respect to each 4D point, treating association in space and time in a unified manner.

Once we have quantified these sets, we can evaluate how well a predicted segment s *agrees* with ground-truth segment t. Because a ground truth segment t may be explained by multiple different predictions, we sum contributions of all pairs with non-zero overlap:

$$S_{assoc} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{1}{|gt_{id}(t)|} \sum_{\substack{s \in S \\ s \cap t \neq 0}} TPA(s,t) IoU(s,t), \quad (7)$$

where the IoU term is evaluated using the TPA, FNA and FPA sets (Eq. 4, 6, 5). In practice, we do not need to perform any point segment association, and even a prediction with a single common point will contribute to this term. We normalize these contributions by the tube volume, and we weigh each contribution by the volume of the TPA set. This weighting term ensures that instances with larger temporal spans have a higher contribution to the final score. Finally, our metric is computed as a geometric mean of the two terms: $LSTQ = \sqrt{S_{cls} \times S_{assoc}}$. The advantage over the arithmetic mean is that the final score will become zero if any of the two terms approach zero. This reflects our intuition that failing at either of two aspects of the task should yield a very low final score.

LSTQ tolerates different semantic predictions within a spatio-temporal segment by design (the IoU term in Eq. 7 is evaluated in a class-agnostic manner). Following STQ [75], we *decouple semantic and association errors*, otherwise, *e.g.*, a truck mistaken for a bus would be harshly penalized by the association term, even though it was tracked correctly. This behavior that dis-entangles association and classification errors is different from MOTSA/PTQ/VPQ where semantics and temporal association are entangled.

4. Experimental Evaluation

In this section, we first evaluate different strategies for forming 4D point cloud volumes, assess the impact of processing multiple scans on the final performance, and discuss several possibilities for the embeddings used for point grouping. We compare our method to baselines for single-scan LiDAR panoptic segmentation [6] and 4D panoptic LiDAR segmentation by extending existing methods.

We use the SemanticKITTI [5] LiDAR dataset to conduct our experiments. It contains 22 sequences from KITTI odometry dataset [25] and provides point-wise semantic and temporally consistent instance annotations [6]. We use the training/validation/test split by SemanticKITTI [5, 6].

4.1. Ablation studies

We perform all ablations on the validation set and interpret results through the lens of *LSTQ* metric (Sec. 3.2.2).

Point Propagation. As discussed in Sec. 3, we cannot simply stack point clouds temporarily due to the memory constraints. We build on the intuition that we can subsample a set of points from the past scans that are most beneficial for the end-task performance. As we are operating in an online setting, and the past scans have been already processed, we can leverage past predictions.

In this experiment, we discuss different temporal point sampling strategies for varying temporal window sizes of $\tau = 2, 4, 6$. In the *thing-propagation* strategy, we exclusively sample points which are assigned only to a thing class, as they represent only a small number of all point. In the *importance sampling* strategy, we sample 10% of points with a probability proportional to the objectness. This way, we focus on points likely to represent *thing* classes, while still allowing to propagate points belonging to *stuff* classes, which can aid the semantic segmentation of the task. Similarly, the *temporal decay* sampling uses the objectness score as a deciding factor, but we decay the number of sampled points based on the proximity to the current scan. Finally, the *strided sampling* samples points with a stride of 2 along the temporal dimension.

As can be seen in Tab. 1, the *importance sampling* strategy yields higher performance compared to sampling only

Strategy	# fr.	*LSTQ*	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}	Mem.
Base	1	51.92	45.16	59.69	64.60	60.40	1x
Thing-prop.	2	59.20	58.71	*59.69*	*64.04*	*61.16*	1.05x
	4	*60.94*	*63.95*	57.84	63.95	56.67	1.15x
	6	58.88	61.13	56.71	63.86	53.97	1.25x
Importance samp.	2	59.86	58.79	*60.95*	64.96	*63.06*	1.1x
	4	**62.74**	**65.11**	60.46	*65.36*	61.26	1.3x
	6	61.52	64.28	58.88	65.32	57.38	1.5x
Temporal decay	2	59.86	58.79	*60.95*	64.96	*63.06*	1.1x
	4	*62.14*	*64.05*	60.30	65.33	60.91	1.3x
	6	61.03	63.34	58.81	*65.38*	57.11	1.5x
Temporal stride	2	*62.29*	*63.52*	**61.08**	65.10	**63.18**	1.1x
	4	61.63	62.89	60.39	**65.45**	60.39	1.3x
	6	59.33	59.95	58.72	65.39	56.88	1.5x

Table 1: Ablation study on point sampling strategies for building 4D point cloud volumes with respect to different temporal window sizes.

Mixing	# Sc.	*LSTQ*	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}
xyz	2	51.65	43.77	60.95	64.96	63.06
$xyzt$	2	51.95	44.30	60.93	64.80	63.15
Emb.	2	48.58	43.63	54.10	59.54	50.41
Emb. + xyz	2	54.15	56.63	55.11	61.14	53.71
Emb. + $xyzt$	2	59.86	58.79	60.95	64.96	63.06
xyz	4	54.29	48.77	60.44	65.29	61.32
$xyzt$	4	54.46	49.55	59.87	64.80	61.15
Emb.	4	56.77	60.63	53.17	58.00	52.25
Emb. + xyz	4	58.43	63.90	54.68	61.12	52.67
Emb. + $xyzt$	4	62.74	65.11	60.46	65.36	61.26

Table 2: Embedding design ablation.

thing classes, at a slightly increased memory cost. As expected, this approach improves association quality and aids semantics as it also propagates points representing *stuff*.

Interestingly, even a temporal window of size 2 drastically improves the performance compared to a single scan baseline, at negligible memory consumption ($1.1\times$). We observe the largest gains when the scans are temporally close: our 4-scan multi-scan baseline improves performance $51.92 \rightarrow 62.74$ in terms of *LSTQ*. The association term benefits more from processing multiple scans compared to the segmentation term. This confirms that our model learns to exploits temporal cues very well. While temporal decaying does aid semantic or temporal aspect, introducing a temporal stride of 2 yields the highest performance gains for semantic point classification. However, denser sampling in the temporal domain benefits association. Hence, we focus on the importance sampling strategy with $\tau = 4$. In the supplementary, we highlight the performance with temporal window size $\tau = 1, 2, 3, 4, 6, 8$. As can be seen, the association accuracy is increasing up to $\tau = 4$ and then saturates, while classification accuracy saturates a $\tau = 2$; however, it only decreases marginally.

Embedding Design. In this experiment, we study different point embeddings for clustering and show our findings in Tab. 2. We investigate the base performance of using only 3D spatial (xyz) and 4D spatio-temporal point coordinates $(xyzt)$, and using only learned embeddings (Emb.).

Method	PQ	$PQ^{\dagger}$	SQ	RQ	mIoU
RangeNet++ [48] + PointPillars [40]	37.1	45.9	75.9	47.0	52.4
KPConv [72] + PointPillars [40]	44.5	52.5	80.0	54.4	58.8
Panoptic RangeNet [47]	38.0	47.0	76.5	48.2	50.9
Our method (single scan)	**50.3**	**57.8**	**81.6**	**61.0**	**61.3**

Table 3: Single scan Panoptic Segmentation (test set).

Method	*LSTQ*	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}
RangeNet++[48] + PP + MOT	35.52	24.06	52.43	64.52	35.82
KPConv [72] + PP + MOT	38.01	25.86	55.86	66.90	47.66
RangeNet++[48] + PP + SFP	34.91	23.25	52.43	64.52	35.82
KPConv [72] + PP + SFP	38.53	26.58	55.86	66.90	47.66
Our (single scan) + MOT	40.18	28.07	57.51	66.95	51.50
Our (single scan) + SFP	43.88	33.48	57.51	66.95	51.50
Ours (multi scan)	56.89	56.36	57.43	66.86	51.64

Table 4: 4D Panoptic (test set). MOT – *tracking-by-detection* by [76]; SFP – *tracking-by-detection* via scene flow based propagation [49]; PP – PointPillars [40].

Next, we investigate the performance of the coordinate mixing formulation that combines learned embeddings with 3D spatial and 4D spatio-temporal coordinates. As can be seen, the variant in which we combine both yields the best results, not only in terms of S_{assoc}, but also S_{cls}. This shows that a well-designed embedding branch has a positive effect on learning the backbone features. Note that for the baseline that uses only spatio-temporal coordinates, we still use our fully trained network.

4.2. Benchmark Results

Single-scan Prediction. First, we evaluate our method using single-scan LiDAR panoptic segmentation [5, 6] to demonstrate the effectiveness of our network solely in the spatial domain. We use points from single scans during training and testing. We follow the standard evaluation protocol and compare to published and peer-reviewed methods.

As can be seen from Tab. 3, our method achieves state-of-the-art results on all metrics for semantic and panoptic segmentation [35, 5, 6]. The first two entries use two different networks for object detection and semantic segmentation, followed by fusion of the results. We use a single network to obtain both semantic and instance segmentation of the point cloud in a single network pass. We note that the recently proposed Panoptic RangeNet [47] and RangeNet++ [48] combined with PointPillars [40] detector operate on the range image and not the point cloud, and therefore, use a different backbone. However, KPConv with PointPillars uses the same backbone as our method.

4D Panoptic Segmentation. For evaluation in the multi-scan setting for the 4D panoptic segmentation task, we extend all single-scan methods reported in Tab. 3, except for the Panoptic RangeNet [47]. We adapt them to the sequential domain using two strategies. AB3DMOT [76] uses a constant-velocity motion model to obtain track predictions associated with object detection based on a 3D bounding

Method	*LSTQ*	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}	sPTQ	PTQ	sMOTSA	MOTSA
RangeNet++[48] + PP + MOT	43.76	36.28	52.78	60.49	42.17	34.58	33.83	-7.88	-4.57
KPConv [72] + PP + MOT	46.27	37.58	56.97	64.21	54.13	39.13	38.11	-6.16	-2.41
RangeNet++[48] + PP + SFP	43.38	35.66	52.78	60.49	42.17	35.83	35.46	-3.13	-0.01
KPConv [72] + PP + SFP	45.95	37.07	56.97	64.21	54.13	41.44	41.05	2.83	6.1
MOPT [32]	24.80	11.73	52.41	62.37	45.27	41.82	42.39	12.88	17.07
Our (single scan) + MOT	51.92	45.16	59.69	64.60	60.40	48.36	47.84	6.65	12.69
Our (single scan) + SFP	45.45	34.61	59.69	64.60	60.40	48.24	47.72	3.01	7.93
Ours (2 scans)	59.86	58.79	**60.95**	64.96	**63.06**	51.14	50.67	**29.04**	**33.2**
Ours (4 scans)	**62.74**	**65.11**	60.46	**65.36**	61.26	**51.50**	**51.20**	0.34	4.8

Table 5: 4D Panoptic (validation set). MOT – *tracking-by-detection* method by [76]; SFP – *tracking-by-detection* via scene flow based segment propagation [49]; PP – PointPillars [40] detector.

Category	# Instances	% Instances	# Scans	TP	FP	FN	IDS	Precision	Recall	MOTSA	S_{assoc}	S_{cls}
Motorcycle	255	0.01	2	209	151	46	31	0.58	0.82	0.11	0.56	0.88
			4	231	747	24	9	0.24	0.91	-2.06	0.81	0.74
Other-vehicle	2138	0.06	2	778	362	1360	162	0.68	0.36	0.12	0.17	0.56
			4	1022	1131	1116	99	0.47	0.48	-0.10	0.38	0.55

Table 6: Per-class evaluation on SemanticKITTI validation set (2 and 4 scan versions).

box overlap. The second strategy, Scene Flow Propagation (SFP) is inspired by standard baselines that perform optical flow warping of points, followed by mask-IoU based association. This approach is commonly used in vision-based video object segmentation [46], video instance segmentation [83], and multi-object tracking and segmentation [54, 55, 74]. Instead of optical flow, we use state-of-the-art LiDAR scene flow [49]. We outline our results, obtained on the test set in Tab. 4. As can be seen, the baseline that uses KPConv [72] to obtain per-pixel classification, PointPillars (PP) detector [40] and a network for point cloud propagation (SFP [49]) performs slightly better in terms of association accuracy than the standard 3D MOT baseline. Our method that unifies all three aspects in a single network outperforms all tracking-by-detection baselines by a large margin, including our single-scan baseline.This confirms the importance of tackling all three aspects of these tasks in a unified manner. An important contribution of our paper is the finding that even when processing smaller overlapping sub-sequences with our network (and resolving intra-window associations with a simple overlap-based approach), we perform significantly better compared to single-scan baselines that use more elaborate association techniques (*e.g.*, Kalman filter), as can be confirmed in Tab. 4.

Metric Insights. In this section, we analyze the performance on the validation split (Tab. 5) through the lens of several evaluation metrics and analyze per-class performance (Tab. 6). Our method outperforms all baselines with respect to all metrics. However, while our 4-scan variant performs better than the 2-scan variant in terms of *LSTQ*, we observe a significant drop in the MOTSA score. Our analysis shows that this is caused by negative MOTSA scores on some classes due to a decrease in precision while having fewer ID switches (see Tab. 6, and supplementary).We visualize such case in Fig. 3. As can be seen, the difference is due to the semantic interpretation of the points and not due to the segmentation and tracking quality at the instance level. This confirms the nonintuitive behavior of MOTSA, while our metric provides insights on both semantic interpretation and instance segmentation and tracking. For a more details, we refer to the supplementary.

5. Conclusion

In this paper, we extended LiDAR panoptic segmentation to the temporal domain resulting in the 4D Panoptic Segmentation task. We presented an evaluation metric suitable for analyzing this task's performance and proposed a new model. Importantly, we have shown that a single model tackling semantic segmentation and point-to-instance association jointly in space and time substantially outperforms methods that independently tackle these aspects. We hope that our unified view and model, accompanied by a public benchmark, will pave the road to future developments.

Acknowledgements. This project funded by the Humboldt Foundation through the Sofja Kovalevskaja Award, the EU Horizon 2020 research and innovation programme under grant agreement No.101017008 (Harmony) and the German Federal Ministry of Education and Research (BMBF) under grant No.01IS18036B. The authors of this work take full responsibility for its content. We thank authors of [32] for providing results for their approach, Ismail Elezi and the whole DVL group for helpful discussions.

References

[1] Anuraag Agrawal, Atsushi Nakazawa, and Haruo Takemura. MMM-classification of 3D Range Data. In *ICRA*, 2009. 2

[2] Dragomir Anguelov, Ben Taskar, Vassil Chatalbashev, Daphne Koller, Dinkar Gupta, Geremy Heitz, and Andrew Ng. Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data. In *CVPR*, 2005. 2

[3] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *CVPR*, 2016. 2

[4] Ali Athar, Sabarinath Mahadevan, Aljoša Ošep, Laura Leal-Taixé, and Bastian Leibe. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *ECCV*, 2020. 3, 4

[5] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *ICCV*, 2019. 1, 2, 6, 7

[6] Jens Behley, Andres Milioto, and Cyrill Stachniss. A benchmark for LiDAR-based panoptic segmentation based on KITTI. In *ICRA*, 2021. 2, 6, 7

[7] Jens Behley and Cyrill Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *RSS*, 2018. 3

[8] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *ICCV*, 2019. 1, 3

[9] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *JIVP*, 2008:1:1–1:10, 2008. 2, 4, 6

[10] Guillem Braso and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *CVPR*, June 2020. 1, 3

[11] William Brendel, Mohamed R. Amer, and Sinisa Todorovic. Multi object tracking as maximum weight independent set. *CVPR*, 2011. 3

[12] Asad A. Butt and Robert T. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *CVPR*, June 2013. 3

[13] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 1, 2, 3

[14] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020. 3

[15] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 2

[16] Peng Chu and Haibin Ling. FAMNet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking. In *ICCV*, 2019. 3

[17] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1

[18] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 1

[19] Patrick Dendorfer, Aljoša Ošep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, and Stefan Roth Laura Leal-Taixé. Motchallenge: A benchmark for single-camera multiple target tracking. *IJCV*, pages 1–37, 2020. 1

[20] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3D-MPA: Multi Proposal Aggregation for 3D Semantic Instance Segmentation. In *CVPR*, 2020. 2

[21] Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc Van Gool. Robust multiperson tracking from a mobile platform. *PAMI*, 31(10):1831–1846, 2009. 3

[22] Mark Everingham, Luc van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010. 1, 5

[23] Pedro Felzenszwalb, David Mcallester, and Deva Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. In *CVPR*, 2008. 1

[24] Davi Frossard and Raquel Urtasun. End-to-end learning of multi-sensor 3d tracking by detection. *ICRA*, 2018. 2

[25] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 1, 2, 6

[26] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 1

[27] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D. Wegner, Konrad Schindler, and Marc Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS*, 2017. 2

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1

[29] David Held, Jesse Levinson, Sebastian Thrun, and Silvio Savarese. Combining 3d shape, color, and motion for robust anytime tracking. In *RSS*, 2014. 3

[30] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020. 2

[31] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *CVPR*, 2018. 2

[32] Juana Valeria Hurtado, Rohit Mohan, and Abhinav Valada. MOPT: Multi-object panoptic tracking. In *CVPR Workshops*, 2020. 2, 3, 5, 8

[33] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *CVPR*, 2020. 2

[34] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. In *CVPR*, 2020. 1, 2, 3, 5

[35] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019. 1, 2, 4, 5, 7

[36] Artem Komarichev, Zichun Zhong, and Jing Hua. A-cnn: Annularly convolutional neural networks on point clouds. In *CVPR*, 2019. 2

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[38] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S Davis. Modeling local geometric structure of 3d point clouds using geo-cnn. In *CVPR*, 2019. 2

[39] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 2

[40] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1, 2, 3, 7, 8

[41] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese cnn for robust target association. *CVPR Workshops*, 2016. 3

[42] Laura Leal-Taixé, Gerard Pons-Moll, and Bodo Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In *ICCV Workshops*, 2011. 3

[43] Bastian Leibe, Konrad Schindler, Nico Cornelis, and Luc Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *PAMI*, 30(10):1683–1698, 2008. 1, 3

[44] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1

[45] Jonathon Luiten, Aljoša Ošep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *IJCV*, 2020. 2, 5

[46] Jonathon Luiten, Paul Voigtlaender, and Bastian Leibe. Premvos: Proposal-generation, refinement and merging for video object segmentation. In *ACCV*, 2018. 8

[47] Andres Milioto, Jens Behley, Chris McCool, and Cyrill Stachniss. Lidar panoptic segmentation for autonomous driving. In *IROS*, 2020. 2, 7

[48] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *IROS*, 2019. 1, 2, 7, 8

[49] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *CVPR*, 2020. 7, 8

[50] Daniel Munoz, J. Andrew Bagnell, Nicolas Vandapel, and Martial Hebert. Contextual Classification with Functional Max-Margin Markov Networks. In *CVPR*, 2009. 2

[51] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *CVPR*, 2019. 3, 4

[52] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, 2004. 3

[53] Aljoša Ošep, Wolfgang Mehner, Markus Mathias, and Bastian Leibe. Combined image- and world-space tracking in traffic scenes. In *ICRA*, 2017. 1, 3

[54] Aljoša Ošep, Wolfgang Mehner, Paul Voigtlaender, and Bastian Leibe. Track, then decide: Category-agnostic vision-based multi-object tracking. *ICRA*, 2018. 8

[55] Aljoša Ošep, Paul Voigtlaender, Mark Weber, Jonathon Luiten, and Bastian Leibe. 4d generic video object proposals. *ICRA*, 2020. 8

[56] Hamed Pirsiavash, Deva Ramanan, and Charles C.Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011. 3

[57] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *CVPR*, 2019. 2, 5

[58] Johannes Pöschmann, Tim Pfeifer, and Peter Protzel. Factor graph based 3d multi-object tracking in point clouds. *arXiv preprint arXiv:2008.05309*, 2020. 1

[59] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 2

[60] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 1, 2

[61] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao. P2b: Point-to-box network for 3d object tracking in point clouds. In *CVPR*, 2020. 1

[62] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1

[63] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *CVPR*, 2020. 2

[64] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020. 1

[65] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *CVPR*, 2019. 1, 3

[66] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network. *PAMI*, 2020. 1

[67] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-object tracking with quadruplet convolutional neural networks. In *CVPR*, 2017. 3

[68] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 1, 2, 3

[69] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching Efficient 3D Ar-

chitectures with Sparse Point-Voxel Convolution. In *ECCV*, 2020. 1

[70] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *CVPR*, 2018. 2

[71] Alex Teichman and Sebastian Thrun. Tracking-based semi-supervised learning. *IJRR*, 31(7):804–818, 2012. 3

[72] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 1, 2, 4, 7, 8

[73] Rudolph Triebel, Krisitian Kersting, and Wolfram Burgard. Robust 3D Scan Point Classification using Associative Markov Networks. In *ICRA*, 2006. 2

[74] Paul Voigtlaender, Michael Krause, Aljoša Ošep, Jonathon Luiten, B.B.G Sekar, Andreas Geiger, and Bastian Leibe. MOTS: Multi-object tracking and segmentation. In *CVPR*, 2019. 1, 2, 3, 4, 8

[75] Mark Weber, Jun Xie, Maxwell Collins, Yukun Zhu, Paul Voigtlaender, Hartwig Adam, Bradley Green, Andreas Geiger, Bastian Leibe, Daniel Cremers, Aljoša Ošep, Laura Leal-Taixé, and Liang-Chieh Chen. STEP: Segmenting and Tracking Every Pixel. *arXiv preprint arXiv:2102.11859*, 2021. 2, 5, 6

[76] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. *IROS*, 2020. 1, 2, 3, 7, 8

[77] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris Kitani. GNN3DMOT: Graph neural network for 3d multi-object tracking with multi-feature learning. In *CVPR*, 2020. 2

[78] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. Identifying unknown instances for autonomous driving. In *CoRL*, 2020. 2

[79] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *ICRA*, 2018. 2

[80] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, 2019. 2

[81] Xuehan Xiong, Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. 3-D Scene Analysis via Sequenced Predictions over Points and Regions. In *ICRA*, 2011. 2

[82] Yihong Xu, Aljoša Ošep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, and Xavier Alameda-Pineda. How to train your deep multi-object tracker. In *CVPR*, 2020. 3

[83] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *ICCV*, 2019. 1, 2, 3, 5, 8

[84] Li Zhang, Li Yuan, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, 2008. 3

[85] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *CVPR*, 2020. 1, 2

[86] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *ECCV*, 2020. 3

[I] J. Behley, A. Milioto, and C. Stachniss, "A Benchmark for LiDAR-based Panoptic Segmentation based on KITTI," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021. DOI: 10.1109/ICRA48506.2021.9561476

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

A Benchmark for LiDAR-based Panoptic Segmentation based on KITTI

Jens Behley Andres Milioto Cyrill Stachniss

***Abstract*— Panoptic segmentation is the recently introduced task that tackles semantic segmentation and instance segmentation jointly [18]. In this paper, we present an extension of SemanticKITTI [1], a large-scale dataset providing dense point-wise semantic labels for all sequences of the KITTI Odometry Benchmark [10]. This extension enables training and evaluation of LiDAR-based panoptic segmentation. We provide the data and discuss the processing steps needed to enrich a given semantic annotation with temporally consistent instance information, i.e., instance information that supplements the semantic labels and identifies the same instance over sequences of LiDAR point clouds. Additionally, we present two strong baselines that combine state-of-the-art LiDAR-based semantic segmentation approaches with a state-of-the-art detector enriching the segmentation with instance information and that allow other researchers to compare their approaches against. We believe that our extension of SemanticKITTI with strong baselines enables the creation of novel algorithms for LiDAR-based panoptic segmentation as much as it has for the original semantic segmentation and semantic scene completion tasks. Data, code, and an online evaluation service using a hidden test set are publicly available at http://semantic-kitti.org.**

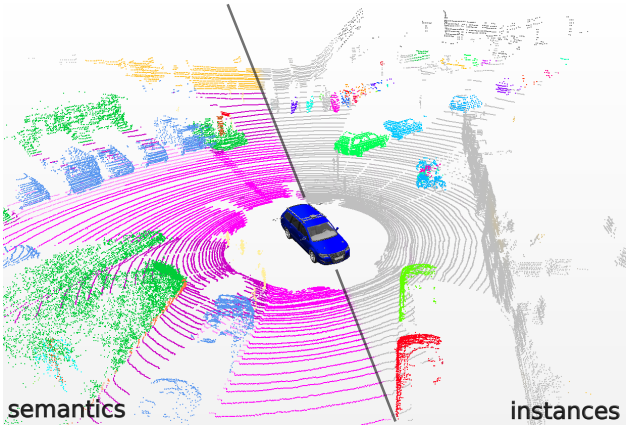


Fig. 1: Using the semantic segmentation (left part) and the point-accurate instance annotations for traffic participants (right part), we provide a benchmark for panoptic segmentation [18] using three-dimensional LiDAR point clouds. Our work extends the SemanticKITTI [1] dataset, which is based on the KITTI Vision Benchmark [10].

I. Introduction

Fine-grained scene understanding is a pre-requisite for truly autonomous systems, such as self-driving cars. This encompasses the type of surfaces, but also identifying individual objects. The former is often designated as *stuff* and the latter as *things* [18]. Only both sources of information together enable autonomous systems to reason about the drivability of surfaces, the type of objects and obstacles, and possibly the intent of other agents in the vicinity.

Assigning to each individual pixel or point a semantic label is called semantic segmentation, while the identification and separation of individual objects is called instance segmentation. These tasks are usually solved in isolation, but an increasing number of methods have been recently developed that solve both jointly using either images [18], [32], [41], [39], [24], [7] or RGB-D data [30], [14], [16], [9], [43], [40], [26], [13]. These developments were mainly driven by the availability of a metric [18] and the swift adaption of the task in different popular semantic segmentation datasets, such as Cityscapes [8], Microsoft's Common Objects in Context (COCO) [20], and Mapillary Vistas [27]. While semantic segmentation will still be relevant in the future, we expect that instance segmentation will be soon replaced and subsumed by the panoptic segmentation task, as it is a part of a panoptic segmentation framework.

In this paper, we present an extension of the SemanticKITTI dataset [1] providing the necessary annotations to evaluate panoptic segmentation on automotive LiDAR scans. Fig. 1 shows an example of the provided instance annotation for all traffic participants, i.e., vehicles, pedestrians, and cyclists. To ease the generation of instance information with provided semantic segmentation of the LiDAR point clouds, we first generate for static and non-static objects instance information using grid-based clustering [4] and distance-based clustering approach. Unfortunately, such a clustering often leads to over- or under-segmentation, which we had to manually correct using our point labeling tool. Furthermore, we provide two baseline approaches that combine state-of-the-art semantic segmentation with state-of-the-art object detection methods.

In summary, our contributions are as follows:

- We provide temporally-consistent instance annotations for all traffic participants including vehicles, pedestrians, bicyclists, and motorcyclists for the KITTI Odometry Benchmark.
- We provide two strong baseline approaches combining current state-of-the-art semantic segmentation and a state-of-the-art 3D object detector.
- We provide a benchmark with a publicly available online evaluation service for approaches solving LiDAR-based panoptic segmentation using a hidden test set.

All authors are with the University of Bonn, Germany.

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant number BE 5996/1-1 and under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob.

TABLE I: Overview of other LiDAR datasets with annotations for instances (top) and semantic segmentation (bottom).

	Name	#Scans[1]	#Boxes/#Points	#Classes[2]	Data[3]	FoV[4]	Sequential	Reference
Instance Annotation	KITTI (Detection)	7k/7k	1k	3(3)	B	F	✗	[10]
	Argoverse	22k	993k	17	B	C	✓	[6]
	Lyft	46k	1.3M	9	B	C	✓	[17]
	CADC	7k	305k	10	B	C	✓	[31]
	Waymo	200k	12M	4	B	C	✓	[37]
	A2D2	12k	12k	14	B	F	✗	[11]
	H3D	27k	1.1M	8	B	F	✗	[29]
	PandaSet	16k	1.4M	12	B	C	✓	[36]
	nuScenes	44k	1.4M	10 (23)	B	C	✓	[5]
	SemanticKITTI	23k/20k	682k	8	P	C	✓	
Semantic Segmentation	Oakland3d	17	1.6M	5 (44)	P	C	✗	[25]
	Freiburg	77	1.1M	4 (11)	P	C	✗	[3]
	Wachtberg	5	400k	5 (5)	P	C	✗	[3]
	Semantic3d	15/15	4009M	8 (8)	P	C	✗	[12]
	Paris-Lille-3D	3	143M	9 (50)	P	C	✗	[35]
	Zhang et al.	140/112	32M	10 (10)	P	F	✗	[42]
	SemanticPOSS	2k	216M	14	P	C	✗	[28]
	A2D2	31k	930k	38	P†	F	✗	[11]
	PandaSet	16k	1388M	42	P	C	✓	[36]
	nuScenes	40k	1400M	32	P	C	(✓)	[5]
	SemanticKITTI	23k/20k	4549M	25 (28)	P	C	✓	[1]

[1] Number of scans for train and test set, [2] Number of classes used for evaluation and number of classes annotated in brackets, [3] type of annotations, where B and P correspond to bounding boxes (B) and point-wise (P), [4] field-of-view (FoV) of LiDAR sensor with annotations, where F denotes frontal and C denotes complete 360°. † point-wise annotations via projection to annotated image and using corresponding image label.

II. Related Work

Shortly after Kirillov *et al*. [18] proposed panoptic segmentation and a metric to measure the performance of approaches providing such labels, the established datasets for semantic segmentation of *image data*, i.e., Cityscapes [8], Microsoft's Common Objects in Context (COCO) [20], and Mapillary's Vistas [27] adopted the metric and added an evaluation for this task.

Due to the availability of the data, we witnessed a wide adoption and interest for panoptic segmentation in the computer vision community [7], [18], [21], [30], [32], [41], [39], [24]. While there have also been approaches for RGB-D data [14], [30], [9], [16], [43], [40], [13], there were basically no approaches available that operate on LiDAR data, when we released the data in April 2020. There was simply no annotated data available that provided both point-wise semantic labels *and* instance information. Recently, first approaches adopted the provided annotations to develop approaches for LiDAR-based panoptic segmentation [22], [15].

Recently, almost all major self-driving car companies release datasets providing besides camera also LiDAR data [37], [17], [11], [6], [5]. While most datasets provide also annotations for object instances by bounding boxes, only a few datasets provide point-wise semantic annotation [1], [11], [36], [5] needed to evaluate panoptic segmentation for LiDAR. Tab. I summarizes the amount of data provided by the different datasets.

SemanticKITTI [1] is a dataset based on the KITTI Vision Benchmark [10], which might not show the diversity of different inner cities traffic and weather conditions, but still provides unparalleled long sequences showing a variety of different environments and driving situations. Our annotations with point-wise labels for the full 360° field-of-view provide labels for 28 classes including labels distinguishing moving and non-moving objects. By providing now instance annotations together with an online evaluation on a hidden test set, we close the gap to the aforementioned established image-based dataset and provide a benchmark for panoptic segmentation using an automotive LiDAR. We hope that the availability of labeled LiDAR scans for panoptic segmentation opens the door for more research in the direction of LiDAR-based panoptic segmentation.

III. Dataset

In this section, we introduce the provided dataset and discuss the annotation process to extract instance information from a given semantic segmentation in a semi-automatic fashion with acceptable manual labeling effort to adjust for wrong over- and under-segmentations.

Fig. 2 shows a qualitative example of the annotation provided by our dataset. The left part of the figure shows the semantic segmentation of our SematicKITTI dataset, which we use to determine the instance annotation. The right side depicts the temporally consistent annotations, where different colors correspond to individual instances. Note, that same colored instances in the top and the bottom row of this figure correspond to the same instance ID.

A. Annotation Process

For annotation of the instances, we employ a semi-automatic process using different strategies to generate a

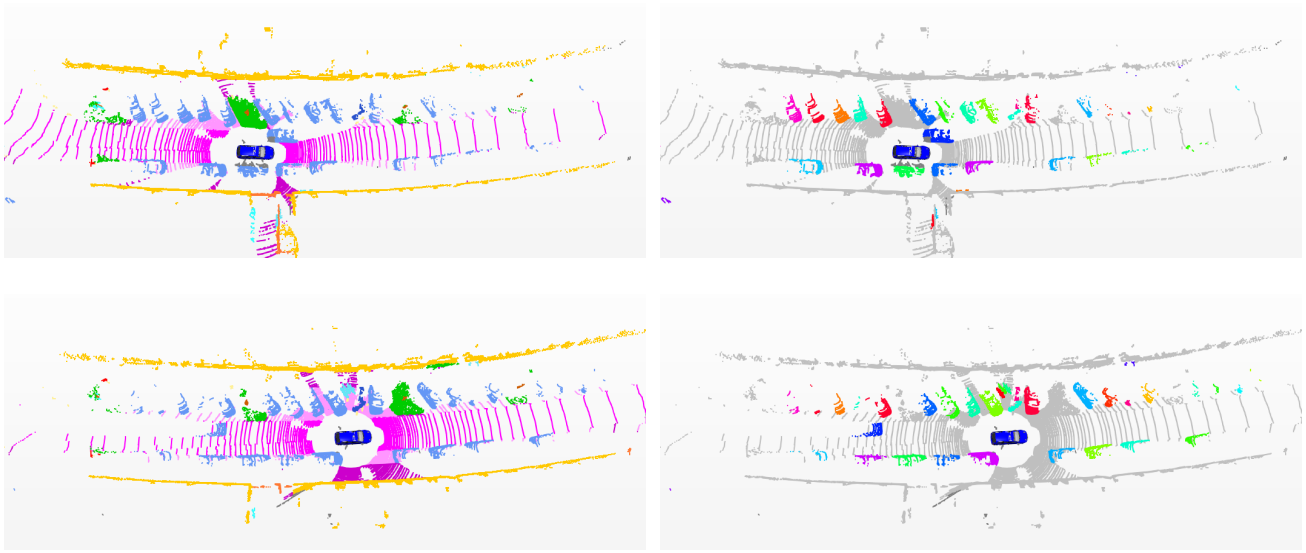


Fig. 2: Qualitative example of the instance annotation over a sequence of scans: on the left is the semantic annotation and on the right is the instance annotation shown. Top and bottom rows show consecutive timestamps from sequence 13. Note, same colors at different timestamps correspond to the same instance id. Best viewed in color.

temporally consistent instance annotation. Our goal is to label the same instance through the whole sequence with the same instance ID – even for instances that move. For static objects, the data association can be simply performed by considering the location of the segment after performing a pose correction using a Simultaneous Localization and Mapping (SLAM) system [2]. For moving objects, we have to account for the motion of the object as well as the motion of the sensor at the same time.

Overall, the SemanticKITTI dataset [1] provides 28 classes (including 6 classes to distinguish moving from non-moving classes) from which we select the traffic participants as *thing* classes for the panoptic segmentation, i.e., car, truck, other-vehicle, motorcycle, bicycle, person, bicyclist, and motorcyclist. The remaining classes are *stuff* classes for the panoptic segmentation, i.e., road, sidewalk, parking, other-ground, building, vegetation, trunk, terrain, fence, pole, and traffic-sign.

For *static thing classes*, we first cluster all points for each individual class using a fast grid-based segmentation approach [4] to handle the large number of points efficiently. We then split the aggregated point cloud into tiles of size $100\,\text{m}$ by $100\,\text{m}$ using the pose information by our SLAM system [2]. For each tile, we use a two-dimensional grid with cell size $0.1\,\text{m}$ by $0.1\,\text{m}$, which allows us often to separate even close parking cars. Next, all points are inserted into the corresponding grid cells using their x and y-coordinates. Finally, only grid cells with points exceeding a height threshold $\Delta > 0.5\,\text{m}$ are combined using a flood fill algorithm to combine neighboring grid cells into segments.

For *moving thing classes*, we generate clusters for each scan individually using a distance-based clustering as this provided more reliable results and could be also used to associate instances between consecutive scans. First, we search for each point in a radius of $0.5\,\text{m}$ for the nearest neighbor and cluster points together that share neighbors. To find associations with the previous 4 scans, we use a slightly larger radius of $1.0\,\text{m}$ to find neighbors between two different timestamps. If we find enough neighbors with the previous segments at different timestamps, we associate them together and assign the same instance ID.

The described clustering leads inevitably to over- and under-segmentation (cf. Fig. 3), but also to wrong or missing associations between consecutive timestamps. We correct these issues manually using an own point labeling tool, which provides tools to create, join, and split instances. Overall, the manual correction for all 22 sequences took roughly $70\,\text{h}$ of additional labor.

B. Statistics

Fig. 4 provides an overview of the number of instances and the actual number of bounding boxes per class. We show in the upper part of the figure the sequence-wise counts of instance annotations, i.e., we count each object only once, even if it is seen multiple times by the sensor. The lower part of the figure shows the accumulated scan-wise counts of instances, where we count the instances without considering the temporally consistent instance ID.

The bulk of the instances correspond to cars, which are naturally occurring in city-like environments and also correspond to the normal statistics in autonomous driving scenarios. Usually, an autonomous car will also encounter some classes far fewer than other classes or situations. They are usually denoted as the 'long tail' problem, referring to the underrepresented entities in a given distribution. This adds complexity to the task, since panoptic segmentation approaches, which are designed to tackle this scenario must be able to deal with such skewed class distributions.

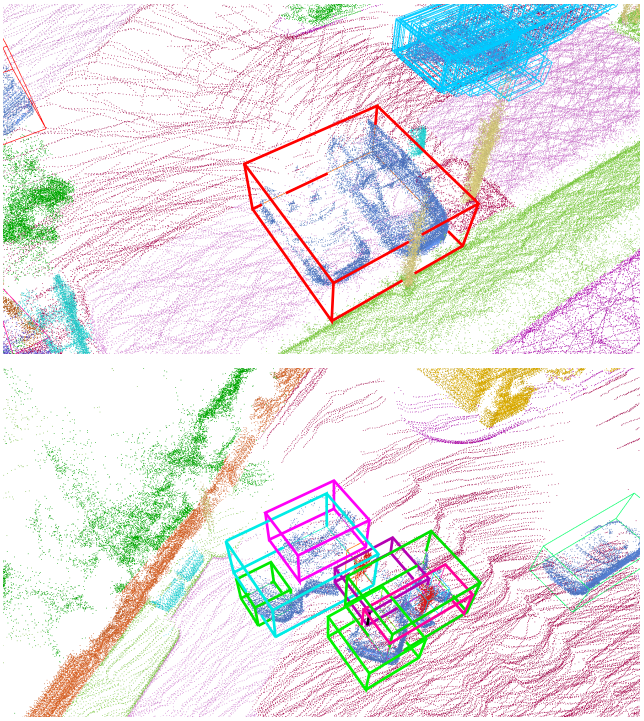


Fig. 3: Example of under- (top) and over-segmentation (bottom) generated by our semi-automated clustering approach, which we manually corrected by splitting or joining segments.

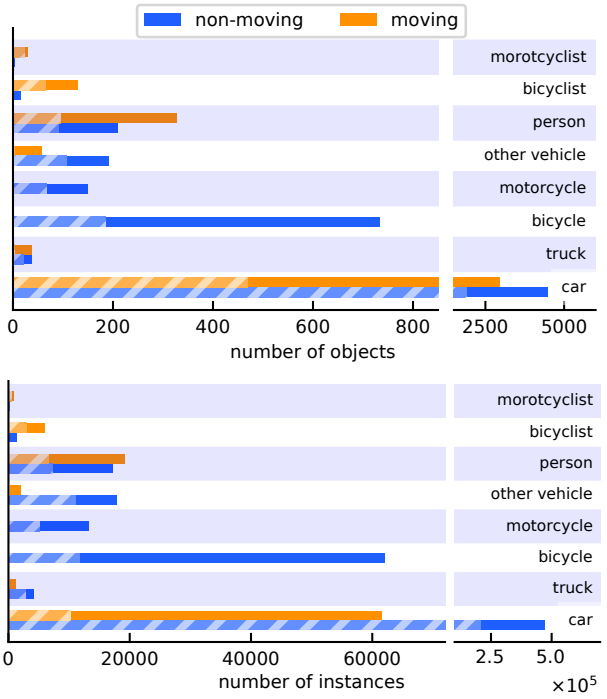


Fig. 4: Top: number of (sequence-wise) objects. Bottom: number of (scan-wise) instances. The hashed bars correspond to the training data. The large number of scan-wise annotations in relation to the number of objects indicates that many objects are seen over an extended period of time.

IV. Baseline Approaches

None of the available RGB-D approaches [14], [30], [13], [9], [16], [43], [40] could be easily adapted to the size of the data and the specific characteristics of the LiDAR point clouds. Available approaches for RGB-D panoptic segmentation either use a truncated signed distance function (TSDF) [14], [26] or voxel grids [13], or employ PointNets [33], [34] for feature extraction [30], [9], [16], [43], [40]. A TDSF representation and PointNets perform purely on single point clouds due to the characteristic sparsity pattern of the rotating LiDAR sensor, which generates dense point clouds at close ranges and sparse point clouds at larger ranges. Furthermore, our evaluation for the SemanticKITTI datatset [1] of variants of the PointNets [33], [34] showed inferior performance with such point clouds.

Thus, we propose two baseline approaches combining current state-of-the-art semantic segmentation approaches on SemanticKITTI, namely KPConv [38] and RangeNet++ [23], paired with a state-of-the-art object detector on the KITTI 3D object detection benchmark, namely PointPillars [19], providing instance-level information.

To this end, we use the oriented bounding boxes of the object detector, i.e., bounding boxes for *cars*, *pedestrians*, and *cyclists* trained on the KITTI detections benchmark [10], to determine the instance ID for points inside the bounding boxes. By combining the predictions of the semantic segmentation and assigning the instance ID of each bounding box to each point inside of it, we obtain a panoptic segmentation. Note that we only assign instance IDs to points from the *thing* classes, i.e., points under a *car* classified as *road* or *parking* are not assigned an instance ID.

For the baseline, we used pre-trained models or publicly available predictions for KPConv [38] and RangeNet++ [23], which were trained on SemanticKITTI. PointPillars had to be trained from scratch using the provided implementation[1], modifying the configuration of the object detector such that it provides region proposals and bounding boxes for the full 360-degree field-of-view of the LiDAR sensor.

These networks were run independently for semantic segmentation and object detection and then merged to generate a panoptic segmentation. None of the approaches can, therefore, run at the frame rate of the LiDAR, i.e., 10 Hz, and thus having computational budgets that are not suitable in an autonomous car. Furthermore, the PointPillars detector [19] requires training separate networks, one for the class *car* and one for *pedestrian* combined with *cyclist*, which accentuates the problem further. We provide an evaluation of the performance of these approaches including runtime information in the experimental section of this paper.

Note that the decision for using an object detector providing oriented bounding boxes was made to minimize the negative effect of axis-aligned bounding boxes, which would lead to large overlaps between cars parking near to each other, see also Fig. 5 for an example. Thus, oriented bounding boxes lead to more accurate instance annotations in the depicted case.

[1]See the GitHub repository at https://git.io/Je25l.

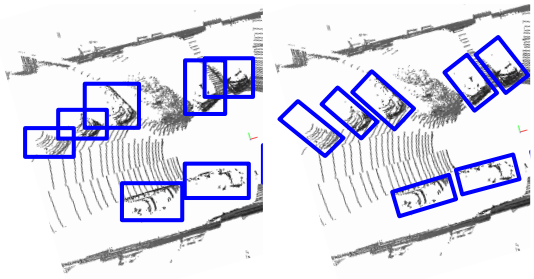


Fig. 5: Overlapping of axis-aligned bounding boxes and therefore wrong or ambiguous assignment of points inside bounding boxes (left). With oriented bounding boxes this ambiguity due to overlapping bounding boxes does not occur (right).

V. Experiments

Before we discuss details of the baseline implementations and the results of our baseline approaches, we shortly provide a summary of the panoptic segmentation metric.

A. Evaluation Metric

In panoptic segmentation, each point $\mathbf{p}_i$ not only carries a class label $y_i \in \mathcal{Y}$, where $|\mathcal{Y}|$ is the number of classes, but also can have an instance ID n_i, where $n_i = 0$ denotes no specific instance.

To measure the quality of this joint assignment, we briefly recapitulate the recently proposed panoptic quality (PQ) metric [18]. Let $\mathcal{S}, \hat{\mathcal{S}}$ denote segments, i.e., sets of points in our specific case, sharing an class and instance ID. Here, we assume that the *stuff* classes simply get instance ID $n_i = 0$ corresponding to no specific instance assigned.

Furthermore, let $\text{IoU}(\mathcal{S}, \hat{\mathcal{S}}) = (\mathcal{S} \cap \hat{\mathcal{S}}) \cdot (\mathcal{S} \cup \hat{\mathcal{S}})^{-1}$ denote the intersection-over-union of these two sets. Let the set of true positive matches TP_c be the pairs of predicted segments $\hat{\mathcal{S}}$ that overlap at least with 0.5 IoU with a ground truth segment $\mathcal{S}$, $\text{TP}_c = \{(\mathcal{S}, \hat{\mathcal{S}}) \mid \text{IoU}(\mathcal{S}, \hat{\mathcal{S}}) > 0.5\}$. Likewise, let FP_c the set of unmatched predicted segments $\hat{\mathcal{S}}$ and FN_c the set of unmatched ground truth segments $\mathcal{S}$.

With the above definitions, the class-wise PQ_c is given by

$$\text{PQ}_c = \frac{\sum_{(\mathcal{S},\hat{\mathcal{S}}) \in \text{TP}_c} \text{IoU}(\mathcal{S}, \hat{\mathcal{S}})}{|\text{TP}_c| + \frac{1}{2}|\text{FP}_c| + \frac{1}{2}|\text{FN}_c|}. \tag{1}$$

The panoptic quality metric is computed for each class independently and averaged over all classes, which makes the metric insensitive to class imbalance [18], i.e.,

$$\text{PQ} = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} \text{PQ}_c. \tag{2}$$

Kirillov *et al.* [18] furthermore define the segmentation quality (SQ) as average IoU over matched segments and the recognition quality (RQ) corresponding to the F_1 score.

Porzi *et al.* [32] proposed to alter the metric to account for *stuff* classes having only a single segment since no pixels (or, in our case, points) have an instance ID. Hence, the IoU-based criterion could often lead to an unmatched prediction. To account for *stuff* classes, Porzi *et al.* use

$$\text{PQ}_c^{\dagger} = \begin{cases} \text{IoU}(\mathcal{S}, \hat{\mathcal{S}}) & \text{, if c is a } \textit{stuff} \text{ class} \\ \text{PQ}_c & \text{, otherwise.} \end{cases} \tag{3}$$

Consequently, we denote by $\text{PQ}^{\dagger}$ the average over the class-wise modified $\text{PQ}_c^{\dagger}$ as defined in (2).

Furthermore, the quality of the semantic segmentation is also measured using the mean intersection-over-union (mIoU), which also enables the comparison with other approaches in the semantic segmentation benchmark. This metric is defined as follows:

$$\text{mIoU} = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} \frac{|\{i \mid y_i = c\} \cap \{j \mid \hat{y}_j = c\}|}{|\{i \mid y_i = c\} \cup \{j \mid \hat{y}_j = c\}|}, \tag{4}$$

where y_i corresponds to the ground truth label of point $\mathbf{p}_i$ and $\hat{y}_i$ to the prediction.

B. Baseline Parameters, Training, and Inference Details

In this section, we provide more details on the training and inference of the two-stage baselines. We, furthermore, provide details on the modifications needed to use the models on the SemanticKITTI [1] benchmark, which requires to use full point clouds of a single turn for training and inference. We will provide code for merging the predictions to enable the reproduction of our results.

KPConv by Thomas *et al.* [38]. For scene classification, Thomas *et al.* [38] extract 10 overlapping spheres of 10 m radius, subsample the point clouds with a voxel grid (resolution of 0.1 m), and drop random points in case there are more than $15,000$ points left. To aggregate predictions, they perform majority vote on the overlapping parts of the predictions. Overall, this achieves state-of-the-art single scan performance of 58.5 mIoU and performs better than taking a subsampled single point cloud with mIoU 56.6 (subsampling with voxel grid of resolution 0.1 m).

RangeNet++ by Milioto *et al.* [23]. Here, we directly use the predictions available in our repository[2], which are also provided for the test set. RangeNet++ uses a range image of size 2048×64 for training and inferences, which is then upsampled to the complete point cloud by using nearest neighbors. To remove artifacts from the reprojection, it applies a k-nearest neighbor filtering, which accounts for k neighbors in a certain range.

PointPillars by Lang *et al.* [19]. We used for training of the approach the aforementioned implementation supported by the Point Pillar authors. Since SemanticKITTI does not offer oriented bounding boxes, we use the 3D object detection part of KITTI Object Detection [10] for training. The KITTI dataset was recorded with the same sensor and a similar environment, but there is no overlap between the point cloud sequences of the odometry and the detection benchmark.

For training on the KITTI object detection, we follow the original approach of Lang *et al.* [19] and use 0.16 m as voxel grid resolution with a maximum of 12.000 pillars with at most 100 points for each pillar for training on the KITTI Object Detection subset of the KITTI Vision Benchmark [10]. As commonly done and also advocated by Lang *et al.* [19], we trained a network for *cars*, car network,

[2] https://github.com/PRBonn/lidar-bonnetal

Method	mIoU	PQ	PQ†	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}
KPConv [38] + PointPillars [19]	58.8	44.5	52.5	54.4	80.0	32.7	38.7	81.5	53.1	65.9	79.0
RangeNet++ [23] + PointPillars [19]	52.4	37.1	45.9	47.0	75.9	20.2	25.2	75.2	49.3	62.8	76.5

TABLE II: Comparison of test set results on SemanticKITTI using *stuff*(St) and *thing*(Th) *classes*. All results in [%].

Method	PQ	road	sidewalk	parking	other ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign
KPConv [38] + PointPillars [19]	44.5	84.6	60.1	34.1	8.8	80.7	72.5	17.2	9.2	30.8	19.6	77.6	53.9	42.2	29.9	59.4	22.8	49.0	46.2	46.8
RangeNet++ [23] + PointPillars [19]	37.1	90.6	63.2	41.3	6.7	79.2	66.9	6.7	3.1	16.2	8.8	71.2	34.6	37.4	14.6	31.8	13.5	38.2	32.8	47.4

TABLE III: Detailed class-wise results of test set results on SemanticKITTI in panoptic quality (PQ) [18]. All results in [%].

Method	PQ†	road	sidewalk	parking	other ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign
KPConv [38] + PointPillars [19]	52.5	88.8	72.7	61.3	31.6	90.5	72.5	17.2	9.2	30.8	19.6	84.8	69.2	69.1	29.9	59.4	22.8	64.2	56.4	47.4
RangeNet++ [23] + PointPillars [19]	45.9	91.8	75.1	65.0	27.7	87.4	66.9	6.7	3.1	16.2	8.8	80.5	55.1	64.8	14.6	31.8	13.5	58.6	47.9	55.9

TABLE IV: Detailed class-wise results of test set results on SemanticKITTI in fixed panoptic quality ($PQ^{\dagger}$) [32]. All results in [%].

and a separate network for *pedestrian* and *cyclist*, called pedcyclist network.

For the car network, we consider the part in front of the sensor inside the ranges $x = (0.0, 69.12)$, $y = (-39.68, 39.68)$, and $z = (-3.0, 1.0)$, where we assume that the sensor is located at $(0, 0, 0)$. For the pedcyclist network, we use $x = (0.0, 48.0)$, $y = (-20.0, 20.0)$, and $z = (-2.5, 0.5)$ as volume of the point pillar grid.

For prediction on the SemanticKITTI dataset, we are interested in predicting bounding boxes for the full field-of-view of the sensor. Thus, we adapted the parameters for inference. For the car network, we use a grid volume of size $x = (-69.12, 69.12)$, $y = (-69.12, 69.12)$, $z = (-3.0, 1.0)$. For the pedcyclist network, we use a similar grid volume of $x = (-69.12, 69.12)$, $y = (-69.12, 69.12)$, $z = (-2.5, 0.5)$. Furthermore, we increase the number of maximal pillars to 30000 and adopt the anchor generation strides to accommodate the large input volume.

Considering the runtime of the proposed two-stage approach, we observe a large discrepancy between the reported runtime and our obtained runtime, which cannot be only explained by using a different system (Nvidia Geforce RTX 2080 Ti vs. a Nvidia Geforce 1080 Ti). First, we have to note that the implementations might be different from the originally used implementation. We believe that the main reason seems to be the 3.4 times increase input volume and the increased number of pillars. The fact that the KITTI object detection benchmark only uses a part of the point cloud is also acknowledged in Sec. 6 of Lang *et al*. [19]. We furthermore do not use TensorRT for inference, which could additionally improve the runtime.

C. Baseline Results

Tab. II summarizes the results in a breakdown according to mean Intersection-over-Union (mIoU) and the different panoptic quality metrics. Due to the overall stronger performance on semantic segmentation of KPConv (58.8 mIoU vs. 52.4 mIoU in Tab. II), the panoptic baseline using KPConv is stronger in all metrics. We believe that this discrepancy can be directly attributed to the stronger performance on small classes. Tab. III and Tab. IV show the detailed results for all classes using the panoptic quality and the fixed panoptic quality, respectively.

For the runtime, we assume that the separate object detectors can be run in parallel (314 ms for *pedestrian/cyclist* and 105 ms for *car*) after the semantic segmentation (200 ms for KPConv and 95 ms for RangeNet++) resulting in 514 ms and 409 ms respectively.

VI. Conclusion

We present an extension of the SemanticKITTI dataset that enables the community to evaluate and benchmark panoptic segmentation approaches using data generated by an automotive LiDAR. We provide the data, code, as well as, an online platform for evaluation using a hidden test set. Additionally, we provide two panoptic segmentation baselines that are built from a combination of state-of-the-art semantic segmentation approaches and a 3D object detector. The goal of this dataset paper is to propel the research on LiDAR-based panoptic segmentation, since it is an important task that will become more relevant, and provide a platform for easy benchmarking of such approaches.

Acknowledgment

We thank Hugues Thomas for allowing us to use the predictions from his original KPConv approach for our work.

REFERENCES

[1] 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 Intl. Conf. on Computer Vision (ICCV)*, 2019.

[2] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[3] J. Behley, V. Steinhage, and A. Cremers. Performance of Histogram Descriptors for the Classification of 3D Laser Range Data in Urban Environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012.

[4] J. Behley, V. Steinhage, and A. Cremers. Laser-based Segment Classification Using a Mixture of Bag-of-Words. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[5] H. Caesar, V. Bankiti, A. Lang, S. Vora, V. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] M. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. Argoverse: 3D Tracking and Forecasting with Rich Maps. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] B. Cheng, M.D. Collins, Y. Zhu, T. Liu, T.S. Huang, H. Adam, and L.C. Chen. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] M. Cordts, S.M. Omran, Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] L. Du, J. Tan, X. Xue, L. Chen, H. Wen, J. Feng, J. Li, and X. Zhang. 3DCFS: Fast and Robust Joint 3D Semantic-Instance Segmentation via Coupled Feature Selection. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[10] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[11] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A.S. Chung, L. Hauswald, V.H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth. A2D2: Audi Autonomous Driving Dataset. *arXiv preprint*, 2020.

[12] T. Hackel, N. Savinov, L. Ladicky, J.D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98, 2017.

[13] L. Han, T. Zheng, L. Xu, and L. Fang. OccuSeg: Occupancy-aware 3D Instance Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] J. Hou, A. Dai, and M. Niessner. 3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[15] J. Hurtado, R. Mohan, W. Burgard, and A. Valada. MOPT: Multi-Object Panoptic Tracking. *arXiv preprint:2004.08189*, 2020.

[16] H. Jiang, F. Yan, J. Cai, J. Zheng, and J. Xiao. End-to-end 3D Point Cloud Instance Segmentation without Detection. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[17] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft Level 5 AV Dataset 2019. https://level5.lyft.com/dataset/, 2019.

[18] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] A. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick. Microsoft COCO: Common Objects in Context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 740–755, 2014.

[21] H. Liu, C. Peng, C. Yu, J. Wang, X. Liu, G. Yu, and W. Jiang. An End-To-End Network for Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[22] A. Milioto, J. Behley, C. McCool, and C. Stachniss. LiDAR Panoptic Segmentation for Autonomous Driving. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[23] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[24] R. Mohan and A. Valada. EfficientPS: Efficient Panoptic Segmentation. *arXiv preprint*, 2004.02307v2, 2020.

[25] D. Munoz, J.A. Bagnell, N. Vandapel, and M. Hebert. Contextual Classification with Functional Max-Margin Markov Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[26] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji. PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[27] G. Neuhold, T. Ollmann, S.R. Bulo, and P. Kontschieder. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[28] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, and H. Zhao. SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances. *arXiv preprint:2002.09147*, 2020.

[29] A. Patil, S. Malla, H. Gang, and Y.T. Chen. The H3D dataset forfull-surround 3D multi-object detection and tracking in crowded urban scenes. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[30] Q. Pham, D. Nguyen, B. Hua, G. Roig, and S. Yeung. JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds With Multi-Task Pointwise Networks and Multi-Value Conditional Random Fields. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[31] M. Pitropov, G. Danson, J. Rebello, M. Smart, C. Wang, K. Czarnecki, and S. Waslander. Canadian Adverse Driving Conditions Dataset. *arXiv preprint:2001.10117*, 2020.

[32] L. Porzi, S.R. Bulo, A. Colovic, and P. Kontschieder. Seamless Scene Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[33] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[34] C. Qi, K. Yi, H. Su, and L.J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2017.

[35] X. Roynard, J. Deschaud, and F. Goulette. Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *Intl. Journal of Robotics Research (IJRR)*, 37(6):545–557, 2018.

[36] scale and Hesai. PandaSet Dataset (Available at https://scale.com/open-datasets/pandaset), 2020.

[37] 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. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[38] H. Thomas, C. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[39] H. Wang, R. Luo, M. Maire, and G. Shakhnarovich. Pixel Consensus Voting for Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[40] X. Wang, S. Liu, X. Shen, C. Shen, and J. Jia. Associatively Segmenting Instances and Semantics in Point Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[41] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun. UPSNet: A Unified Panoptic Segmentation Network. In *Proc. of*

the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), 2019.

[42] J. Zhang and S. Singh. Visual-Lidar Odometry and Mapping: Low-Drift, Robust, and Fast. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[43] L. Zhao and W. Tao. JSNet: Joint Instance and Semantic Segmentation of 3D Point Clouds. In *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, 2020.

[J] X. Chen, I. Vizzo, T. Läbe, *et al.*, "Range Image-based LiDAR Localization for Autonomous Vehicles," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021. DOI: 10.1109/ICRA48506.2021.9561335

Range Image-based LiDAR Localization for Autonomous Vehicles

Xieyuanli Chen Ignacio Vizzo Thomas Läbe Jens Behley Cyrill Stachniss

***Abstract*— Robust and accurate, map-based localization is crucial for autonomous mobile systems. In this paper, we exploit range images generated from 3D LiDAR scans to address the problem of localizing mobile robots or autonomous cars in a map of a large-scale outdoor environment represented by a triangular mesh. We use the Poisson surface reconstruction to generate the mesh-based map representation. Based on the range images generated from the current LiDAR scan and the synthetic rendered views from the mesh-based map, we propose a new observation model and integrate it into a Monte Carlo localization framework, which achieves better localization performance and generalizes well to different environments. We test the proposed localization approach on multiple datasets collected in different environments with different LiDAR scanners. The experimental results show that our method can reliably and accurately localize a mobile system in different environments and operate online at the LiDAR sensor frame rate to track the vehicle pose.**

I. Introduction

Precise localization is a fundamental capability required by most autonomous mobile systems. With a localization system, a mobile robot or an autonomous car is capable to estimate its pose in a map based on observations obtained with onboard sensors. Precise and reliable LiDAR-based global localization is needed for autonomous driving, especially in GPS-denied environments or situations where GPS cannot provide accurate localization results.

Most autonomous mobile systems have a 3D LiDAR sensor onboard to perceive the environment and directly provide 3D range measurements. In this paper, we tackle the problem of vehicle localization based on such 3D LiDAR sensors. For localization, probabilistic state estimation techniques are used in most localization systems today. In particular, particle filters are a versatile tool as they do not need to restrict the motion or observation model to follow a specific distribution, such as a Gaussian. When utilizing particle filters, we need to design an appropriate observation model in lieu with a map representation. Frequently used observation models for LiDARs are the beam-end point model, also called the likelihood field [25], the ray-casting model [8], or models based on handcrafted features [23], [33]. These methods either only work efficiently with 2D LiDAR scanners or need carefully designed features to work properly. Recently, researchers also focused on data-driven learning of such observation models [12], [29], [6], which provide accurate and reliable results as long as the environment is close to the environment used for learning the model.

All authors are with the University of Bonn, Germany.

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob, and by the Chinese Scholarship Committee.

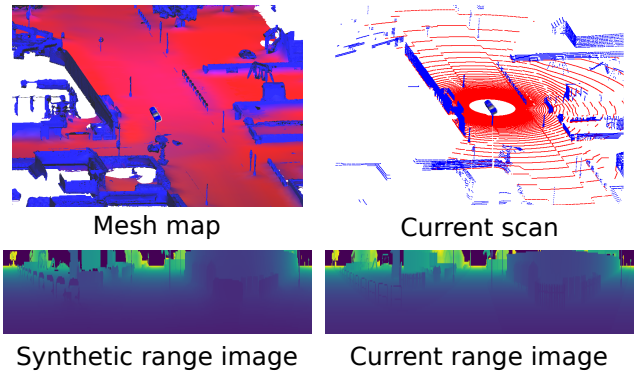


Fig. 1: Visualization of range images and triangular mesh map. On the left, we show the triangular mesh used as the map and the rendered synthetic range image from the mesh. In the mesh map, red parts correspond to ground planes and blue parts represent non-ground structures. On the right, we show the LiDAR point cloud at the same location and the corresponding range image generated from the LiDAR scan.

Instead of using raw point clouds obtained from a 3D LiDAR sensor or features generated or learned from the point clouds, we investigate range images for 3D LiDAR-based localization for autonomous vehicles. As shown in Fig. 1, we project the point clouds into range images and localize the autonomous system with rendered views from a map that is represented with a triangular mesh. There are several reasons to use range image representation and maps represented by a mesh. The cylindrical range image is a natural and light-weight representation of the scan from a rotating 3D LiDAR, and a mesh map is more compact than a large point cloud. Those properties enable our approach to achieve global localization in large-scale environments. Furthermore, the rendering of range images from a mesh map can be performed efficiently using computer graphics techniques. Therefore, range images and mesh maps are a perfect match for achieving LiDAR-based global localization.

The main contribution of this paper is a novel observation model for 3D LiDAR-based localization. Our model is based on range images generated from both, the real LiDAR scans and synthetic renderings of the *mesh* map. We use the difference between them to formulate the observation model for a Monte Carlo localization (MCL) for updating the importance weights of the particles. Based on our novel observation model, our approach provides (x, y, θ)-pose estimates for the vehicle and achieves global localization using 3D LiDAR

scans. Furthermore, our approach generalizes well over different environments collected with different types of LiDAR scanners. In sum, we make three key claims: Our approach is able to (i) achieve global localization accurately and reliably using 3D LiDAR data, (ii) can be used for different types of LiDAR sensors, and, (iii) generalize well over different environments. These claims are backed up by the paper and our experimental evaluation. The source code of our approach is available at: https://github.com/PRBonn/range-mcl.

II. Related Work

For localization given a map, one often distinguishes between pose tracking and global localization. In pose tracking, the vehicle starts from a known pose and the pose is updated over time. In global localization, no pose prior is available. In this work, we address global localization using 3D LiDAR data without assuming any pose prior from GPS or other sensors. Therefore, we concentrate here mainly on LiDAR-based approaches.

In the context of autonomous cars, many approaches were proposed for accurate pose tracking using multiple sensor modalities and high-definition (HD) maps. Levinson et al. [16] utilize GPS, IMU, and LiDAR scans to build HD maps for localization. They generate a 2D surface image of ground reflectivity in the infrared spectrum and define an observation model that uses these intensities. The uncertainty in intensity values has been handled by building a prior map [31]. Barsan et al. [12] use a fully convolutional neural network (CNN) together with HD maps to perform online-to-map matching for improving the robustness to dynamic objects and eliminating the need for LiDAR intensity calibration. Merfels and Stachniss [18] present an efficient chain-like pose graph for vehicle localization exploiting graph optimization techniques and different sensing modalities. Based on this work, Wilbers et al. [30] propose a LiDAR-based localization system performing a combination of local data association between laser scans and HD map features, temporal data association smoothing, and a map matching approach for robustification. The approaches above show good performance for tracking vehicles' poses but require GPS information for operating on HD maps. In contrast, our approach addresses global localization using only 3D LiDAR data without assuming any pose prior.

To achieve global localization, traditional approaches rely on probabilistic state estimation techniques [25]. A popular framework is Monte Carlo localization [8], [26], [10], which uses a particle filter to estimate the robot's pose and is widely used in robot localization systems [3], [6], [15], [24], [32].

Recently, several approaches exploiting deep neural networks and semantic information for 3D LiDAR localization have been proposed. For example, Ma et al. [17] combine semantic information such as lanes and traffic signs in a Bayesian filtering framework to achieve accurate and robust localization within sparse HD maps. Yan et al. [32] exploit buildings and intersections information from a LiDAR-based semantic segmentation system [19] to localize in OpenStreetMap data. Schaefer et al. [21] detect and extract pole landmarks from 3D LiDAR scans for long-term urban vehicle localization whereas Tinchev et al. [27] propose a learning-based method to match segments of trees and localize in both urban and natural environments. Sun et al. [24] use a deep-probabilistic model to accelerate the initialization of the Monte Carlo localization and achieve a fast localization in outdoor environments. In our previous work [6], [5], we also exploit CNNs with semantics to predict the overlap between LiDAR scans as well as their yaw angle offset, and use this information to build a learning-based observation model for Monte Carlo localization. The learning-based methods perform well in the trained environments, while they usually cannot generalize well in different environments or different LiDAR sensors.

In sum, our method only uses LiDAR data to achieve global localization outdoors without using any GPS. Moreover, our approach uses range information directly without exploiting neural networks, semantics, or extracting landmarks. Therefore, it generalizes well to different environments and different LiDAR sensors and does not require new training data when moving to different environments.

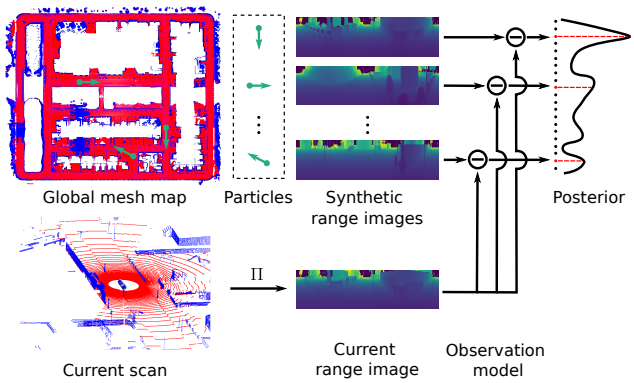


Fig. 2: Overview of our approach. We project the LiDAR point cloud into a range image and compare it to synthetic range images rendered at each particle location from a mesh map. Based on the range images, we propose a new observation model for localization and integrate it into a Monte Carlo localization system to estimate the pose posterior of the vehicle.

III. Our Approach

In this paper, we propose a probabilistic global localization system for autonomous vehicles using a 3D LiDAR sensor, see Fig. 2 for an illustration. To this end, we project the LiDAR point cloud into a range image (see Sec. III-A) and compare it to synthetic range images rendered at each particle location from a map represented by a triangular mesh (see Sec. III-B and Sec. III-C). Based on the range images, we propose a new observation model for LiDAR-based localization (see Sec. III-E) and integrate it into a Monte Carlo localization system (see Sec. III-D). Furthermore, we employ a tile map to accelerate the rendering and decide when the system converges (see Sec. III-F). Using an OpenGL-based rendering pipeline, the proposed system operates online at the frame rate of the LiDAR sensor after convergence.

A. Range Image Generation

The key idea of the proposed method is to use range images generated from LiDAR scans and rendered from the triangular mesh map for robot localization. To generate range images, we use a spherical projection [2], [19], [7], [5]. We project the point cloud $\mathcal{P}$ to the so-called vertex map $\mathcal{V}_D : \mathbb{R}^2 \mapsto \mathbb{R}^3$, where each pixel contains the nearest 3D point. Each point $\mathbf{p}_i = (x, y, z)$ is converted via the function $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates and finally to image coordinates (u, v), i.e.,

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x) \cdot \pi^{-1}\right] \cdot w \\ \left[1 - \left(\arcsin(z \cdot r^{-1}) + \mathrm{f}_{\mathrm{up}}\right) \mathrm{f}^{-1}\right] \cdot h \end{pmatrix}, \quad (1)$$

where $r = ||\mathbf{p}||_2$ is the range, $\mathrm{f} = \mathrm{f}_{\mathrm{up}} + \mathrm{f}_{\mathrm{down}}$ is the vertical field-of-view of the sensor, and w, h are the width and height of the resulting vertex map $\mathcal{V}_D$. Given the vertex map $\mathcal{V}_D$ and range r of points at each coordinate (u, v), we generate the corresponding range image $\mathcal{R}_D$, on which the subsequent computations are based upon.

B. Mesh-based Map Representation

We use a triangular mesh as a map $\mathcal{M}$ of the environment. A triangular mesh provides us with a compact representation that enables us to render the aforementioned range images at the frame rate of the LiDAR sensor.

To generate the map, we use LiDAR scans together with their poses provided a SLAM system uses for mapping. Note that it is not necessary for our approach to use the same LiDAR sensor for map generation and localization. We employ Poisson surface reconstruction (PSR) [13] to obtain the representation of the map as a triangular mesh from point clouds. The PSR algorithm requires an oriented point cloud, thus normals for all the points in the input cloud. To estimate the surface normals, we use a range image-based normal estimation [2], which computes normals in the so-called normal map $\mathcal{N}_D$ for each coordinate (u, v) using cross products over forward differences of the corresponding vertex map pixel, i.e.,

$$\begin{aligned} \mathcal{N}_D((u, v)) = &\left(\mathcal{V}_D((u+1, v)) - \mathcal{V}_D((u, v))\right) \\ &\times \left(\mathcal{V}_D((u, v+1)) - \mathcal{V}_D((u, v))\right). \end{aligned} \quad (2)$$

To decrease the storage size of the map, we use a ground segmentation algorithm. We first compute the empirical covariance matrix of all the points in the cloud. We then compute the normalized Eigenvectors of $\Sigma : \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ with corresponding Eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$. We use a simple, yet effective, approach to label a point $\mathbf{p}_i = (x, y, z)$ with corresponding normal $\mathbf{n}_i$ as ground. $\mathbf{p}_i$ is considered to belong to the ground surface if it satisfies the following criteria: $\mathbf{n}_i \cdot \mathbf{e}_3 > \cos \alpha_{\mathrm{thres}}$ and $z < z_{\mathrm{thres}}$.

After all ground points have been labeled, we proceed to run the reconstruction algorithm that retains the labels encoded as two different colors of the vertices $\mathbf{p}_i$ in the reconstructed triangles. To simplify the ground mesh, we split the complete mesh into non-ground and ground mesh. For the ground mesh, we contract all vertices to a single vertex that are inside a voxel of a given size s_{voxel}. Then, we filter the ground vertices using an average filter by replacing each vertex $\mathbf{v}_i$ with $\mathbf{v}_i^*$ averaging all adjacent vertices $v_n \in \mathcal{N}$:

$$\mathbf{v}_i^* = \frac{\mathbf{v}_i + \sum_{n \in \mathcal{N}} \mathbf{v}_n}{|\mathcal{N}| + 1}. \quad (3)$$

After averaging, invalid edges are removed. Once the ground surface has been simplified, it is combined with the rest of the mesh with no further processing. This simplification allows us to decrease the size of the mesh model to about 50% of its original size.

The PSR algorithm provides a global solution that considers the whole input data at once without resorting to heuristic partitioning [4] or blending [1], [22]. Therefore, instead of building the map incrementally [28], we aggregate all the oriented point clouds into a global reference frame using map poses, then, this oriented point cloud map is directly fed to the PSR algorithm, yielding a globally consistent triangular mesh of the environment.

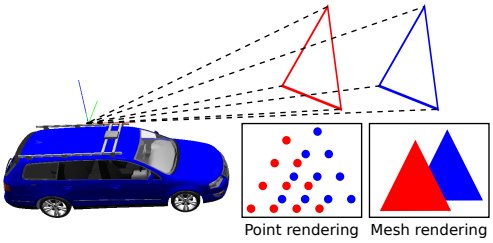


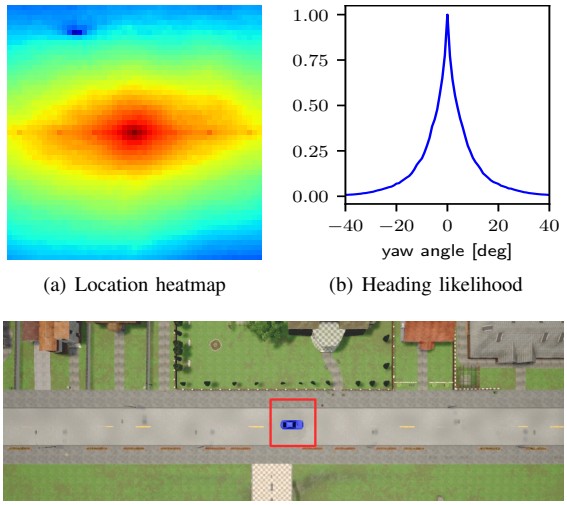
Fig. 3: Rendering example. In contrast to a map represented by a point cloud, triangular meshes are smoother and more compact. During the projection, for each triangle, only three vertices need to be projected. Moreover, triangles can better represent the occlusion relationship between different objects.

C. Rendering Synthetic Range Images

Given a particle j with its state vector (x^j, y^j, θ^j) and the triangular mesh map $\mathcal{M}$, we use OpenGL to render a synthetic range image for that particle. Using the spherical projection, cf. Eq. (1), we project vertices of the triangles from the given particle pose and let OpenGL shade the triangle surface considering the occlusion, as shown in Fig. 3. To further accelerate rendering, we render batches of range images for multiple particles using instancing of the map, which allows us to avoid reading vertex positions multiple times and minimizes the number of draw calls.

D. Monte Carlo Localization

Monte Carlo localization (MCL) is commonly implemented using a particle filter [8]. In our case, each particle represents a hypothesis for the autonomous vehicle's 2D pose $\mathbf{x}_t = (x, y, \theta)_t$ at time t. When the robot moves, the pose of each particle is updated based on a motion model with the control input $\mathbf{u}_t$. The expected observation from the predicted pose of each particle in the map $\mathcal{M}$ is then compared to the actual observation $\mathbf{z}_t$ acquired by the robot to update the particle's weight based on an observation model. Particles are resampled according to their weight distribution



(a) Location heatmap (b) Heading likelihood

(c) Scene used to generate the heatmap

Fig. 4: Range image-based observation model. (a) A local heatmap shows the location likelihood of the scan at the car's position with respect to the map with the same heading. Red shades correspond to higher weights. (b) Heading likelihood of the observation model when changing the yaw angle with the same location. (c) A top-down view of the Carla scene used in this example.

and resampling is triggered whenever the effective number of particles drops below a specific threshold [11]. After several iterations of this procedure, the particles eventually converge around the true pose.

MCL realizes a recursive Bayesian filter estimating a probability density $p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ over the pose $\mathbf{x_t}$ given all observations $\mathbf{z}_{1:t}$ up to time t and motion controls $\mathbf{u}_{1:t}$ up to time t. This posterior is updated as follows:

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \; p(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M}) \cdot \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}) \; p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \; d\mathbf{x}_{t-1}, \quad (4)$$

where η is a normalization constant, $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$ is the motion model, $p(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M})$ is the observation model, and $p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})$ is the probability distribution for the prior state $\mathbf{v}_{t-1}$.

In this work, we focus on the observation model and employ a standard motion model for vehicles [25].

E. Range Image-based Observation Model

Based on the generated range image from the current LiDAR scan and the rendered synthetic range images for all particles, we design an observation model.

Each particle j represents a pose hypothesis $\mathbf{x}_t^j = (x, y, \theta)_t^j$ at time t. Given the corresponding synthetic rendered range image $\mathbf{z}^j$ for the j-th particle rendered at the particle's pose hypothesis, we compare it to the current range image $\mathbf{z}_t$ generated from the LiDAR point cloud. The likelihood $p\left(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M}\right)$ of the j-th particle is then approximated using a Gaussian distribution:

$$p\left(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M}\right) \propto \exp\left(-\frac{1}{2}\frac{d\left(\mathbf{z}_t, \mathbf{z}^j\right)^2}{\sigma_d^2}\right), \quad (5)$$

where d corresponds to the difference between or similarity of the range images $\mathbf{z}_t$ and $\mathbf{z}^j$.

There are several ways to calculate this similarity. For example, we could directly compare two range images at pixel level with absolute differences or using a cross-correlation. One could also generate features and compare two images in the feature space. Recently, there are also many deep learning-based algorithms proposed [12], [6], [24], [29].

In this paper, our goal is to investigate the use of range images generated from LiDAR scans and triangular meshes for a Monte Carlo localization system and the particular choice of similarity computation is application dependent. To keep the whole system fast and easy to use, we opted for a fast-to-compute and effective method and use $d = N^{-1} \sum |\mathbf{z}_t - \mathbf{z}^j|$, i.e., the mean of the absolute pixel-wise differences, where N is the number of valid pixels in the current range image. Our results show that this choice is effective and generalizes well to datasets collected in different environments with different types of LiDAR sensors (see Sec. IV-B).

Fig. 4 shows the probabilities in a local area calculated by our proposed observation model and shows that it encodes the pose hypotheses very well. In contrast to our prior work [5], which decouples the observation model into two parts, location likelihood and heading likelihood, the proposed observation model can estimate the likelihood for the whole state space $\mathbf{x}_t = (x, y, \theta)_t$ at once using one model, which is elegant and fast.

F. Tiled Map Representation

We split the global mesh-based map into tiles to accelerate the Monte Carlo localization by more efficient rendering. We only use parts of the mesh associated to a tile, which are close to the particle position. Besides more efficient rendering, we also use tiles to determine when the localization has converged. If all particles are localized in at most N_{conv} tiles, we assume that the localization has converged and reduce the number of particles to track the pose. Tiles also enable the runtime of our method to be independent of the size of the whole environment after converging.

IV. Experimental Evaluation

We present our experiments to show the capabilities of our method and to support our claims, that our approach is able to: (i) achieve global localization accurately and reliably using 3D LiDAR data, (ii) can be used for different types of LiDAR scanners, and, (iii) generalize well to different environments without changing parameters.

Implementation Details. We implement our code based on Python and OpenGL. To generate a triangular mesh map, we use the PSR implementation from Open3D [34]. For ground point extraction, we use $\alpha_{\text{thres}} = 30°$ and z_{thres} as the sensor mounted height, and employ a voxel grid with voxel sizes $s_{\text{voxel}} = 1.0\,m$. We use tiles of size $100 \times 100\,\text{m}^2$ and we reduce the number of particles from initially $10,000$ to 100 particles after convergence, i.e., only $N_{\text{conv}} = 1$ tile is covered by particles. The size of the tile map and the reduced number of particles are trade-offs between runtime

TABLE I: Dataset Overview

Dataset	Sensor	Sequence	Acquisition time	Length
Carla (Simulator)	8 - 128 beam LiDAR	map (0.2 Gb)	n/a	3.5 km
		00	n/a	0.7 km
IPB-Car (Germany)	Ouster 64	map (0.8 Gb)	02/2020	6.2 km
		00	09/2019	1.7 km
		01	11/2019	1.9 km
MulRan (Korea)	Ouster 64	map (0.5 Gb)	08/2019	6.0 km
		00	06/2019	6.1 km
Apollo (U.S.)	Velodyne 64	map (5.4 Gb)	09/2018	44.8 km
		00	10/2018	8.8 km

and accuracy. We set $\sigma_d = 5$ in Eq. (5), and we only update the weights of particles when the car is moving. All parameters are tuned with one dataset (IPB-Car) and kept the same for all other experiments with different datasets and sensors.

Datasets. We evaluate the generalization ability of our method using multiple datasets, including Carla [9], IPB-Car [6], MulRan (KAIST) [14] and Apollo (Columbia-Park) [35]. These datasets are collected in different environments with different types of LiDAR scanners over different times, see Tab. I for more details. For the Carla simulator, we added objects for the test sequences, which are not present in the map to simulate a changing environment like in the real datasets. For all experiments on different datasets, we only change the intrinsic and extrinsic calibration parameters of the LiDAR sensors for the generation of the range images and keep all other parameters, especially those of the MCL, the same.

Baselines. In the following experiments, we use the same MCL framework and only change the observation models. We compare our method with three baseline observation models: the typical beam-end model [25], a histogram-based model derived from the work of Röhling et al. [20], and a deep learning-based model [6].

The beam-end observation model is often used for 2D LiDAR data. For 3D LiDAR scans, it needs many more particles to make sure that it converges to the correct pose, which causes the computation time to increase substantially. In this paper, we implement the beam-end model with a down-sampled point cloud map using voxelization with a resolution of 10 cm.

Our second baseline for comparison is inspired by Röhling et al. [20], which exploit the use of similarity measures on histograms extracted from 3D LiDAR data.

The third baseline is the overlap-based localization [6]. It uses a deep neural network to estimate the overlap and yaw angle offset between a query scan and map data, and on top of this builds an observation model for MCL. The overlap-based method utilizes a grid map and stores virtual frames for each grid cell. We refer to the corresponding paper [6] for more details.

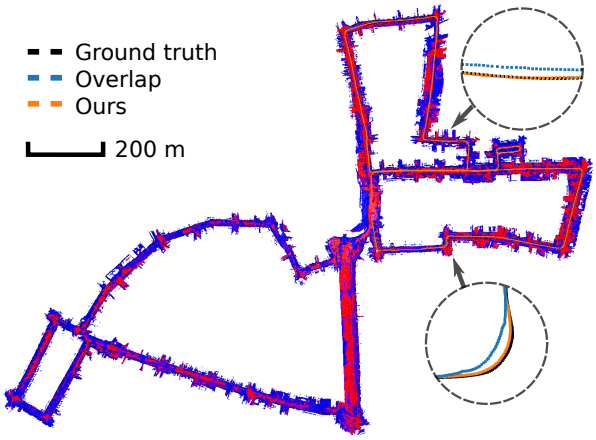


Fig. 5: Localization results using $10,000$ particles on the IPB-Car dataset. Shown are the mesh map, the ground truth trajectory (black), the overlap-based result (blue), and the result of our proposed method (orange).

A. Localization Performance

The experiment presented in this section investigates the localization performance of our approach. It supports the claim that our approach achieves global localization accurately and reliably using 3D LiDAR data.

For qualitative results, we show the trajectories of the localization results tested on the IPB-Car dataset in Fig. 5. The results illustrate that the proposed method localizes well in the map using only LiDAR data collected in dynamic environments at different times. Comparing to the baseline methods, the proposed method tracks the pose more accurately.

For quantitative results, we first calculate the success rate for different methods with different numbers of particles comparing our approach to the aforementioned methods, see Fig. 6. We tested the methods using five different numbers of particles $N = \{1,000,\ 5,000,\ 10,000,\ 50,000,\ 100,000\}$. For each setup, we sample 10 trajectories and perform global localization. The x-axis represents the number of particles, while the y-axis is the success rate of different setups. The success rate for a specific setup of one method is calculated using the number of success cases divided by the total number of the tests. To decide whether one test is successful or not, we check the location error by every 100 frames after convergence. If the location error is smaller than 5 m, we count this run as a success.

Quantitative results of localization accuracy are shown in Tab. II. The upper part shows the location error of all methods tested with both sequences. The location error is defined as the root mean square error (RMSE) of each test in terms of (x, y) Euclidean error with respect to the ground truth poses. It shows the mean and the standard deviation of the error for each observation model. Note that the location error is only calculated for success cases with $10,000$ particles. The lower part shows the yaw angle error. It is the RMSE of each test in terms of yaw angle error with respect to the ground truth poses. The table shows the mean

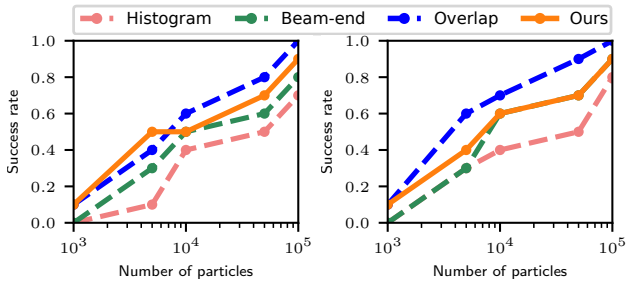


Fig. 6: Success rate of the different observation models for 10 globalization runs. Here, we use sequence 00 (left) and sequence 01 (right) to localize in the map of the IPB-Car dataset.

TABLE II: Localization results on the IPB-Car dataset

Sequence	Location error [m]			
	Beam-end	Histogram-based	Overlap-based	Ours
00	0.92 ± 0.27	1.85 ± 0.34	0.81 ± 0.13	$\mathbf{0.66 \pm 0.12}$
01	0.67 ± 0.11	1.86 ± 0.34	0.88 ± 0.07	$\mathbf{0.44 \pm 0.03}$
Sequence	Yaw angle error [deg]			
	Beam-end	Histogram-based	Overlap-based	Ours
00	1.87 ± 0.47	3.10 ± 3.07	1.74 ± 0.11	$\mathbf{1.69 \pm 0.11}$
01	2.10 ± 0.59	3.11 ± 3.08	$\mathbf{1.88 \pm 0.09}$	2.53 ± 0.79

and the standard deviation of the error for each observation model. As before, the yaw angle error is also only calculated for cases in which the global localization converged with $10,000$ particles.

The quantitative results show that our method outperforms all baseline methods in location accuracy while achieving similar heading accuracy. The reason is that our method uses online rendered range images and does not rely on discrete grids. Therefore, our method will not be affected by the resolution of the grid. However, this means that our method requires more particles to achieve the same success rate. Thus, we use a large amount of particles for initialization. It will then achieve a high success rate while without influencing the runtime after convergence, due to the use of tiles, see in Sec. IV-C.

B. Generalization

The experiment supports the claim that our method is able to use different types of LiDAR sensors to localize in the same mesh map. We test 5 different types of LiDAR sensors in the Carla simulator, including Quanergy MQ-8 (8-beams), Velodyne Puck (16-beams), Velodyne HDL-32E (32-beams), Velodyne HDL-64E (64-beams), and Ouster OS1-128 (128-beams). We use the real parameters from the aforementioned LiDAR sensors, which include the number of beams and the field of view. As shown in Tab. III, our method works well with all different types of sensors and achieves good localization results even with relatively sparse scans (location RMSE of $0.48\,\mathrm{m}$ with the 8-beam LiDAR).

Tab. III and Fig. 7 also verify the claim that our method generalizes well over different environments. We test our method on both the MulRan and Apollo datasets with the same parameters used in the Carla and IPB-Car dataset. Our method works well also in Korean and U.S. urban environments.

TABLE III: Localization results on datasets using different sensors.

Dataset	Scanner	Location RMSE [m]	Yaw angle RMSE [deg]
Carla	8-beams	0.48	3.87
	16-beams	0.43	3.87
	32-beams	0.42	3.40
	64-beams	0.36	3.46
	128-beams	0.33	3.31
MulRan	Velodyne 64	0.83	3.14
Apollo	Ouster 64	0.57	3.40

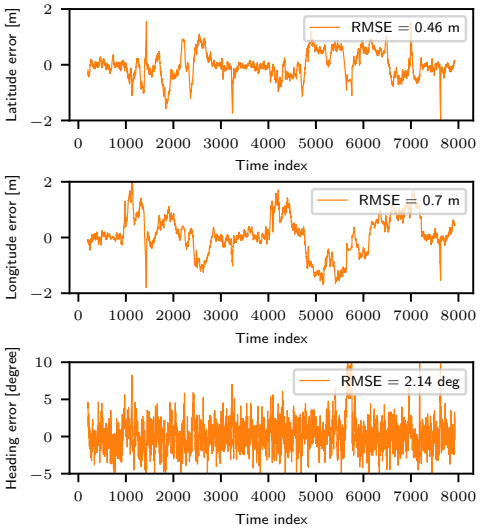


Fig. 7: Localization results on the MulRan dataset w.r.t. provided GPS locations. The top figure shows the latitude error, the middle figure shows the longitude error, and the bottom figure shows the heading error.

C. Runtime

Here we show that our approach runs fast enough to support online processing on the robot at sensor frame rate. We tested our method on a regular computer with an Intel i7-8700 with 3.2 GHz and an Nvidia GeForce GTX 1080 Ti with 11 GB of memory. On the Carla dataset, before convergence, the maximum time for one query frame is $56.7\,\mathrm{s}$ with $10,000$ particles. After convergence, the average frame rates of our method is $21.8\,\mathrm{Hz}$ with 100 particles and the tile maps size of $100 \times 100\,\mathrm{m}^2$.

V. Conclusion

In this paper, we presented a novel range image-based online LiDAR localization approach. Our method exploits range images generated from LiDAR scans and a triangular mesh-based map representation. This allows us to localize autonomous systems in the given map successfully. We implemented and evaluated our approach on different datasets and provided comparisons to other existing techniques. The experiments suggest that our method is able to achieve global localization reliably and accurately. Moreover, our method generalizes well to different environments and can be used with different LiDAR sensors. After the localization convergence, our method can operate online at the sensor frame rate.

REFERENCES

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Point set surfaces. In *Proc. of the IEEE Visualization, (VIS)*, pages 21–29. IEEE, 2001.

[2] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[3] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric Localization with Scale-Invariant Visual Features using a Single Perspective Camera. In H.I. Christiensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer Tracts in Advanced Robotics*, pages 143–157. Springer Verlag, 2006.

[4] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 67–76, 2001.

[5] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proc. of Robotics: Science and Systems (RSS)*, 2020.

[6] X. Chen, T. Läbe, L. Nardi, J. Behley, and C. Stachniss. Learning an Overlap-based Observation Model for 3D LiDAR Localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[7] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[8] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 1999.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2017.

[10] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, 1999(343-349), 1999.

[11] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Trans. on Robotics (TRO)*, 23(1):34–46, 2007.

[12] I. Andrei I.A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun. Learning to Localize Using a LiDAR Intensity Map. In *Proc. of the Second Conference on Robot Learning (CoRL)*, pages 605–616, 2018.

[13] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[14] G. Kim, Y.S. Park, Y. Cho, J. Jeong, and A. Kim. Mulran: Multimodal range dataset for urban place recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[15] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. Autonomous Robot Navigation in Highly Populated Pedestrian Zones. *Journal of Field Robotics (JFR)*, 2014.

[16] J. Levinson, M. Montemerlo, and S. Thrun. Map-Based Precision Vehicle Localization in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[17] W. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mattyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky, and R. Urtasun. Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[18] Ch. Merfels and C. Stachniss. Pose fusion with chain pose graphs for automated driving. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[19] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[20] T. Röhling, J. Mack, and D. Schulz. A Fast Histogram-Based Similarity Measure for Detecting Loop Closures in 3-D LIDAR Data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 736–741, 2015.

[21] A. Schaefer, D. Büscher, J. Vertens, L. Luft, and W. Burgard. Long-term urban vehicle localization using pole landmarks extracted from 3-D lidar scans. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, pages 1–7, 2019.

[22] C. Shen, J.F. O'Brien, and J.R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 896–904, 2004.

[23] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place Recognition in 3D Scans Using a Combination of Bag of Words and Point Feature Based Relative Pose Estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[24] L. Sun, D. Adolfsson, M. Magnusson, H. Andreasson, I. Posner, and T. Duckett. Localising Faster: Efficient and precise lidar-based robot localisation in large-scale environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[25] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[26] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128(1-2), 2001.

[27] G. Tinchev, A. Penate-Sanchez, and M. Fallon. Learning to see the wood for the trees: Deep laser localization in urban and natural environments on a CPU. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1327–1334, 2019.

[28] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss. Poisson Surface Reconstruction for LiDAR Odometry and Mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[29] X. Wei, I. A. Bârsan, S. Wang, J. Martinez, and R. Urtasun. Learning to Localize Through Compressed Binary Maps. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10316–10324, 2019.

[30] D. Wilbers, Ch. Merfels, and C. Stachniss. Localization with Sliding Window Factor Graphs on Third-Party Maps for Automated Driving. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[31] R.W. Wolcott and R.M. Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2814–2821, 2015.

[32] F. Yan, O. Vysotska, and C. Stachniss. Global Localization on OpenStreetMap Using 4-bit Semantic Descriptors. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2019.

[33] C. Zhang, M. H. Ang, and D. Rus. Robust lidar localization for autonomous driving in rain. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3409–3415, 2018.

[34] Q.Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[35] Y. Zhou, G. Wan, S. Hou, L. Yu, G. Wang, X. Rui, and S. Song. Da4ad: End-to-end deep attention-based visual localization for autonomous driving. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[K] I. Vizzo, X. Chen, N. Chebrolu, *et al.*, "Poisson Surface Reconstruction for LiDAR Odometry and Mapping," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021. DOI: 10.1109/ICRA48506.2021.9562069

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

Poisson Surface Reconstruction for LiDAR Odometry and Mapping

Ignacio Vizzo Xieyuanli Chen Nived Chebrolu Jens Behley Cyrill Stachniss

***Abstract*— Accurately localizing in and mapping an environment are essential building blocks of most autonomous systems. In this paper, we present a novel approach for LiDAR odometry and mapping, focusing on improving the mapping quality and at the same time estimating the pose of the vehicle. Our approach performs frame-to-mesh ICP, but in contrast to other SLAM approaches, we represent the map as a triangle mesh computed via Poisson surface reconstruction. We perform the surface reconstruction in a sliding window fashion over a sequence of past scans. In this way, we obtain accurate local maps that are well suited for registration and can also be combined into a global map. This enables us to build a 3D map showing more geometric details than common mapping approaches relying on a truncated signed distance function or surfels. Our experimental evaluation shows quantitatively and qualitatively that our maps offer higher geometric accuracies than these other map representations. We also show that our maps are compact and can be used for LiDAR-based odometry estimation with a novel ray-casting-based data association.**

I. Introduction

Without a map of the environment as well as the knowledge about their pose, most autonomous systems cannot navigate effectively. Thus, localization, mapping, as well as simultaneous localization and mapping [4], [34] are important building blocks of autonomous systems.

We employ a rotating LiDAR sensor to map the environment and investigate the usage of an alternative scene representation for mapping and registration. Scene representation is essential since it is used to register incoming scans. To obtain accurate relative pose estimates and compelling mapping results, a scene representation must capture and represent the environment with a high level of detail.

Our paper aims at improving the geometrical accuracy of LiDAR-based mapping while simultaneously estimating the pose of the vehicle with low-drift over time. We achieve this by using a triangle mesh representation computed via the Poisson surface reconstruction technique [15]. This is in contrast to other state-of-the-art approaches which often use either surfels [1] or a truncated signed distance function [7], [23] as a representation, that often provide a rather low reconstruction quality, at least for large outdoor scenes. With our approach, we reconstruct meshes from robotic 3D LIDAR data for outdoor environments with a quality that was previously common only at the object level, for indoor scenes, by using terrestrial scanners, or by aggregating multiple passes over the same scene.

All authors are with the University of Bonn, Germany.

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob and by the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008 (Harmony).

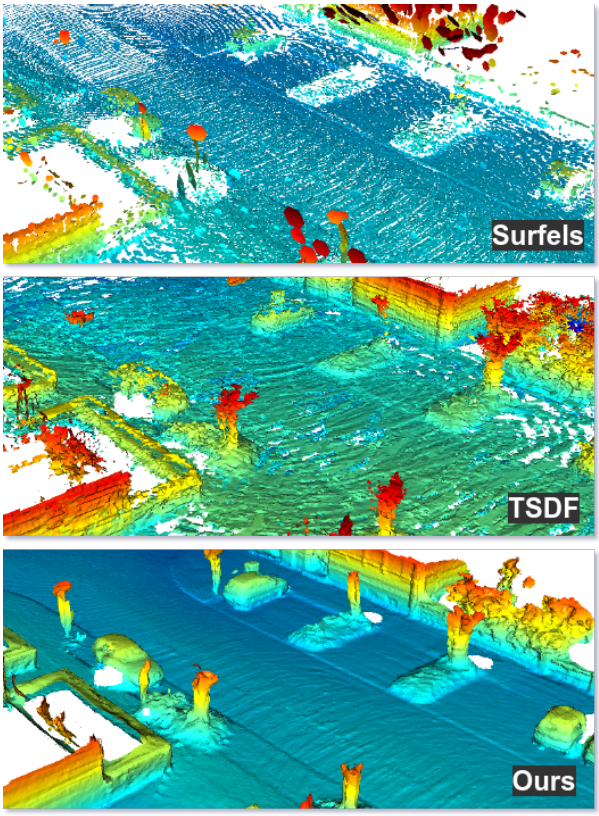


Fig. 1: Qualitative comparison between the different mapping techniques for sequence 00 of the KITTI odometry benchmark [11].

The main contribution of this paper is a novel LiDAR odometry and mapping system that builds upon a surface reconstruction method providing geometrical accurate maps. We aggregate individual scans into a local point cloud and use these to reconstruct a triangular mesh of the scene. Our experimental evaluation shows that such a triangle mesh is well-suited for the registration of 3D LiDAR scans as it is comparably compact, preserves rather detailed structures, and allows for accurate frame-to-mesh registration. This yields a new 3D LiDAR-based mapping approach that provides geometrically accurate maps and can be used for pose estimation, as shown in Fig. 1. We show that the proposed map representation (i) is a geometrically accurate representation of the environment, (ii) has better memory efficiency as compared to other map representations, and (iii) allows for accurate registration of the incoming scans with the model using a novel frame-to-mesh registration algorithm. We support these claims through experimental evaluation on synthetic and real-world data.

978-1-7281-9077-8/21/$31.00 ©2021 IEEE

II. Related Work

Simultaneous localization and mapping (SLAM) has been investigated for decades [4], [34] and here we concentrate on 3D LiDAR-based approaches.

Laser-based SLAM systems either rely on sparse features [39], [22] or perform registration with a dense map representation [1], [8]. Deschaud [8] propose to use implicit moving least squares (IMLS) surfaces for the map representation. Behley and Stachniss [1] perform mapping with a surfel-based map representation. In contrast to these approaches, we register LiDAR scans to a dense map representation using triangular meshes instead of surfels or IMLS surfaces.

A common technique to obtain triangular meshes from point clouds is 3D surface reconstruction [2]. Traditional approaches determine an implicit function modelling the underlying surface, for example, using tangent planes [12], radial basis functions [5], truncated signed distance functions (TSDF) [7], or polynomial representations [17]. Poisson surface reconstruction [14], [15] provides geometrically accurate reconstruction based on this principle.

In contrast to feature-based SLAM systems, dense approaches aim to use all the input data and aggregate them into a dense map representation. A popular approach is using the aforementioned TSDFs popularized by Newcombe et al. [23] for RGB-D mapping. Most TSDF approaches need to know a priori the volume of the environment as they rely on a fixed voxel grid, but Whelan et al. [37] propose to use a rolling grid to alleviate this limitation and store the intermediate results in a triangular mesh. Other approaches use an octree [35] or allocate blocks on demand [24], [26]. Contrary to these methods, we do not make any assumption of the size of the environment to map or make use of any optimized data structures.

In robotics, the triangular mesh representation for reconstruction was employed by Marton et al. [19]. In recent years, such representations were also explored for visual-inertial systems [30], LiDAR-based approaches [6], [31] and purely vision-based systems [27]. In contrast to our work, these approaches typically compute a sparse reconstruction of the traversed environment, while we aim to reconstruct a continuous triangle surface capturing geometric details.

Also, 3D LiDAR-based approaches use variants of TSDF working towards larger volumes and building globally consistent maps [21], [25], [28], [29]. Kühner et al. [18] exploit a TSDF for large outdoor environments, but they need multiple passes over the same scene to obtain compelling reconstruction results. We propose a method that needs only a single pass to obtain a high level of detail.

III. Our Approach

We perform the following three steps for each scan: First, we compute per-point normals, second, we register the scan to the local map, and third, we fuse the registered scan into a global map. We furthermore propose a novel frame-to-mesh registration strategy that exploits that the map is represented by a triangle mesh.

Our approach distinguishes between a local map and the global map. The local map is used to perform the odometry estimation and is build from the last N localized scans. The global map is the aggregated mesh of the entire environment.

A. Normal Computation and Point Cloud Registration

For computing the normals, we project the point cloud into a range image using a spherical projection and estimate the normal vectors using the cross product from neighboring pixels [1]. While this is sometimes less accurate than estimating normals via principal component analysis on the covariance of a point neighborhood, it is far more efficient as it does not need to determine a point neighborhood.

For point cloud registration, we iteratively perform data association of the point cloud with the triangular mesh and determine pose increments to minimize an error metric.

For the data association between point clouds, the closest point association found via neighbor search or projectively [32] is a common choice. We can also use this strategy with our mesh representation by searching neighbors over the triangle vertices, but this is suboptimal as we will show in the experimental evaluation.

We propose to use ray-casting to determine ray-triangle intersections. For each intersection, we extract the point and associated normal of the intersected triangle. To this end, we first apply the last estimated pose at time $t-1$, i.e., $\mathbf{T}_{t-1} \in \mathbb{R}^{4\times4}$, to the current scan as an initial alignment. We then create a set of n rays $\mathcal{R} = \{\mathbf{r}_i\}$. Each ray $\mathbf{r}_i$ is defined by $\mathbf{o}_i + \tau\mathbf{d}_i$ with the origin $\mathbf{o}_i = \mathbf{t}_t^k$ at the currently estimated sensor position and directions $\mathbf{d}_i = \mathbf{T}_t^k\mathbf{p}_i$, passing through all points $\mathbf{p}_i$ of the current scan. Here, $\mathbf{T}_t^k$ is the estimated pose at iteration k and $\mathbf{t}_t^k \in \mathbb{R}^4$ is the translational part of $\mathbf{T}_t^k$.

The intersection of each ray $\mathbf{r}_i \in \mathcal{R}$ with the mesh results in the correspondence for point $\mathbf{p}_i$, denoted as $\mathbf{q}_i$, and the normal of the intersected triangle is the corresponding normal $\mathbf{n}_i$. To compute the relative transformation between the scan and the mesh, we can now use different error metrics $E(\cdot,\cdot)$, such as point-to-point, point-to-plane, or plane-to-plane error [3], [32], [33].

The data association step can also result in wrong correspondences, where a given point from a surface is associated with an intersected point in the mesh from another surface. This typically happens when the ray does not hit any close by surface and hits a far away triangle. Therefore as outlier rejection, we remove correspondences $(\mathbf{p}_i, \mathbf{q}_i) \in \mathcal{C}$ from the set of correspondences $\mathcal{C}$ that satisfy $||\mathbf{p_i} - \mathbf{q_i}|| > \delta_d$. In our implementation, we use $\delta_d = 1\,\text{m}$.

To obtain the estimated pose of the vehicle at time t, we then optimize the following objective:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{(\mathbf{p}_i,\mathbf{q}_i)\in\mathcal{C}} E(\mathbf{T}\mathbf{T}_t^{k-1}\mathbf{p}_i, \mathbf{q}_i), \tag{1}$$

considering the error metric $E(\mathbf{p},\mathbf{q})$ between points $\mathbf{p},\mathbf{q}$ and an increment $\mathbf{T}_t^k = \mathbf{T}^*\mathbf{T}_t^{k-1}$ at iteration k. In our experiments, we found out that the best performing metric is the point-to-plane metric [32]. To reduce the influence of

outliers that were not filtered by the outlier rejection, we use a Huber weighting kernel [13].

The main advantage of the proposed data association is that it does not need to compute nearest neighbors, instead, the association step exploits the map representation through ray-casting the mesh, which as shown in the experiments can be executed faster, especially when dealing with high-resolution meshes. Nonetheless, this approach does not properly handle large rotational motions and requires a good initial estimate to converge, even though this also holds for nearest neighbor approaches.

B. Meshing Algorithm

A common technique to performing 3D surface reconstruction with point sets is to build an implicit function that aims to recover the underlying surface of the input data [12]. Such implicit function f is usually defined as a scalar field in $\mathbb{R}^3$, i.e., $f : \mathbb{R}^3 \mapsto \mathbb{R}$, where typically the zero set of f represents the surface we aim at modeling. A popular technique in robotics and SLAM is to approximate f by the signed distance function [7], i.e., the projected distance, from the sensor to the surface.

Our work, in contrast, explores the use of Poisson surface reconstruction [14], [15] to build consistent, smooth, high-quality maps for mobile robots, in particular, for autonomous vehicles. We refer to the original publications [14], [15] for details on the reconstruction algorithm. Our goal is to investigate the use of triangular meshes in SLAM besides the particular choice of the algorithm being used for the reconstruction[1].

Next, we explain out mesh post-processing. The aforementioned Poisson reconstruction is designed to recover closed surfaces of a single object in 3D as illustrated in Fig. 2. Our 3D world, especially outside environments, is not composed of closed surfaces. Therefore, we need to refine the reconstructed surface and perform a post-processing step, which involves removing vertices with low density. The density $\delta(\mathbf{v})$ of a vertex $\mathbf{v}$ on the mesh measures how many points from the input point cloud support the vertex $\mathbf{v}$. Intuitively, a low value means that the vertex is only supported by a low number of points, and therefore, is not densely measured in the original LiDAR scan or not measured at all (as the Poisson surface reconstruction algorithm will also extrapolate points where there is no data). After reconstructing the mesh, we compute the distribution of the per-vertex densities, as illustrated by the histogram in Fig. 2, right to the legend. The vertices of interest have a high density, i.e. those that are closer in space to point cloud data and are colored in yellow to red in the figure. We trim away the low-density vertices independently of the size of the triangles of the mesh. We make that decision only based on the density $\delta(\mathbf{v})$ of a vertex. We consider the cumulative histogram of densities starting with the highest density values and trim those vertices that belong to the last

[1]We only consider horizontally placed LiDAR sensors, therefore, we do not examine LiDARs shooting towards the sky, like profiler scanners.

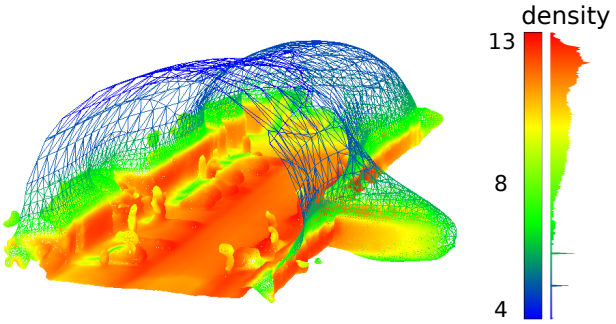


Fig. 2: Cross-section view of the vertex densities for an urban scene reconstruction, where the colors represent the density of the vertices, as shown in the colorbar. We trim away 10% of the vertices based on the density threshold $\delta(\mathbf{v})$.

10%. This means we trim away 10% of the vertices with the smallest density values.

This post-processing produces a tighter reconstruction of the input data, showing little artifacts, which allow us to incrementally build the global mesh as described in Sec. III-C. Note that without this step, it is not possible to build a global map of the environment, due to the artifacts shown in blue in Fig. 2.

Note that, as an interesting side-effect, this density-based filtering also tends to eliminate most of the moving objects in the scene, since 3D points on the surface of moving objects often only support a small number of triangles as the position of the surface changes in every scan. This leads to low-density triangles on moving objects and thus no surface will be reconstructed at these locations.

C. Local and Global Map

In our approach, we distinguish between a local and a global map. The local map is built from the last N aggregated scans. The global map is only used for visualization and for reporting the final output but is not used inside our approach, which will change in case we add loop closing. A new mesh is reconstructed from the local map each time a new LiDAR frame has been registered to the local map. This produces a rolling grid-like mesh that moves with the estimated pose of the vehicle and stores enough information for the registration of new incoming scans. During the initial N scans, we disable the mesh reconstruction module and rely on standard point-to-plane ICP for estimating the pose of the vehicle. After M scans have been registered, the last generated local mesh is integrated into the global mesh map. This means that the global mesh will be updated only after M scans have arrived and registered (in contrast to the local mesh that is updated every time a new scan arrives). To do so, we add all the triangles in the local mesh to the global one and then we remove the duplicated triangles that can occur due to overlaps in the local map region. In our implementation we use $N = M = 30$.

IV. Experimental Evaluation

We implemented our approach on top of the Open3D library [40] and use Intel's Embree library [36] for effi-

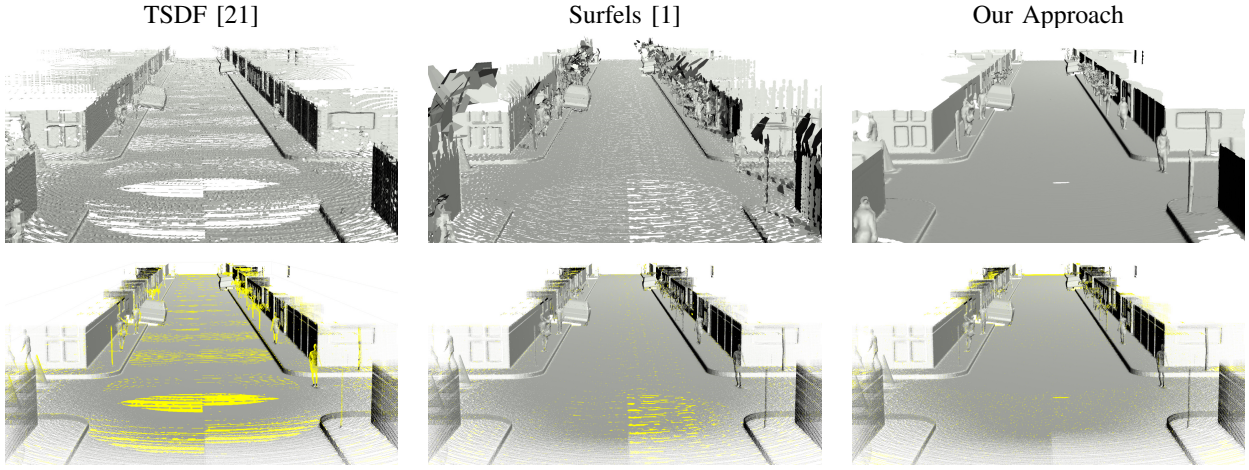


Fig. 3: Qualitative examples showing the map accuracy. The first row exhibits the three map representations, TSDF, surfels and our approach, respectively. The second row depicts the dense GT point clouds used to compute the metrics in Tab. I. Along with the GT clouds, we highlight in yellow the points in the GT model whose distances to the closest point in the built models shown in the first row are greater than $\delta_d = 3\,\text{cm}$. Intuitively, the greater number of yellow points, the more mistakes or gaps a model contains.

cient ray-to-triangle intersection queries. Our algorithm runs entirely on CPU and was tested on an Intel Xeon W-2145 with 8 cores @3.70 GHz with 32 GB RAM. The source code of our approach is available at: `https://github.com/PRBonn/puma`.

For all our experiments we used the default SuMa settings and a voxel size of $0.10\,\text{m}$ for TSDF. For our approach, we set the depth of the octree used in the Poisson surface reconstruction [14] to $\Delta_{\text{tree}} = 10$.

A. Datasets

We use simulated and real-world data for our evaluation. For assessing the accuracy of our map, we need ground truth data and thus we generate synthetic sensor data for an urban environment[2]. To obtain the ground truth (GT) model of the scene, we sample points from the CAD using a virtual LiDAR sensor model with the same field of view and properties as the one used to obtain the scans, however using 320 beams instead of 64. This way we obtain a dense ground truth point cloud but only from the parts of the scene that can be observed with the sensor which is key to a fair comparison of the built 3D model. The final GT point cloud contains 62.5 million points and is shown in the second row of Fig. 3.

For real-world experiments, we use the odometry benchmark from the KITTI Dataset [11].

B. Mapping Accuracy

The first set of experiments is designed to show the geometrical accuracy of our map representation. We compare our mesh generation pipeline with two commonly used map representations: surfels [1] and TDSF [21]. To decouple errors in the resulting map caused by the representation itself and the pose estimation accuracy of the SLAM approaches, we use the same poses provided by the ground truth for all systems for a fair comparison.

[2]The data is available at `http://www.ipb.uni-bonn.de/data/mai-city-dataset`.

Method	Chamfer Distance $d_{\mathcal{CD}}(\mathcal{P},\mathcal{G})$	Precision $P(\delta_d)$	Recall $R(\delta_d)$	F-score $F(\delta_d)$
TSDF [21]	0.66	79.96	85.42	82.6
Surfels [1]	0.11	75.54	98.43	85.48
Ours	**0.05**	**93.28**	**98.69**	**95.91**

TABLE I: Distance evaluation metrics for mapping accuracy. In all experiments $\delta_d = 0.03\,\text{m}$

For a quantitative evaluation of the accuracy of the mapping systems, we densely sample the map representations with a density of 1,000 points per cm^2. Sampling the evaluated map representation allows us to have a fair comparison between each approach and the GT model using standard point-cloud metrics [16], [9].

For our evaluation, we use the following metrics. Let $\mathcal{P}$ be the point cloud sampled from the map and let $\mathcal{G}$ be the GT point cloud. For a point $\mathbf{p} \in \mathcal{P}$, we define the distance to the GT model as:

$$d(\mathbf{p}, \mathcal{G}) = \min_{\mathbf{g} \in \mathcal{G}} \|\mathbf{p} - \mathbf{g}\|, \tag{2}$$

where $\|\cdot\|$ is the L2 norm. Analogously, we define the distance for a point $\mathbf{g} \in \mathcal{G}$ to the reconstructed map as $d(\mathbf{g}, \mathcal{P}) = \min_{\mathbf{p} \in \mathcal{P}} \|\mathbf{g} - \mathbf{p}\|$. For computing precision, recall, and f-score metrics, we follow exactly the work by Knapitsch et al. [16].

We also define the Chamfer distance [9] as:

$$d_{\mathcal{CD}}(\mathcal{P}, \mathcal{G}) = \frac{1}{2|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} d(\mathbf{p}, \mathcal{G})^2 + \frac{1}{2|\mathcal{G}|} \sum_{\mathbf{g} \in \mathcal{G}} d(\mathbf{g}, \mathcal{P})^2. \tag{3}$$

The results for the mapping accuracy are shown in Tab. I. We are especially interested in the areas where the 3D models deviate from the ground truth. Therefore, we highlighted all points that have an error of $\delta_d \geq 3\,\text{cm}$ in *yellow* (second row of Fig. 3).

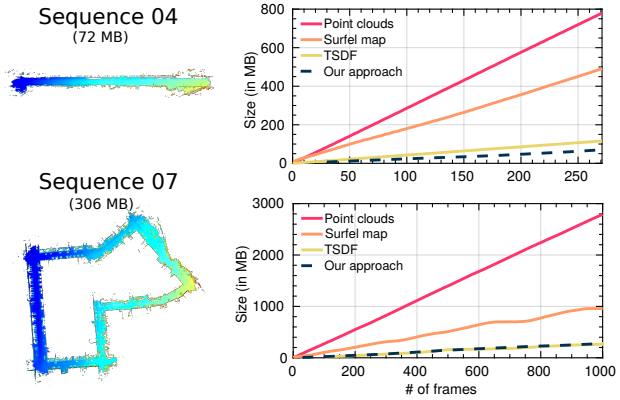


Fig. 4: Memory consumption for different map representations for two KITTI sequences. On the left, we show the triangular mesh maps built by our approach along with their final size. Sequence 04 is a country environment and sequence 07 was recorded in an urban scenario. We see that point clouds and surfels result in higher memory usage. TSDF performs similarly to our approach in terms of memory usage, but as seen in Fig. 3, our approach outperforms TSDFs in terms of geometrical accuracy.

We can see that the TSDF-based approach fails to reconstruct important distinctive objects of the scene such as trees, pedestrians, or poles. Is important to remark that this is a limitation of the method as the framework used [20] does not drop any input scans. The surfel-based approach often provides more accurate reconstructions, however, it also adds spurious artifacts to the reconstructed scene. Our approach outperforms both in terms of accuracy and correctness of the model since it does not miss important features and does not add artifacts at the same time.

C. Memory Efficiency

To use these high definition maps in practice, it is necessary to be able to represent this map efficiently. The second evaluation is designed to investigate the memory requirements of different map representations. We show that our map representation, based on triangular meshes, is a good choice in terms of a trade-off between geometrical accuracy and the corresponding memory footprint. We evaluate the amount of memory needed to represent a given map using our approach, TSDFs [21], surfels [1], and the LiDAR point clouds, only considering the raw geometric model, not any other additional information (such as normals, color information, etc.) that could be stored. The voxel size of $0.10\,\text{m}$ for the TSDF-based [21] approach is chosen such that the spatial resolution of the extracted mesh results similar to the one built with our approach. SuMa [1] does not provide a mechanism to control its map size.

For a mesh-based map representation, the minimum amount of memory needed to represent a triangular mesh is given by $N_v\, v_{\text{size}} + N_f\, f_{\text{size}}$, where N_v and N_f are the numbers of vertices and faces in the model respectively and v_{size} and f_{size} are the size of a vertex and a triangular face respectively.

In the case of a TSDF based map representation, Whelan et al. [37] shows that it is not directly feasible to store the TSDF volume as a map representation for large outdoor environments in general. In practice, we need to extract the triangular mesh from the TSDF volume to store these maps. As a result, while comparing the memory footprint of the TSDF approach, we use the size of the extracted mesh instead of the TSDF volume. For a surfel, we need to represent the center position of the surfel, the normal, and a radius. Therefore, the size of a surfel map is given by $N_S\,(c_{\text{size}} + n_{\text{size}} + r_{\text{size}})$, where N_S is the number of surfels, c_{size} is the size of the center position, n_{size} is the size of the normal vector, and r_{size} is the size of the radius. For the case of a point cloud map, the size of the map is given by $N_P\, p_{\text{size}}$, where N_P is the number of points and p_{size} is the size of a point.

Fig. 4 shows the memory consumption of the different approaches over time for two different sequences of the KITTI odometry benchmark. We see that our approach scales well with the number of input scans, while surfel-based maps or point cloud maps require much more memory, which makes it not feasible to run on mobile platforms. TSDF-based maps exhibit similar memory consumption, however with a reduced mapping accuracy, as shown in Sec. IV-B.

D. Odometry and Localization Accuracy

In this experiment, we illustrate that our approach is well-suited for estimating the pose of the vehicle by registering the LiDAR scans using our map representation. We evaluate our performance on the KITTI dataset [11] where we estimate the poses by registering the 3D LiDAR scans incrementally. We compare our approach to commonly used registration algorithms based on ICP, namely point-to-point ICP [3], point-to-plane ICP [32] and generalized-ICP [33]. For all these three approaches, we use the same optimization framework described in Sec. III-A, including also a Huber loss [13] function to reject outliers. As we do with our approach, we also initialize the ICP with the last increment estimate obtained at time $t-1$. Additionally, we consider a different scenario for these three methods, instead of performing frame-to-frame ICP, we perform frame-to-model ICP, where the *model* is the last N aggregated scans. This is represented as *point cloud map* in Tab. II. This way we can evaluate the benefits of running the reconstruction algorithm on this local map or not. For performing data-associations on the three ICP variants, we employ nearest neighbors search. Lastly, we also consider a different data-association scenario, the projective data association. For this, we compare our approach to *frame-to-frame* SuMA [1]. The TSDF [20] framework does not provide pose estimates and therefore is omitted in this experiment. It is important to remark that to compare all methods equally, we use the same normal computation for all approaches as described in Sec. III-A. This normal computation will generally decrease the performance of any normal-based metric for estimating the pose.

In terms of estimation performance, we can see in Tab. II that our approach provides solid pose estimate performance when comparing the poses provided by our system to the ground truth poses provided by KITTI.

Map	Method	DA	avg.	00	01	02	03	04	05	06	07	08	09	10
None	point-to-point ICP [3]	NN	4.86	3.65	15.40	4.60	5.74	2.44	3.38	2.94	4.92	3.75	3.38	3.24
			1.71	1.62	1.06	1.70	2.70	1.37	1.47	1.36	2.43	1.78	1.59	1.75
	point-to-plane ICP [32]	NN	7.60	9.12	10.00	6.19	6.04	4.51	7.69	8.24	5.53	8.70	9.37	8.18
			3.49	3.89	1.53	2.42	3.05	2.65	3.79	4.10	4.17	3.97	3.50	5.29
	GICP [33]	NN	14.35	7.35	73.10	14.40	9.37	13.60	5.23	2.23	6.40	7.27	10.90	8.00
			4.78	3.00	24.10	4.04	2.85	2.15	2.14	1.06	2.87	3.17	3.78	3.38
	SuMa [1]	Proj.	2.93	2.09	4.05	2.30	**1.43**	11.90	1.46	0.95	1.75	2.53	1.92	1.81
			0.92	0.93	1.22	0.79	**0.75**	1.06	0.79	0.64	1.17	0.96	0.78	0.97
Point cloud	point-to-point ICP [3]	NN	29.98	9.25	93.20	30.40	10.90	91.40	**0.92**	33.90	8.35	2.81	33.90	14.80
			2.61	1.58	8.75	5.04	2.21	0.99	**0.46**	6.38	0.56	0.68	6.59	1.65
	point-to-plane ICP [32]	NN	18.92	9.99	77.10	11.70	2.31	70.00	2.62	1.84	1.79	3.67	17.40	9.70
			4.01	4.29	17.60	2.70	1.01	5.21	1.17	0.84	1.16	1.47	5.47	3.15
	GICP [33]	NN	20.43	4.34	93.10	10.70	2.21	83.70	1.56	1.42	1.19	2.33	21.80	2.37
			2.76	0.94	17.20	2.25	1.19	1.81	0.73	0.68	0.78	0.95	2.91	0.90
Mesh	Ours ($\Delta_{\text{tree}} = 10$)	NN	2.15	3.14	4.32	1.91	1.34	2.09	1.56	1.41	1.88	1.97	1.80	2.21
			1.14	1.49	1.04	0.73	1.07	1.46	1.07	0.72	1.36	1.10	0.82	1.67
	Ours ($\Delta_{\text{tree}} = 10$)	**RC**	**1.55**	**1.46**	**3.38**	**1.86**	1.60	**1.63**	1.20	**0.88**	**0.72**	**1.44**	**1.51**	**1.38**
			0.74	**0.68**	**1.00**	**0.72**	1.10	**0.92**	0.61	**0.42**	**0.55**	**0.61**	**0.66**	**0.84**

TABLE II: Odometry estimation results for the KITTI Odometry benchmark [10]. The rows highlighted in gray correspond to the translational error and the row below to the rotational error. All errors are averaged over trajectories of 100 to 800 m length. The translational error is in % and the relative rotational error in degrees per 100 m. DA for data association, RC for ray casting, NN for nearest neighbor. Bold numbers indicate the best approach for the given sequence.

E. Registration Algorithm

We briefly show that the novel registration scheme depicted in Sec. III-A is superior in terms of accuracy and speed. We compare our registration pipeline against a standard point-to-plane ICP [32], where the source in both cases is the incoming scan from the LiDAR sensor and the target is set to be the triangular mesh built for our approach. We compare two different registration algorithms, one that sample all vertices and normals of the mesh to obtain a point cloud, and the second, that uses our novel data association algorithm based on ray-casting (RC) as explained in Sec. III-A. While traditional ICP makes use of nearest neighbor searches, we perform ray-casting through the mesh to obtain the target correspondences and the normal information. The registration is evaluated by running our registration algorithm in the full KITTI training sequences, registering all LiDAR scans to the local mesh-map (Sec. III-C), using different map resolutions, i.e., by setting different resolutions of the Δ_{tree} on the Poisson surface reconstruction [15]. To investigate the accuracy of the registration method we compute the average translational and rotational errors overall training sequences of the dataset. We see that the proposed registration algorithm scales better when the size of the input mesh increases. The results of this experiment are shown in Tab. III.

		Mesh vertex-sampling			**Mesh ray-casting**		
Δ_{tree}	**Vertices**	t_{err}	r_{err}	runtime	t_{err}	r_{err}	runtime
8	30K	2.82	1.23	682	**1.73**	**0.90**	**395**
9	100K	2.05	1.08	645	**1.53**	**0.74**	**418**
10	300K	2.14	1.17	789	**1.56**	**0.74**	**535**

TABLE III: Ray-casting vs mesh-sampling registration evaluation in the full KITTI [11] training sequences. Relative errors are averaged over trajectories of 100 to 800 m length. Relative translational error (t_{err}) in % and relative rotational error (r_{err}) in degrees per 100 m. The runtime values are expressed in *milliseconds*.

F. Runtime

The pre-processing and the normal estimation take 45 ms on average per scan, and the scan-matching algorithm takes another extra 500 ms. However, the bottleneck is the meshing algorithm which takes on average 5 s when executed on a CPU. This makes our approach infeasible for online operation on autonomous vehicles. In this work, we have primarily focused on the use of triangular meshes in developing SLAM pipelines and show that both the reconstruction quality and the pose estimation accuracy are promising. In future work, we need to investigate techniques for optimizing the meshing algorithm to enable online performance, for instance, running the reconstruction algorithm on the GPU, additionally, GPU-enabled ray tracing engines, like the NVIDIA OptiX™, can be used to accelerate the ray-casting registration algorithm.

V. Conclusion

In this publication, we presented a novel approach for simultaneous odometry and mapping with 3D LiDARs. Our approach performs a novel frame-to-mesh registration but in contrast to other SLAM or odometry and mapping approaches, we represent the map as a triangle mesh estimated using Poisson surface reconstruction in a sliding window over a sequence of past scans. We show that we obtain high-quality local meshes that show more details than common alternative methods such as state-of-the-art TSDF or surfel representations. We also show that our map representation is well-suited for incremental scan registration for pose estimation. For future work, our system can be extended by incorporating loop closures [38] and a pose graph optimization framework for efficient dense map correction.

REFERENCES

[1] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[2] M. Berger, A.Tagliasacchi, L. Seversky, P. Alliez, G. Guennebaud, J. Levine, A. Sharf, and C. Silva. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, volume 36, pages 301–329, 2017.

[3] P.J. Besl and N.D. McKay. A Method for Registration of 3D Shapes. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. *IEEE Trans. on Robotics (TRO)*, 32:1309–1332, 2016.

[5] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 67–76, 2001.

[6] X. Chen, I. Vizzo, T. Läbe, J. Behley, and C. Stachniss. Range Image-based LiDAR Localization for Autonomous Vehicles. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[7] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 303–312. ACM, 1996.

[8] J. Deschaud. Imls-slam: scan-to-model matching based on 3d data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[9] H. Fan, H. Su, and L.J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 605–613, 2017.

[10] M. Gehrig, E. Stumm, T. Hinzmann, and R. Siegwart. Visual Place Recognition with Probabilistic Voting. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[12] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 71–78, 1992.

[13] P. J. Huber. *Robust Statistics*. Wiley, 1981.

[14] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[15] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, 2013.

[16] A. Knapitsch, J. Park, Q. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):1–13, 2017.

[17] R. Kolluri. Provably good moving least squares. *ACM Transactions on Algorithms (TALG)*, 4(2):18, 2008.

[18] T. Kühner and J. Kümmerle. Large-scale volumetric scene reconstruction using lidar. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[19] Z.C. Marton, R.B. Rusu, and M. Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3218–3223. IEEE, 2009.

[20] A. Milan, T. Pham, V.K.B. G, D. Morrison, A.M.W. Tow, L. Liu, J. Erskine, R. Grinover, A. Gurman, T. Hunn, N. Kelly-Boxall, D.Q. Lee, M. McTaggart, G.M. Rallos, A. Razjigaev, T.J. Rowntree, T. Shen, R. Smith, S. Wade-McCue, Z. Zhuang, C. Lehnert, G. Lin, I. Reid, P. Corke, and J. Leitner. Semantic segmentation from limited training data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[21] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena. C-Blox: A Scalable and Consistent TSDF-based Dense Mapping Approach. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[22] F. Moosmann and C. Stiller. Velodyne SLAM. In *Proc. of the IEEE Vehicles Symposium (IV)*, pages 393–398, 2011.

[23] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proc. of the Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011.

[24] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. *Proc. of the SIGGRAPH Asia*, 32(6), 2013.

[25] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwar, and J. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1366–1373, 2017.

[26] E. Palazzolo, J. Behley, P. Lottes, P. Giguere, and C. Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[27] E. Piazza, A. Romanoni, and M. Matteucci. Real-time cpu-based large-scale 3d mesh reconstruction. *arXiv preprint arXiv:1801.05230*, 2018.

[28] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto. Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps. *IEEE Robotics and Automation Letters (RA-L)*, 5(1):227–234, 2019.

[29] L. Roldão, R. Charette, and A. Verroust-Blondet. 3d surface reconstruction from voxel-based lidar data. *arXiv preprint*, (1906.10515v1), 2019.

[30] A. Rosinol, T. Sattler, M. Pollefeys, and L. Carlone. Incremental visual-inertial 3d mesh generation with structural regularities. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 8220–8226. IEEE, 2019.

[31] F. Ruetz, E. Hernández, M. Pfeiffer, H. Oleynikova, M. Cox, T. Lowe, and P. Borges. Ovpc mesh: 3d free-space representation for local ground vehicle navigation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 8648–8654. IEEE, 2019.

[32] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. of Int. Conf. on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[33] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[34] C. Stachniss, J. Leonard, and S. Thrun. *Springer Handbook of Robotics, 2nd edition*, chapter Chapt. 46: Simultaneous Localization and Mapping. Springer Verlag, 2016.

[35] F. Steinbrücker, J. Sturm, and D. Cremers. Volumetric 3D Mapping in Real-Time on a CPU. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2014.

[36] I. Wald, S. Woop, C. Benthin, G.S. Johnson, and M. Ernst. Embree: a kernel framework for efficient cpu ray tracing. *ACM Transactions on Graphics*, 33(4):1–8, 2014.

[37] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially Extended KinectFusion. In *Proc. RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[38] T. Whelan, M Kaess, J. Leonard, and J. McDonald. Deformation-based loop closure for large scale dense rgb-d slam. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[39] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*, 2014.

[40] Q.Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[L] X. Chen, T. Läbe, A. Milioto, *et al.*, "OverlapNet: Loop Closing for LiDAR-based SLAM," in *Proc. of Robotics: Science and Systems (RSS)*, 2020. DOI: 10.15607/RSS.2020.XVI.009

OverlapNet: Loop Closing for LiDAR-based SLAM

Xieyuanli Chen* Thomas Läbe* Andres Milioto* Timo Röhling*,‡
Olga Vysotska†,* Alexandre Haag† Jens Behley* Cyrill Stachniss*

*Photogrammetry & Robotics Lab, University of Bonn, Germany
‡Fraunhofer FKIE, Wachtberg, Germany
†Autonomous Intelligent Driving GmbH, Munich, Germany

***Abstract*—Simultaneous localization and mapping (SLAM) is a fundamental capability required by most autonomous systems. In this paper, we address the problem of loop closing for SLAM based on 3D laser scans recorded by autonomous cars. Our approach utilizes a deep neural network exploiting different cues generated from LiDAR data for finding loop closures. It estimates an image overlap generalized to range images and provides a relative yaw angle estimate between pairs of scans. Based on such predictions, we tackle loop closure detection and integrate our approach into an existing SLAM system to improve its mapping results. We evaluate our approach on sequences of the KITTI odometry benchmark and the Ford campus dataset. We show that our method can effectively detect loop closures surpassing the detection performance of state-of-the-art methods. To highlight the generalization capabilities of our approach, we evaluate our model on the Ford campus dataset while using only KITTI for training. The experiments show that the learned representation is able to provide reliable loop closure candidates, also in unseen environments.**

I. Introduction

Simultaneous localization and mapping or SLAM [1, 29] is an integral part of most robots and autonomous cars. Graph-based SLAM often relies on (i) pose estimation relative to a recent history, which is called odometry or incremental scan matching, and (ii) loop closure detection, which is needed for data association on a global scale. Loop closures enable SLAM approaches to correct accumulated drift resulting in a globally consistent map.

In this paper, we propose a new method to loop closing for laser range scans produced by a rotating 3D LiDAR sensor installed on a wheeled robot or similar vehicle. Instead of using handcrafted features [15, 31], we propose a deep neural network designed to find loop closure candidates. Our network predicts both, a so-called overlap defined on range images and a relative yaw angle between two 3D LiDAR scans recorded with a typical sensor setup often used on automated cars. The concept of overlap has been used in photogrammetry to estimate image overlaps, see also Sec. III-A and III-B, and we use it on LiDAR range images. It is a useful tool for loop closure detection as illustrated in Fig. 1 and can quantify the quality of matches. The yaw estimate serves as an initial guess for a subsequent application of iterative closest point (ICP) [4] to determine the relative pose between scans to derive loop closures constraints for the pose graph optimization. Instead of ICP, one could also use global scan matching [7, 37, 34] to estimate the relative pose between scans.

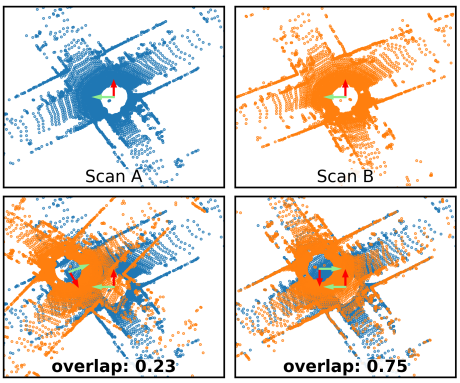


Fig. 1: Overlap of two scans (blue and orange points) at a loop closure location but computed with different relative transformations. The overlap depends on the relative transformation and larger overlap values often correspond to better alignment between the point clouds. Our approach can predict the overlap *without* knowing the relative transformation between the scans.

The main contribution of this paper is a deep neural network that exploits different types of information generated from LiDAR scans to provide overlap and relative yaw angle estimates between pairs of 3D scans. This information includes depth, normals, and intensity or remission values. We additionally exploit a probability distribution over semantic classes that can be computed for each laser beam. Our approach relies on a spherical projection of LiDAR scans, rather than the raw point clouds, which makes the proposed OverlapNet comparably lightweight. We furthermore integrate it into a state-of-the-art SLAM system [3] for loop closure detection and evaluate its performance also with respect to generalization to different environments.

We train the proposed OverlapNet on parts of the KITTI odometry dataset and evaluate it on unseen data.We thoroughly evaluate our approach, provide ablation studies using different modalities, and test the integrated SLAM system in an online manner. Furthermore, we provide results for the Ford campus dataset, which was recorded using a different sensor setup in a different country and a differently structured environment. The experimental results suggest that our method outperforms other state-of-the-art baseline methods and is also able to generalize well to unseen environments.

In sum, our approach is able to (i) predict the overlap and relative yaw angle between pairs of LiDAR scans by

exploiting multiple cues without using relative poses, (ii) combine odometry information with overlap predictions to detect correct loop closure candidates, (iii) improve the overall pose estimation results in a state-of-the-art SLAM system yielding more globally consistent maps, (iv) solve loop closure detection without prior pose information, (v) initialize ICP using the OverlapNet predictions yielding correct scan matching results. The implementation of our approach is available at: https://github.com/PRBonn/OverlapNet

II. Related Work

Loop closure detection using various sensor modalities [16, 28] is a classical topic in robot mapping. We refer to the article by Lowry *et al*. [21] for an overview of approaches using cameras. Here, we mainly concentrate on related work addressing 3D LiDAR-based approaches.

Steder *et al*. [31] propose a place recognition system operating on range images generated from 3D LiDAR data that uses a combination of bag-of-words and a NARF-feature-based [30] relative poses estimation exploiting ideas of FABMAP [11]. Röhling *et al*. [26] present an efficient method for detecting loop closures through the use of similarity measures on histograms extracted from 3D LiDAR scans. The work by He *et al*. [15] presents M2DP, which projects a LiDAR scan into multiple reference planes to generate a descriptor using a density signature of points in each plane. Besides using pure geometric information, there is also work [9, 14] exploiting the remission information, i.e., how well LiDAR beams are reflected by a surface, to create descriptors for localization and loop closure detection with 3D LiDAR data.

Motivated by the success of deep learning in computer vision [20], deep learning-based methods have been proposed recently. Barsan *et al*. [2] propose a deep network-based localization method, which embeds LiDAR sweeps and intensity maps into a joint embedding space and achieves localization by matching between these embeddings. Dubé *et al*. [12] advocate the usage of segments for loop closure detection. Cramariuc *et al*. [10] train a CNN to extract descriptors from segments and use it to retrieve near-by place candidates. Schaupp *et al*. [27] propose a system called OREOS for place recognition, that also estimates the yaw discrepancy between scans. Furthermore, Yin *et al*. [35] develop LocNet, which uses semi-handcrafted feature learning based on a siamese network to solve place recognition. Lu *et al*. [22] proposed L^3-net, which uses 3D convolutions and a recurrent neural network to learn local descriptors for global localization. Uy and Lee [33] proposed PointNetVLAD to generate a global descriptor for 3D point clouds. Kim *et al*. [19] proposed a learning-based descriptor called SCI to solve long-term global localization. Most recently, Sun *et al*. [32] also proposed a learning-based method combined with Monte Carlo localization to achieve a fast global localization.

Contrary to the above-mentioned methods, our method exploits multiple types of information extracted from 3D LiDAR scans, including depth, normal information, intensity/remission and probabilities of semantic classes generated by a semantic segmentation system [23].

Similar to LocNet [35], we also use a siamese network, but we learn features and yield predictions end-to-end. Our network can directly provide estimates for overlap and the relative yaw angle between pairs of LiDAR scans. Different from OREOS [27], our method not only provides loop closures candidates but also an estimate of the matching quality in terms of the overlap.

Recently, Zaganidis *et al*. [36] proposed a Normal Distributions Transform (NDT) histogram-based loop closure detection method, which is also assisted by semantic information. In contrast to ours, their method needs a dense global map and cannot estimate the relative yaw angle.

III. Our Approach

A. The Concept of Overlap

The idea of overlap that we are using here has its origin in the photogrammetry and computer vision community [18]. To successfully match two images and calculate their relative pose, the images must overlap. This can be quantified by defining the overlap percentage as the percentage of pixels in the first image, which can successfully be projected back into the second image without occlusion. Note that this measure is not symmetric: If there is a large scale difference of the image pair, e.g., one image shows a wall and the other shows many buildings around that wall, the overlap percentage for the first to the second image can be large and from the second to the first image low. In this paper, we use the idea of overlap for range images, exploiting the range information explicitly.

For loop closing, a threshold on the overlap percentage can be used to decide whether two LiDAR scans are at the same place and/or a loop closing can be done. For loop closing, this measure maybe even better than the commonly used distance between the recorded positions of a pair of scans, since the positions might be affected by drift and therefore unreliable. The overlap predictions are independent of the relative poses and can be therefore used to find loop closures without knowing the correct relative pose between scans. Fig. 1 shows the overlap of two scans as an example.

B. Definition of the Overlap between Pairs of LiDAR Scans

We use spherical projections of LiDAR scans as input data, which is often used to speed up computations [3, 5, 8]. We project the point cloud $\mathcal{P}$ to a so-called vertex map $\mathcal{V} : \mathbb{R}^2 \mapsto \mathbb{R}^3$, where each pixel is mapped to the nearest 3D point. Each point $\mathbf{p}_i = (x, y, z)$ is converted via the function $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates and finally to image coordinates (u, v) by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y,x)\pi^{-1}\right] w \\ \left[1 - \left(\arcsin(zr^{-1}) + \mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right] h \end{pmatrix}, \quad (1)$$

where $r = ||\mathbf{p}||_2$ is the range, $\mathrm{f} = \mathrm{f}_{\mathrm{up}} + \mathrm{f}_{\mathrm{down}}$ is the vertical field-of-view of the sensor, and w, h are the width and height of the resulting vertex map $\mathcal{V}$.

For a pair of LiDAR scans $\mathcal{P}_1$ and $\mathcal{P}_2$, we generate the corresponding vertex maps $\mathcal{V}_1$, $\mathcal{V}_2$. We denote the sensor-centered coordinate frame at time step t as C_t. Each pixel in coordinate frame C_t is associated with the world frame W by a pose $\mathbf{T}_{WC_t} \in \mathbb{R}^{4\times4}$. Given the poses $\mathbf{T}_{WC_1}$ and $\mathbf{T}_{WC_2}$, we can reproject scan $\mathcal{P}_1$ into the coordinate frame of the other's vertex map $\mathcal{V}_2$ and generate a reprojected vertex map $\mathcal{V}_1'$:

$$\mathcal{V}_1' = \Pi\left(\mathbf{T}_{WC_1}^{-1}\mathbf{T}_{WC_2}\mathcal{P}_1\right). \tag{2}$$

We then calculate the absolute difference of all corresponding pixels in $\mathcal{V}_1'$ and $\mathcal{V}_2$, considering only those pixels that correspond to valid range readings in both range images. The overlap is then calculated as the percentage of all differences in a certain distance ϵ relative to all valid entries, i.e., the overlap of two LiDAR scans $O_{C_1C_2}$ is defined as follows:

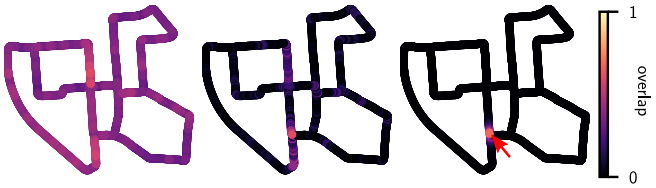
$$O_{C_1C_2} = \frac{\sum_{(u,v)} \mathbb{I}\Big\{ ||\mathcal{V}_1'(u,v) - \mathcal{V}_2(u,v)|| \leq \epsilon \Big\}}{\min\left(\text{valid}(\mathcal{V}_1'), \text{valid}(\mathcal{V}_2)\right)}, \tag{3}$$

where $\mathbb{I}\{a\} = 1$ if a is true and 0 otherwise. valid$(\mathcal{V})$ is the the number of valid pixels in $\mathcal{V}$, since not all pixel might have a valid LiDAR measurement associated after the projection.

We use Eq. (3) only for creating training data, i.e., only positive examples of correct loop closures get a non-zero overlap assigned using the relative poses between scans, as shown in Fig. 2(c). However, when performing loop closure detection for online SLAM, the approximate relative poses from SLAM before loop closure are not accurate enough to calculate usable overlaps by using Eq. (3) because of accumulated drift. We tried directly estimating overlaps using Eq. (3) assuming the relative pose as identity and applying different orientations, e.g., every 30 degrees rotation around the vertical axis, and using the maximum over all these overlaps as an estimate. Fig. 2 shows the estimated overlaps for all scans using a query scan produced by this method and the result of the estimated overlap for all scans using OverlapNet. We leave out the 100 most recent scans because they will not be loop closure candidates. In the case of the exhaustive approach, many wrong loop closure candidates get high overlap values, while our approach performs better since it produces a highly distinctive peak around the correct location. Furthermore, it takes on average 1.2 s to calculate the overlap for one pair of scans using the exhaustive approach, which makes it unusable in real-world scenarios. In contrast, the complete OverlapNet needs on average 17 ms for one pair overlap estimation when using depth and normal information only.

C. Overlap Network Architecture

The overview of the proposed OverlapNet is depicted in Fig. 3. We exploit multiple cues, which can be generated from a single LiDAR scan, including depth, normal, intensity, and semantic class probability information. The depth information is stored in the range map $\mathcal{R}$, which consists of one channel. We use neighborhood information of the vertex map to generate a normal map $\mathcal{N}$, which has three channels encoding the normal coordinates. We directly obtain the intensity information, also called remission, from the sensor and represent the



(a) Exhaustive evaluation of Eq. (3) (b) OverlapNet estimates (c) Ground truth

Fig. 2: Overlap estimations of one frame to all others. The red arrow points out the position of the query scan. If we directly use Eq. (3) to estimate the overlap between two LiDAR scans without knowing the accurate relative poses, it is hard to decide which pairs of scans are true loop closures, since most evaluations of Eq. (3) show high values. In contrast, our OverlapNet can predict the overlaps between two LiDAR scans well.

intensity information as a one-channel intensity map $\mathcal{I}$. The point-wise semantic class probabilities are computed using RangeNet++ [23] and we represent them as a semantic map $\mathcal{S}$. RangeNet++ delivers probabilities for 20 different classes. For efficiency sake, we reduce the 20-dimensional RangeNet++ output to a compressed 3-dimensional vector using principal component analysis. The information is combined into an input tensor of size $64 \times 900 \times D$, where $64, 900$ are the height and width of the inputs, and D depends on the types of data used.

Our proposed OverlapNet is a siamese network architecture [6], which consists of two legs sharing weights and two heads that use the same pair of feature volumes generated by the two legs. The trainable layers are listed in Tab. I.

1) Legs: The proposed OverlapNet has two legs, which have the same architecture and share the same weights. Each leg is a fully convolutional network (FCN) consisting of 11 convolutional layers. This architecture is quite lightweight and generates feature volumes of size $1 \times 360 \times 128$. Note that our range images are cyclic projections and that a change in the yaw angle of the vehicle results in a cyclic column shift of the range image. Thus, the single row in the feature volume can represent a relative yaw angle estimate (because a yaw angle rotation results in a pure horizontal shift of the input maps). As the FCN is translation-equivariant, the feature volume will be shifted horizontally. The number of columns of the feature volume defines the resolution of the yaw estimation, which is 1 degree in the case of our leg architecture.

2) Delta Head: The delta head is designed to estimate the overlap between two scans. It consists of a delta layer, three convolutional layers, and one fully connected layer.

The delta layer, shown in Fig. 4, computes all possible absolute differences of all pixels. It takes the output feature volumes $\mathbf{L}^l \in \mathbb{R}^{H\times W\times C}$ from the two legs l as input. These are stacked in a tiled tensor $\mathbf{T}^l \in \mathbb{R}^{HW\times HW\times C}$ as follows:

$$\mathbf{T}^0(iW + j, k, c) = \mathbf{L}^0(i, j, c) \tag{4}$$

$$\mathbf{T}^1(k, iW + j, c) = \mathbf{L}^1(i, j, c), \tag{5}$$

with $k = \{0, \ldots, HW - 1\}$, $i = \{0, \ldots, H - 1\}$ and $j = \{0, \ldots, W - 1\}$.

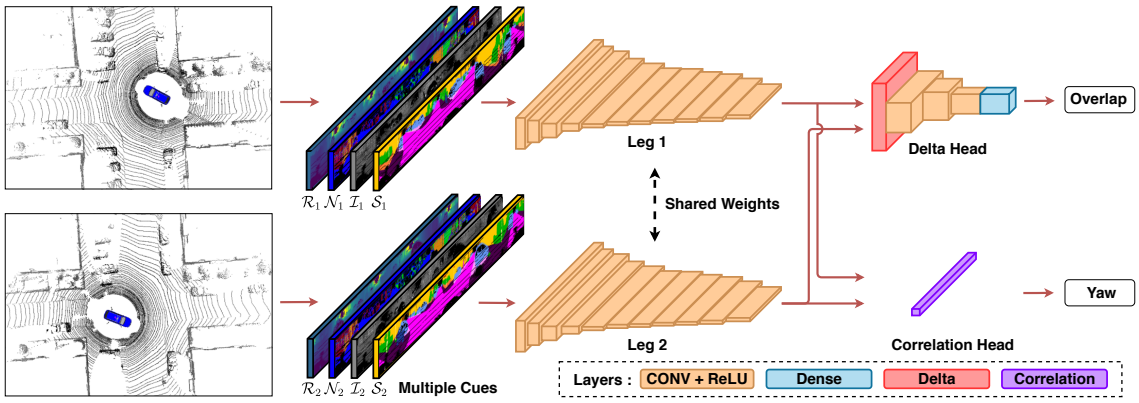


Fig. 3: Pipeline overview of our proposed approach. The left-hand side shows the preprocessing of the input data which exploits multiple cues generated from a single LiDAR scan, including range $\mathcal{R}$, normal $\mathcal{N}$, intensity $\mathcal{I}$, and semantic class probability $\mathcal{S}$ information. The right-hand side shows the proposed OverlapNet which consists of two legs sharing weights and the two heads use the same pair of feature volumes generated by the two legs. The outputs are the overlap and relative yaw angle between two LiDAR scans.

TABLE I: Layers of our network architecture

	Operator	Stride	Filters	Size	Output Shape
Legs	Conv2D	(2, 2)	16	(5, 15)	$30 \times 443 \times 16$
	Conv2D	(2, 1)	32	(3, 15)	$14 \times 429 \times 32$
	Conv2D	(2, 1)	64	(3, 15)	$6 \times 415 \times 64$
	Conv2D	(2, 1)	64	(3, 12)	$2 \times 404 \times 64$
	Conv2D	(2, 1)	128	(2, 9)	$1 \times 396 \times 128$
	Conv2D	(1, 1)	128	(1, 9)	$1 \times 388 \times 128$
	Conv2D	(1, 1)	128	(1, 9)	$1 \times 380 \times 128$
	Conv2D	(1, 1)	128	(1, 9)	$1 \times 372 \times 128$
	Conv2D	(1, 1)	128	(1, 7)	$1 \times 366 \times 128$
	Conv2D	(1, 1)	128	(1, 5)	$1 \times 362 \times 128$
	Conv2D	(1, 1)	128	(1, 3)	$1 \times 360 \times 128$
Delta Head	Conv2D	(1, 15)	64	(1, 15)	$360 \times 24 \times 64$
	Conv2D	(15, 1)	128	(15, 1)	$24 \times 24 \times 128$
	Conv2D	(1, 1)	256	(3, 3)	$22 \times 22 \times 256$
	Dense	-	-	-	1

Note that $\mathbf{T}^1$ is transposed in respect to $\mathbf{T}^0$, as depicted in the middle of Fig. 4. After that, all differences are calculated by element-wise absolute differences between $\mathbf{T}^0$ and $\mathbf{T}^1$.

By using the delta layer, we can obtain a representation of the latent difference information, which can be later exploited by the convolutional and fully-connected layers to estimate the overlap. Different overlaps induce different patterns in the output of the delta layer.

3) Correlation Head: The correlation head [24] is designed to estimate the yaw angle between two scans using the feature volumes of the two legs. To perform the cross-correlation, we first pad horizontally one feature volume by copying the same values (as the range images are cyclic projections around the yaw angle). This doubles the size of the feature volume. We then use the other feature volume as a kernel that is shifted over the first feature volume generating a 1D output of size 360. The argmax of this feature serves as the estimate of the relative yaw angle of the two input scans with a 1 degree resolution.

D. Loss Functions

We train our OverlapNet end-to-end to estimate the overlap and the relative yaw angle between two LiDAR scans at the same time. Typically, to train a neural network one needs a large amount of manually labeled ground truth data. In our case, this is $(I_1,\ I_2,\ Y_O,\ Y_Y)$, where I_1, I_2 are two inputs and Y_O, Y_Y are the ground truth overlaps and the ground truth yaw angles respectively. We are however able to generate the input and the ground truth without any manual effort in a fully automated fashion given a dataset with pose information. From given poses, we can calculate the ground truth overlap and relative yaw angles directly. We denote the legs part network with trainable weights as $f_L(\cdot)$, the delta head as $f_D(\cdot)$ and the correlation head as $f_C(\cdot)$.

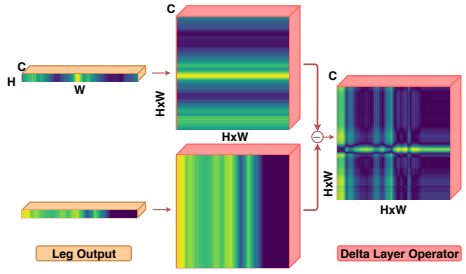


Fig. 4: Delta layer. Computation of pairwise differences is efficiently performed by concatenating the feature volumes and transposition of one concatenated feature volume.

For training, we combine the loss $L_O(\cdot)$ for the overlap and the loss $L_Y(\cdot)$ for the yaw angle using a weight α:

$$L\left(I_1, I_2, Y_O, Y_Y\right) = L_O\left(I_1, I_2, Y_O\right) + \alpha L_Y\left(I_1, I_2, Y_Y\right). \quad (6)$$

We treat the overlap estimation as a regression problem and use a weighted absolute difference of ground truth Y_O and network output $\hat{Y}_O = f_D\left(f_L\left(I_1\right), f_L\left(I_2\right)\right)$ as the loss function. For weighting, we use a scaled sigmoid function:

$$L_O\left(I_1, I_2, Y_O\right) = \text{sigmoid}\left(s\left(\left|\hat{Y}_O - Y_O\right| + a\right) - b\right), \quad (7)$$

with $\text{sigmoid}(v) = (1 + \exp(-v))^{-1}$, a, b are offsets and s being a scaling factor.

For the yaw angle estimation, we use a lightweight representation of the correlation head output, which leads to a

one-dimensional vector of size 360. We take the index of the maximum, the argmax, as the estimate of the relative angle in degrees. As the argmax is not differentiable, we cannot treat this as a simple regression problem. The yaw angle estimation, however, can be regarded as a binary classification problem that decides for every entry of the head output whether it is the correct angle or not. Therefore, we use the binary cross-entropy loss given by

$$L_Y\left(I_1, I_2, Y_Y\right) = \sum\nolimits_{i=\{1,\ldots,N\}} \mathrm{H}\left(Y_Y^i, \hat{Y}_Y^i\right), \tag{8}$$

where $\mathrm{H}(p,q) = p\log(q) - (1-p)\log(1-q)$ is the binary cross entropy and N is the size of the output 1D vector. $\hat{Y}_Y = f_C\left(f_L\left(I_1\right), f_L\left(I_2\right)\right)$ is the relative yaw angle estimate. Note that we only train the network to estimate the relative yaw angle of a pair of scans with overlap larger than 30%, since this minimum overlap was needed to result in correct pose estimates of the ICP as explained in Sec. IV-A, but also experimentally validated in Sec. IV-F.

E. SLAM Pipeline

We use the surfel-based mapping system called SuMa [3] as our SLAM pipeline and integrate OverlapNet in SuMa replacing its original heuristic loop closure detection method. We only summarize here the steps of SuMa relevant to our approach and refer for more details to the original paper [3].

SuMa uses the same vertex map $\mathcal{V}_D$ and normal map $\mathcal{N}_D$ as discussed in Sec. III-B. Furthermore, SuMa uses projective ICP with respect to a rendered map view $\mathcal{V}_M$ and $\mathcal{N}_M$ at timestep $t-1$, the pose update $\mathbf{T}_{C_{t-1}C_t}$ and consequently $\mathbf{T}_{WC_t}$ by chaining all pose increments. Therefore, each vertex $\mathbf{u} \in \mathcal{V}_D$ is projectively associated to a reference vertex $\mathbf{v_u} \in \mathcal{V}_M$. Given this association information, SuMa estimates the transformation between scans by incrementally minimizing the point-to-plane error given by

$$E(\mathcal{V}_D, \mathcal{V}_M, \mathcal{N}_M) = \sum_{\mathbf{u}\in\mathcal{V}_D} \left(\mathbf{n}_u^\top \left(\mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} - \mathbf{v}_u \right) \right)^2. \tag{9}$$

Each vertex $\mathbf{u} \in \mathcal{V}_D$ is projectively associated to a reference vertex $\mathbf{v}_u \in \mathcal{V}_M$ and its normal $\mathbf{n}_u \in \mathcal{N}_M$ via

$$\mathbf{v}_u = \mathcal{V}_M \left(\Pi \left(\mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} \right) \right) \tag{10}$$

$$\mathbf{n}_u = \mathcal{N}_M \left(\Pi \left(\mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} \right) \right). \tag{11}$$

SuMa then minimizes the objective of Eq. (9) using Gauss-Newton and determines increments δ by iteratively solving

$$\delta = \left(\mathbf{J}_\delta^\top \mathbf{W} \mathbf{J}_\delta \right)^{-1} \mathbf{J}_\delta^\top \mathbf{W} \mathbf{r}, \tag{12}$$

where $\mathbf{W} \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing weights w_u, $\mathbf{r} \in \mathbb{R}^n$ is the stacked residual vector, and $\mathbf{J}_\delta \in \mathbb{R}^{n\times 6}$ the Jacobian of $\mathbf{r}$ with respect to the increment δ.

SuMa employs a loop closure detection module, which considers the nearest frame in the built map as the candidate for loop closure given the current pose estimate. Loop closure detection works well for small loops, but the heuristic fails in areas with only a few large loops. Furthermore, drift in the odometry estimate can lead to large displacements, where the heuristic of just taking the nearest frame in the already mapped areas does not yield correct candidates, which will be also shown in our experiments.

F. Covariance Propagation for Geometric Verification

SuMa's loop closure detection uses a fixed search radius. In contrast, we use the covariance of the pose estimate and error propagation to automatically adjust the search radius.

We assume a noisy pose $\mathbf{T}_{C_{t-1}C_t} = \{\bar{\mathbf{T}}_{C_{t-1}C_t}, \Sigma_{C_{t-1}C_t}\}$ with mean $\bar{\mathbf{T}}_{C_{t-1}C_t}$ and covariance $\Sigma_{C_{t-1}C_t}$. We can estimate the covariance matrix by

$$\Sigma_{C_{t-1}C_t} = \frac{1}{K} \frac{E}{N-M} \left(\mathbf{J}_\delta^\top \mathbf{W} \mathbf{J}_\delta \right)^{-1}, \tag{13}$$

where K is the correction factors of the Huber robustized covariance estimation [17], E is the sum of the squared point-to-plane errors (sum of squared residuals) given the pose $\mathbf{T}_{C_{t-1}C_t}$, see Eq. (9), N is the number of correspondences, $M = 6$ is the dimension of the transformation between two 3D poses.

To estimate the propagated uncertainty during the incrementally pose estimation, we can update the mean and covariance as follows:

$$\bar{\mathbf{T}}_{C_{t-1}C_{t+1}} = \bar{\mathbf{T}}_{C_{t-1}C_t} \bar{\mathbf{T}}_{C_tC_{t+1}} \tag{14}$$

$$\Sigma_{C_{t-1}C_{t+1}} \approx \Sigma_{C_{t-1}C_t} + \mathbf{J}_{C_tC_{t+1}}^\top \Sigma_{C_tC_{t+1}} \mathbf{J}_{C_tC_{t+1}}, \tag{15}$$

where $\mathbf{J}_{C_tC_{t+1}}$ is the Jacobian of $\mathbf{T}_{C_tC_{t+1}}$.

Since we need the Mahalanobis distance D_M as a probabilistic distance measure between two poses, we make use of Lie algebra to express $\mathbf{T}$ as a 6D vector $\xi \in \mathfrak{se}(3)$ using $\xi = \log \mathbf{T}$, yielding

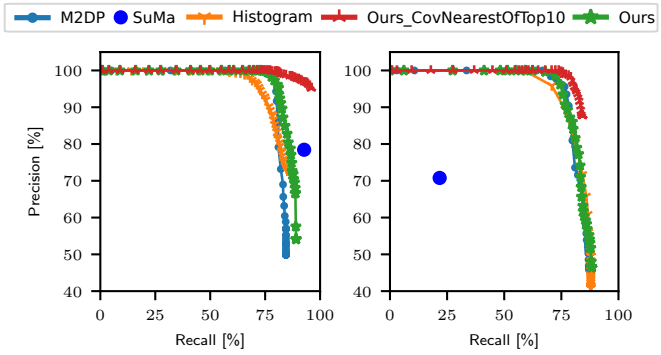
$$D_M\left(\mathbf{T}_{C1}, \mathbf{T}_{C2}\right) = \sqrt{\Delta\xi_{C1C2}^\top \Sigma_{C1C2}^{-1} \Delta\xi_{C1C2}}. \tag{16}$$

Using the scaled distance, we can now restrict the search space depending on the pose uncertainty to save computation time. However, we can use our framework also without any prior information, i.e., perform place recognition.

IV. Experimental Evaluation

The experimental evaluation is designed to support the key claims that our approach is able to: (i) predict the overlap and relative yaw angle between pairs of LiDAR scans by exploiting multiple cues without given poses, (ii) combine odometry information with overlap predictions to detect correct loop closure candidates, (iii) improve the overall pose estimation results in graph-based SLAM yielding more globally consistent maps, (iv) solve loop closure detection without prior pose information, (v) initialize ICP using OverlapNet predictions yielding correct scan matching results.

We train and evaluate our approach on the KITTI odometry benchmark [13], which provides LiDAR scans recorded with a Velodyne HDL-64E of urban areas around Karlsruhe in Germany. We follow the experimental setup of Schaupp *et al.* [27] and use sequence 00 for evaluation. Sequences $03-10$ are used for training and sequence 02 is used for validation.



(a) KITTI sequence 00 (b) Ford campus sequence 00

Fig. 5: Precision-Recall curves of different approaches.

To evaluate the generalization ability of our method, we also test it on the Ford campus dataset [25], which is recorded on the Ford research campus and downtown Dearborn in Michigan using a different version of the Velodyne HDL-64E. In the case of the Ford campus dataset, we test our method on sequence 00 which has several large loops. Note that we never trained our approach on the Ford campus dataset.

For generating overlap ground truth, we only use points within a distance of 75 m to the sensor. For overlap computation, see Eq. (3), we use $\epsilon = 1\,\text{m}$. We use a learning rate of 10^{-3} with a decay of 0.99 every epoch and train at most 100 epochs. For the combined loss, Eq. (6), we set $\alpha = 5$. For the overlap loss, Eq. (7), we use $a = 0.25, b = 12$, and scale factor $s = 24$.

A. Loop Closure Detection

In our first experiments, we investigate the loop closure performance of our approach and compare it to existing methods. Loop closure detection typically assumes that robots revisit places during the mapping while moving with uncertain odometry. Therefore, the prior information about robot poses extracted from the pose graph is available for the loop closure detection. The following criteria are used in these experiments:

- To avoid detecting a loop closure in the most recent scans, we do not search candidates in the latest 100 scans.
- For each query scan, only the best candidate is considered throughout this evaluation.
- Most SLAM systems search for potential closures only within the 3σ area around the current pose estimate. We do the same, either using the Euclidean or the Mahalanobis distance, depending on the approach.
- We use a relatively low threshold of 30 % for the overlap to decide if a candidate is a true positive. We aim to find more loops even in some challenging situations with low overlaps, e.g., when the car drives back to an intersection from the opposite direction (as highlighted in the supplementary video[1]). Furthermore, ICP can find correct poses if the overlap between pairs of scans is around 30 %, as illustrated in the experimental evaluation.

[1]https://youtu.be/YTfliBco6aw

We evaluate OverlapNet on both KITTI sequence 00 and Ford campus sequence 00 using the precision-recall curves shown in Fig. 5. We compare our method, trained with two heads and all cues (labeled as *Ours (AllChannel, TwoHeads)*) with three state-of-the-art approaches, M2DP [15], Histogram [26], and the original SuMa [3]. Since SuMa always uses the nearest frame as the candidate for loop closure detection, we can only get one pair of precision and recall value resulting in a single point. We also show the result of our method using prior information, named *Ours_CovNearestOfTop10*, which uses covariance propagation (Sec. III-F) to define the search space with the Mahalanobis distance and use the nearest in Mahalanobis distance of the top 10 predictions of OverlapNet as the loop closure candidates.

Tab. II shows the comparison between our approach and the state of the art using the F1 score and the area under the curve (AUC) on both KITTI and Ford campus dataset. For the KITTI dataset, our approach uses the model trained with all cues, including depth, normals, intensity, and a probability distribution over semantic classes. For the Ford campus dataset, our approach uses the model trained with geometric information only, namely *Ours (GeoOnly)*, since other cues are not available in this dataset. We can see that our method outperforms the other methods on the KITTI dataset and attains a similar performance on the Ford campus dataset. There are two reasons to explain the worse performance on the Ford campus dataset. First, we never trained our network on the Ford campus dataset or even US roads, and secondly, there is only geometric information available on the Ford campus dataset. However, our method outperforms all baseline methods in both, KITTI and Ford campus dataset, if we integrate prior information.

We also show the performance in comparison to variants of our method in Tab. III. We compare our best model *AllChannel* using two heads and all available cues to a variant which only uses a basic multilayer perceptron as the head named *MLPOnly* which consists of two hidden fully connected layers and a final fully connected layer with two neurons (one for overlap, one for yaw angle). The substantial difference of the AUC and F1 scores shows that such a simple network structure is not sufficient to get a good result. Training the network with only one head (only the delta head for overlap estimation, named *DeltaOnly*), has not a significant influence on the performance. A huge gain can be observed when regarding the nearest frame in Mahalanobis distance of the top 10 candidates in overlap percentage (*CovNearestOfTop10*).

B. Qualitative Results

The second experiment is designed to support the claim that our method is able to improve the overall mapping result. Fig. 6 shows the odometry results on KITTI sequence 02. The color in Fig. 6 shows the 3D translation error (including height). The left figure shows the SuMa and the right figure shows *Ours_CovNearestOfTop10* using the proposed OverlapNet to detect loop closures. We can see that after integrating our method, the overall odometry is much more

TABLE II: Comparison with state of the art.

Dataset	Approach	AUC	F1 score
KITTI	Histogram [26]	0.83	0.83
	M2DP [15]	0.83	0.87
	SuMa [3]	-	0.85
	Ours (AllChannel, TwoHeads)	**0.87**	**0.88**
Ford Campus	Histogram [26]	0.84	0.83
	M2DP [15]	0.84	**0.85**
	SuMa [3]	-	0.33
	Ours (GeoOnly)	**0.85**	0.84

TABLE III: Comparison with our variants.

Dataset	Variant	AUC	F1 score
KITTI	MLPOnly	0.58	0.65
	DeltaOnly	0.85	0.88
	CovNearestOfTop10	**0.96**	**0.96**
	Ours (AllChannel, TwoHeads)	0.87	0.88
Ford Campus	Ours (GeoOnly)	0.85	0.84
	GeoCovNearestOfTop10	**0.85**	**0.88**

accurate since we can provide more loop closure candidates with higher accuracy in terms of overlap. The colors represent the translation error of the estimated poses with respect to the ground truth. Furthermore, after integrating the proposed OverlapNet, the SLAM system can find more loops even in some challenging situations, e.g., when the car drives back to an intersection from the opposite direction, which is highlighted in the supplementary video[1].

C. Loop Closure Detection without Odometry Information

The third experiment is designed to support the claim that our approach is well-suited for solving the more general loop closure detection task without using odometry information.

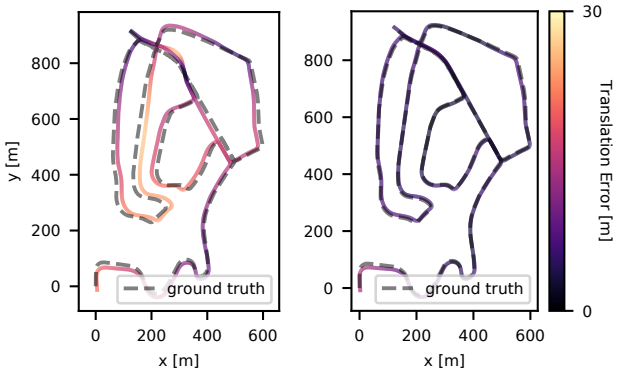
In this case, we assume that we have no prior information about the robot pose. To compare with the state-of-the-art method OREOS [27], we follow their experimental setup and refer to the original paper for more details. The OREOS results are those produced by the authors of OREOS.

The respective loop closure candidates recall results are shown in Fig. 7. Our method outperforms all the baseline methods with a small number of candidates and attains similar performance as baseline methods for higher values of numbers of candidates. However, OREOS and LocNet++ attain a slightly higher recall if more candidates are considered.

D. Yaw Estimation

We aim at supporting our claim that our network provides good relative yaw angle estimates. We use the same experimental setup as described in Sec. IV-C. Tab. IV summarizes the yaw angle errors on KITTI sequence 00.

We can see that our method outperforms the other methods in terms of mean error and standard deviations. In terms of recall, OverlapNet and OREOS always provide a yaw angle estimate, since both approaches are designed to estimate the



(a) SuMa (b) Ours_CovNearestOfTop10

Fig. 6: Qualitative result on KITTI sequence 02.

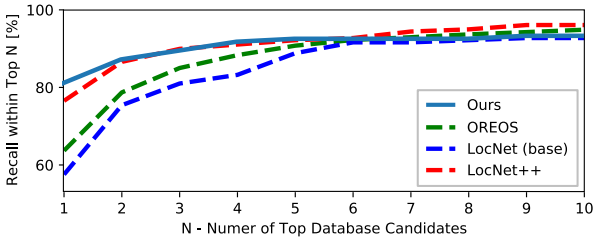


Fig. 7: Loop closure detection performance on KITTI sequence 00.

relative yaw angle for any pairs of scans in contrast to the RANSAC-based method that sometimes fails.

The superior performance can be mainly attributed to the correlation head exploiting the fact that the orientation in LiDAR scans can be well represented by the shift in the range projection. Therefore, it is easier to train the correlation head to accurately predict the relative yaw angles rather than a multilayer perceptron used in OREOS [27]. Furthermore, there is also a strong relationship between overlap and yaw angle, which also improves the results when trained together.

Fig. 8 shows the relationship between real overlap and yaw angle estimation error. As expected, the yaw angle estimate gets better with increasing overlap. Based on these plots, our method not only finds candidates but also measures the quality, i.e., when the overlap of two scans is larger than 90%, our method can accurately estimate the relative yaw angle with an average error of about only 1 degree.

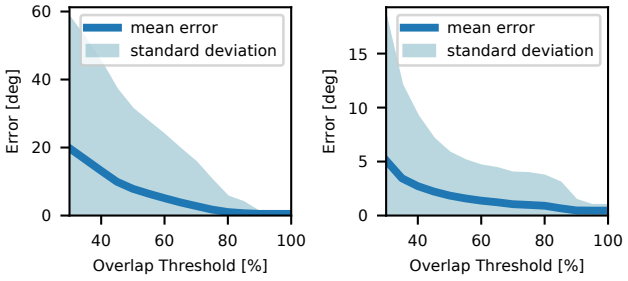
E. Ablation Study on Input Modalities

An ablation study on the usage of different inputs is shown in Tab. V. As can be seen, when employing more input modalities, the proposed method is more robust. We notice that exploiting only depth information with OverlapNet can already perform reasonable in terms of overlap prediction, while it does not perform well in yaw angle estimation. When combining with normal information, the OverlapNet can perform well in both tasks. Another interesting finding is the drastic reduction of yaw angle mean error and standard deviation when using semantic information. One reason could

TABLE IV: Yaw estimation errors without ICP

Approach	Mean[deg]	std[deg]	Recall[%]
FPFH+RANSAC*	13.28	32.19	97
OREOS*	12.67	15.23	100
Ours (AllChannel, TwoHeads)	**1.13**	**3.34**	**100**

*: The results are those produced by the authors of OREOS [27].



(a) KITTI sequence 00 (b) Ford campus sequence 00

Fig. 8: Overlap and yaw estimation relationship.

be that adding semantic information will make the input data more distinguishable when the car drives in symmetrical environments. We also notice that semantic information will increase the computation time, see Sec. IV-G. However, from the ablation study, one could also notice that the proposed method can also achieve good performance by only employing geometric information (depth and normals).

F. Using OverlapNet Predictions as Initial Guesses for ICP

We aim at supporting the claim that our network provides good initializations for ICP with 3D laser scans collected on autonomous cars. Fig. 9 shows the relation between the overlap and ICP registration error with and without using OverlapNet predictions as initial guesses. The error of ICP registration is here depicted by the Euclidean distance between the estimated relative translation and the ground-truth translation. As can be seen, the yaw angle prediction of the OverlapNet increases the chance to get a good result from the ICP even if two frames are relatively far away from each other (with low overlap). Therefore in some challenging cases, e.g. the car drives back into an intersection from a different street, our approach can still find loop closures (see in the supplementary video[1]). The results also show that the overlap estimates measure the quality of the found loop closure: larger overlap values result in better registration results of the involved ICP.

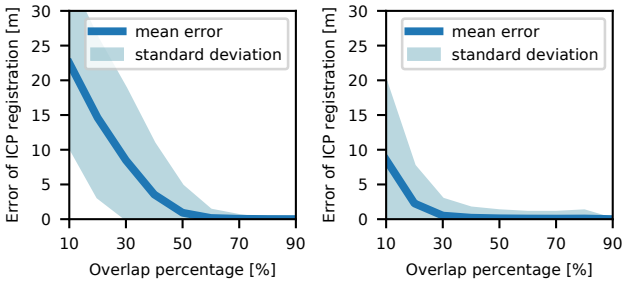
G. Runtime

We tested our method on a system equipped with an Intel i7-8700 with 3.2 GHz and an Nvidia GeForce GTX1080 Ti with 11 GB memory.

For the KITTI sequence 00, we could exploit all input cues including the semantic classes provided by RangeNet++ [23]. We need on average 75 ms per frame for the input data preprocessing, 6 ms per frame for the legs feature extraction, 27 ms per frame for the head matching. The worst case for the head matching takes 630 ms for all candidates in the search space.

TABLE V: Ablation study on usage of input modalities.

Depth	Normals	Intensity	Semantics	overlap AUC	overlap F1	yaw angle[deg] Mean	yaw angle[deg] Std
✓				0.86	0.87	11.67	25.32
✓	✓			0.86	0.85	2.97	14.28
✓	✓	✓		0.87	0.87	2.53	14.56
✓	✓	✓	✓	0.87	0.88	1.13	3.34



(a) With identity as initial guess (b) With yaw angle as initial guess

Fig. 9: ICP using OverlapNet predictions as initial guess. The error of ICP registration here is the Euclidean distance between the estimated translation and the ground-truth translation.

For the Ford campus dataset, we used only geometric information, which could be generated in 10 ms on average per frame, 2 ms for feature extraction and 24 ms for matching with the worst case of 550 ms. In real SLAM operation, we only search loop closure candidates inside a certain search space given by pose uncertainty using the Mahalanobis distance (see Sec. III-F). Therefore, our method can achieve online operation in long-term tasks, since we usually only have to evaluate a small number of candidate poses.

V. Conclusion

In this paper, we presented a novel approach for LiDAR-based loop closure detection. It is based on the overlap between LiDAR scan range images and provides a measure for the quality of the loop closure. Our approach utilizes a siamese network structure to leverage multiple cues and allows us to estimate the overlap and relative yaw angle between scans. The experiments on two different datasets suggest that when combined with odometry information our method outperforms other state-of-the-art methods and that it generalizes well to different environments never seen during training.

Despite these encouraging results, there are several avenues for future research. First, we want to investigate the integration of other input modalities, such as vision and radar information. We furthermore plan to test our approach with other datasets collected in different seasons.

Acknowledgments

This work has been supported in part by the German Research Foundation (DFG) under Germany's Excellence Strategy, EXC-2070 - 390732324 (PhenoRob) and under grant number BE 5996/1-1 as well as by the Chinese Scholarship Committee.

References

[1] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part I. *IEEE Robotics and Automation Magazine (RAM)*, 13(2):99–110, 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.1638022.

[2] Ioan Andrei Barsan, Shenlong Wang, Andrei Pokrovsky, and Raquel Urtasun. Learning to Localize Using a LiDAR Intensity Map. In *Proc. of the Second Conference on Robot Learning (CoRL)*, pages 605–616, 2018.

[3] Jens Behley and Cyrill Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[4] Paul J. Besl and Neil D. McKay. A Method for Registration of 3D Shapes. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 14(2): 239–256, 1992.

[5] Igor Bogoslavskyi and Cyrill Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[6] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature Verification using a "Siamese" Time Delayed Neural Network. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, 07(04):669–688, 1993. doi: 10.1142/S0218001493000339.

[7] Andrea Censi and Stefano Carpin. HSM3D: feature-less global 6DOF scan-matching in the Hough/Radon domain. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3899–3906. IEEE, 2009.

[8] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguère, Jens Behley, and Cyrill Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[9] Konrad P. Cop, Paulo V.K. Borges, and Renaud Dubé. Delight: An efficient descriptor for global localisation using lidar intensities. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[10] Andrei Cramariuc, Renaud Dubé, Hannes Sommer, Roland Siegwart, and Igor Gilitschenski. Learning 3D Segment Descriptors for Place Recognition. *arXiv preprint*, 2018.

[11] Mark Cummins and Paul Newman. Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[12] Renaud Dubé, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart, and Cesar Cadena. SegMatch: Segment Based Place Recognition in 3D Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[14] Jiadong Guo, Paulo V.K. Borges, Chanoh Park, and Abel Gawel. Local descriptor for robust place recognition using LiDAR intensity. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1470–1477, 2019.

[15] Li He, Xiaolong Wang, and Hong Zhang. M2DP: A Novel 3D Point Cloud Descriptor and Its Application in Loop Closure Detection. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[16] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-Time Loop Closure in 2D LIDAR SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[17] Peter J. Huber. *Robust Statistics*. Wiley, 1981.

[18] Mushtaq Hussain and James Bethel. Project and mission planing. In Chris McGlone, Edward Mikhail, James Bethel, and Roy Mullen, editors, *Manual of Photogrammetry*, chapter 15.1.2.6, pages 1109–1111. American Society for Photogrammetry and Remote Sensing, 2004.

[19] Giseop Kim, Byungjae Park, and Ayoung Kim. 1-day learning, 1-year localization: Long-term LiDAR localization using scan context image. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1948–1955, 2019.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):8490, May 2017. ISSN 0001-0782. doi: 10.1145/3065386.

[21] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Trans. on Robotics (TRO)*, 32(1):1–19, 2016. ISSN 1552-3098. doi: 10.1109/TRO.2015.2496823.

[22] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[23] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[24] Sei Nagashima, Koichi Ito, Takafumi Aoki, Hideaki Ishii, and Koji Kobayashi. A high-accuracy rotation estimation algorithm based on 1D phase-only correlation. In *Proc. of the Intl. Conf. on Image Analysis and Recognition*, pages 210–221, 2007.

[25] Gaurav Pandey, James R. McBride, and Ryan M. Eustice. Ford campus vision and lidar data set. *Intl. Journal of Robotics Research (IJRR)*, 30(13):1543–1552, 2011.

[26] Timo Röhling, Jennifer Mack, and Dirk Schulz. A Fast Histogram-Based Similarity Measure for Detecting Loop Closures in 3-D LIDAR Data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*,

pages 736–741, 2015.

[27] Lukas Schaupp, Mathias Bürki, Renaud Dubé, Roland Siegwart, and Cesar Cadena. OREOS: Oriented Recognition of 3D Point Clouds in Outdoor Scenarios. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[28] Cyrill Stachniss, Dirk Hähnel, Wolfram Burgard, and Giorgio Grisetti. On Actively Closing Loops in Grid-based FastSLAM. *Advanced Robotics*, 19(10):1059–1080, 2005.

[29] Cyrill Stachniss, John J. Leonard, and Sebastian Thrun. *Springer Handbook of Robotics, 2nd edition*, chapter Chapt. 46: Simultaneous Localization and Mapping. Springer Verlag, 2016.

[30] Basitan Steder, Radu B. Rusu, Kurt Konolige, and Wolfram Burgard. NARF: 3D range image features for object recognition. In *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[31] Bastian Steder, Michael Ruhnke, Slawomir Grzonka, and Wolfram Burgard. Place Recognition in 3D Scans Using a Combination of Bag of Words and Point Feature Based Relative Pose Estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[32] Li Sun, Daniel Adolfsson, Martin Magnusson, Henrik Andreasson, Ingmar Posner, and Tom Duckett. Localising Faster: Efficient and precise lidar-based robot localisation in large-scale environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[33] Mikaela A. Uy and Gimm H. Lee. PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4470–4479, 2018.

[34] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and Certifiable Point Cloud Registration. *arXiv preprint*, 2020.

[35] Huan Yin, Yue Wang, Xiaqing Ding, Li Tang, Shoudong Huang, and Rong Xiong. 3D LiDAR-Based Global Localization Using Siamese Neural Network. *IEEE Trans. on Intelligent Transportation Systems (TITS)*, 2019.

[36] Anestis Zaganidis, Alexandros Zerntev, Tom Duckett, and Grzegorz Cielniak. Semantically Assisted Loop Closure in SLAM Using NDT Histograms. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[37] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 766–782, 2016.

[M] X. Chen, T. Läbe, L. Nardi, *et al.*, "Learning an Overlap-based Observation Model for 3D LiDAR Localization," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020. DOI: 10.1109/IROS45743.2020.9340769

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Learning an Overlap-based Observation Model for 3D LiDAR Localization

Xieyuanli Chen Thomas Läbe Lorenzo Nardi Jens Behley Cyrill Stachniss

***Abstract*— Localization is a crucial capability for mobile robots and autonomous cars. In this paper, we address learning an observation model for Monte-Carlo localization using 3D LiDAR data. We propose a novel, neural network-based observation model that computes the expected overlap of two 3D LiDAR scans. The model predicts the overlap and yaw angle offset between the current sensor reading and virtual frames generated from a pre-built map. We integrate this observation model into a Monte-Carlo localization framework and tested it on urban datasets collected with a car in different seasons. The experiments presented in this paper illustrate that our method can reliably localize a vehicle in typical urban environments. We furthermore provide comparisons to a beam-endpoint and a histogram-based method indicating a superior global localization performance of our method with fewer particles.**

I. Introduction

All mobile systems that navigate autonomously in a goal-directed manner need to know their position and orientation in the environment, typically with respect to a map. This task is often referred to as localization and can be challenging especially at the city-scale and in the presence of a lot of dynamic objects, e.g., vehicles and humans. Over the last decades, a wide range of localization systems have been developed relying on different sensors. Frequently used sensors are GPS receivers, inertial measurement units, cameras, and laser range scanners. All sensing modalities have their advantages and disadvantages. For example, GPS does not work indoors, cameras do not work well at night or under strong appearance changes, and LiDARs are active sensors and are still rather expensive.

Most autonomous robots as well as cars have a 3D LiDAR sensor onboard to perceive the scene and directly provide 3D data. In this paper, we consider the problem of vehicle localization *only* based on a 3D LiDAR sensor. For localization, probabilistic state estimation techniques such as extended Kalman filters (EKF) [9], particle filters [13], or incremental factor graph optimization approaches [24] can be found in most localization systems today. Whereas EKFs and most optimization-based approaches track only a single mode, particle filters inherently support multi-modal distributions. Furthermore, PFs do not restrict the motion or observation model to follow a specific distribution such as a Gaussian. The motion model can often be defined quite easily based on the physics of vehicle motion. The observation model, however, is trickier to define. Often, these hand-designed models are used and they strongly impact the performance of the resulting localization system. Frequently used observation models for LiDARs are the beam-end point model, also called the likelihood field [27], the ray-casting model [10], or models based on handcrafted features [26], [35]. Recently, researchers also focused on learning such models completely from data [1], [30].

In this paper, we address the problem of learning observation models for 3D LiDARs and propose a deep neural network-based approach for that. We explore the possibilities of learning an observation model based on OverlapNet [4] that predicts the overlap between two LiDAR scans, see Fig. 1, where the overlap is defined as the ratio of points that can be seen from both LIDAR scans. In this work, we investigate this concept for 3D LiDAR-based observation models.

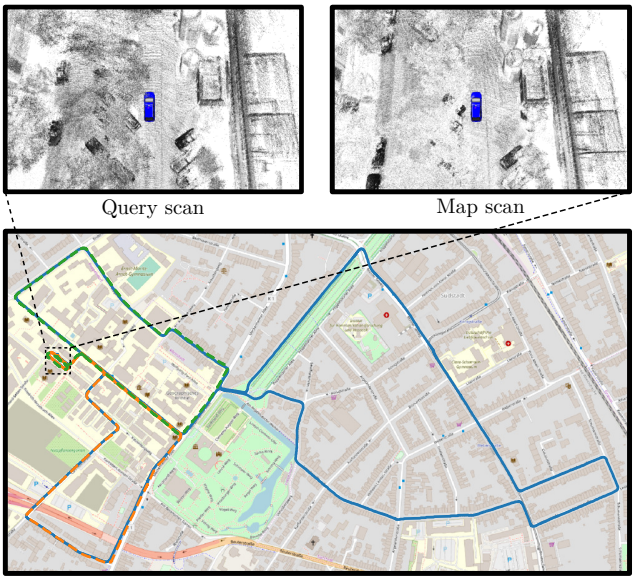


Fig. 1. The image in the lower part shows the trajectories of the dataset used in this paper, overlayed on OpenStreetMap. The blue trajectory represents the sequence used to generate a map for localization. The green and the orange trajectories represent two different test sequences. The 3D point clouds shown in the upper part show LiDAR scans of the same place, once during mapping and once during localization. Since the LiDAR data was collected in different seasons, the appearance of the environment changed quite significantly due to changes in the vegetation but also due to parked vehicles at different places.

All authors are with the University of Bonn, Germany.

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob as well as under grant number BE 5996/1-1 and by the Chinese Scholarship Committee.

The main contribution of this paper is a novel observation model for 3D LiDAR-based localization. Our model is learned from range data using a deep neural network. It estimates the overlap and yaw angle offset between a query frame and map data. We use this information as the observation model in Monte-Carlo localization (MCL) for updating the importance weights of the particles. Based on our novel observation model, our approach achieves online localization using 3D LiDAR scans over extended periods of time with a comparably small number of particles.

The source code of our approach is available at:
`https://github.com/PRBonn/overlap_localization`

II. Related Work

Localization is a classical topic in robotics [27]. For localization given a map, one often distinguishes between pose tracking and global localization. In pose tracking, the vehicle starts from a known pose and the pose is updated over time. In global localization, no pose prior is available. In this work, we address global localization using 3D laser scanners without assuming any pose prior from GPS or other sensors. Thus, we focus mainly on LiDAR-based approaches in this section.

Traditional approaches to robot localization rely on probabilistic state estimation techniques. These approaches can still be found today in several localization systems [19], [7]. A popular framework is Monte-Carlo localization [10], [28], [12], which uses a particle filter to estimate the pose of the robot.

In the context of autonomous cars, there are many approaches building and using high-definition (HD) maps for localization, i.e., tackling the simultaneous localization and mapping problem [25] and additionally adding relevant information for the driving domain. Levinson *et al*. [17] utilize GPS, IMU, and LiDAR data to build HD maps for localization. They generate a 2D surface image of ground reflectivity in the infrared spectrum and define an observation model that uses these intensities. The uncertainty in intensity values has been handled by building a prior map [18], [32]. Barsan *et al*. [1] use a fully convolutional neural network (CNN) to perform online-to-map matching for improving the robustness to dynamic objects and eliminating the need for LiDAR intensity calibration. Their approach shows a strong performance but requires a good GPS prior for operation. Based on this approach, Wei *et al*. [30] proposed a learning-based compression method for HD maps. Merfels and Stachniss [20] present an efficient chain graph-like pose-graph for vehicle localization exploiting graph optimization techniques and different sensing modalities. Based on this work, Wilbers et al. [31] propose a LiDAR-based localization system performing a combination of local data association between laser scans and HD map features, temporal data association smoothing, and a map matching approach for robustification.

Other approaches aim at performing LiDAR-based place recognition to initialize localization. For example, Kim *et al*. [16] transform point clouds into scan context images and train a CNN based on such images. They generate scan context images for both the current frame and all grid cells of the map and compare them to estimate the current location as the cell presenting the largest score. Yin *et al*. [34] propose a Siamese network to first generate fingerprints for LiDAR-based place recognition and then use iterative closest points to estimate the metric poses. Cop *et al*. [8] propose a descriptor for LiDAR scans based on intensity information. Using this descriptor, they first perform place recognition to find a coarse location of the robot, eliminate inconsistent matches using RANSAC, and then refine the estimated transformation using iterative closest points. In contrast to approaches that perform place recognition first, our approach integrates a neural network-based observation model into an MCL framework to estimate the robot pose.

Recently, several approaches exploiting semantic information for 3D LiDAR localization have been proposed. In our earlier work [6], we used a camera and a LiDAR to detect victims and localize the robot in an urban search and rescue environment. Ma *et al*. [19] combine semantic information such as lanes and traffic signs in a Bayesian filtering framework to achieve accurate and robust localization within sparse HD maps. Yan *et al*. [33] exploit buildings and intersections information from a LiDAR-based semantic segmentation system [21] to localize on OpenStreetMap. Schaefer *et al*. [23] detect and extract pole landmarks from 3D LiDAR scans for long-term urban vehicle localization. Tinchev *et al*. [29] propose a learning-based method to match segments of trees and localize in both urban and natural environments. Dubé *et al*. [11] propose to perform localization by extracting segments from 3D point clouds and matching them to accumulated data. Whereas, Zhang *et al*. [35] utilize both ground reflectivity features and vertical features for localizing autonomous car in rainy conditions. In our previous work [5], we also exploit semantic information [21], [2] to improve the localization and mapping results by detecting and removing dynamic objects.

Different to the above discussed methods [1], [19], [30], which use GPS as prior for localization, our method only exploits LiDAR information to achieve global localization without using any GPS information. Moreover, our approach uses range scans without explicitly exploiting semantics or extracting landmarks. Instead, we rely on CNNs to predict the overlap between range scans and their yaw angle offset and use this information as an observation model for Monte-Carlo localization. The localization approach proposed in this paper is based on our previous work called OverlapNet [4], which focuses on loop-closing for 3D LiDAR-based SLAM.

III. Our Approach

The key idea of our approach is to exploit the neural network, OverlapNet [4] that can estimate the overlap and yaw angle offset of two scans for building an observation model for localization in a given map. To this end, we first generate virtual scans at 2D locations on a grid rendered from the aggregated point cloud of the map. We train the network completely self-supervised on the map used for

localization (see Sec. III-A). We compute features using our pre-trained network that allows us to compute the overlap and yaw angles between a query and virtual scans (see Sec. III-B). Finally, we integrate an observation model using the overlap (see Sec. III-D) and a separate observation model for the yaw angle estimates (see Sec. III-E) in a particle filter to perform localization (see Sec. III-C).

A. OverlapNet

We proposed the so-called OverlapNet to detect loop closures candidates for a 3D LiDAR-based SLAM. The idea of overlap has its origin in the photogrammetry and computer vision community [15]. The intuition is that to successfully match two images and calculate their relative pose, the images must overlap. This can be quantified by defining the overlap percentage as the percentage of pixels in the first image, which can successfully be projected back into the second image.

In OverlapNet, we use the idea of overlap for range images generated from 3D LiDAR scans exploiting the range information explicitly. First, we generate from the LiDAR scans a range-image like input tensor $\mathbf{I} \in \mathbb{R}^{H \times W \times 4}$, where each pixel (i, j) in the range-image like representation corresponds to the depth and the corresponding normal, i.e., $\mathbf{I}(i, j) = (\mathbf{r}, \mathbf{n}_x, \mathbf{n}_y, \mathbf{n}_z)$, where $\mathbf{r} = ||\mathbf{p}||_2$. The indices for inserting points and estimated normals are computed using a spherical projection that maps points $\mathbf{p} \in \mathbb{R}^3$ to two-dimensional coordinates (i, j), also see [4], [5].

OverlapNet uses a siamese network structure and takes the tensors $\mathbf{I}_1$ and $\mathbf{I}_2$ of two scans as input for the legs to generate feature volumes, $\mathbf{F}_1$ and $\mathbf{F}_2$. These two feature volumes are then used as inputs for two heads. One head $\mathcal{H}_{\text{overlap}}(\mathbf{F}_1, \mathbf{F}_2)$ estimates the overlap percentage of the scans and the other head $\mathcal{H}_{\text{yaw}}(\mathbf{F}_1, \mathbf{F}_2)$ estimates the relative yaw angle.

For training, we determine ground truth overlap and yaw angle values using known poses estimated by a SLAM approach [3]. Given these poses, we can train OverlapNet completely self-supervised. For more details on the network structure and the training procedure, we are referring to our prior work [4].

B. Map of Virtual Scans

OverlapNet requires two LiDAR scans as input. One is the current scan and the second one has to be generated from the map. Thus, we build a map of virtual LiDAR scans given an aggregated point cloud by using a grid of locations with grid resolution γ, where we generate virtual LiDAR scans for each location. The grid resolution is a trade-off between the accuracy and storage size of the map. Instead of storing these virtual scans, we just need to use one leg of the OverlapNet to obtain a feature volume $\mathbf{F}$ using the input tensor $\mathbf{I}$ of this virtual scan. Storing the feature volume instead of the complete scan has two key advantages: (1) it uses more than an order less space than the original point cloud (ours: 0.1 Gb/km, raw scans: 1.7 Gb/km) and (2) we do not need to compute the $\mathbf{F}$ during localization on the map. The features volumes of the virtual scans are then used to compute overlap and yaw angle estimates with a query scan that is the currently observed LiDAR point cloud in our localization framework.

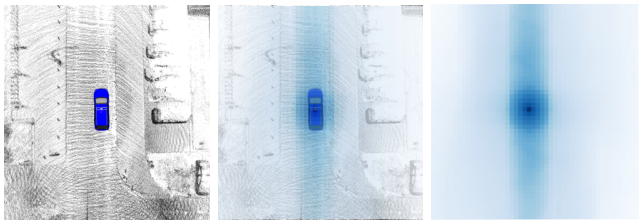


Fig. 2. Overlap observation model. Local heatmap of the scan at the car's position with respect to the map. Darker blue shades correspond to higher probabilities.

C. Monte-Carlo Localization

Monte-Carlo localization (MCL) is a localization algorithm based on the particle filter proposed by Dellaert *et al*. [10]. Each particle represents a hypothesis for the robot's or autonomous vehicle's 2D pose $\mathbf{x}_t = (x, y, \theta)_t$ at time t. When the robot moves, the pose of each particle is updated with a prediction based on a motion model with the control input $\mathbf{u}_t$. The expected observation from the predicted pose of each particle is then compared to the actual observation $\mathbf{z}_t$ acquired by the robot to update the particle's weight based on an observation model. Particles are resampled according to their weight distribution and resampling is triggered whenever the effective number of particles, see for example [14], drops below 50% of the sample size. After several iterations of this procedure, the particles will converge around the true pose.

MCL realizes a recursive Bayesian filtering scheme. The key idea of this approach is to maintain a probability density $p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ of the pose $\mathbf{x_t}$ at time t given all observations $\mathbf{z}_{1:t}$ up to time t and motion control inputs $\mathbf{u}_{1:t}$ up to time t. This posterior is updated as follows:

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \; p(\mathbf{z}_t \mid \mathbf{x}_t) \cdot \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}) \; p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \; d\mathbf{x}_{t-1}, \quad (1)$$

where η is a normalization constant, $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$ is the motion model, and $p(\mathbf{z}_t \mid \mathbf{x}_t)$ is the observation model.

This paper focuses on the observation model. For the motion model, we follow a standard odometry model for vehicles [27].

We split the observation model into two parts:

$$p(\mathbf{z}_t \mid \mathbf{x}_t) = p_L\left(\mathbf{z}_t \mid \mathbf{x}_t\right) \; p_O\left(\mathbf{z}_t \mid \mathbf{x}_t\right), \quad (2)$$

where $\mathbf{z}_t$ is the LiDAR observation at time t, $p_L\left(\mathbf{z}_t \mid \mathbf{x}_t\right)$ is the probability encoding the location (x, y) agreement between the current query LiDAR scan and the virtual scan at the grid where the particle locate and $p_O\left(\mathbf{z}_t \mid \mathbf{x}_t\right)$ is the probability encoding the yaw angle θ agreement between the same pairs of scans.

D. Overlap Observation Model

Given a particle i with the state vector (x_i, y_i, θ_i), the overlap estimates encode the location agreement between the query LiDAR scan and virtual scans of the grid cells where particles locate. It can be directly used as the probability:

$$p_L\left(\mathbf{z}_t \mid \mathbf{x}_t\right) \propto f\left(\mathbf{z}_t, \mathbf{z}_i; \mathbf{w}\right), \tag{3}$$

where f corresponds to the neural network providing the overlap estimation between the input scans $\mathbf{z}_t, \mathbf{z}_i$ and $\mathbf{w}$ is the pre-trained weights of the network. $\mathbf{z}_t$ and $\mathbf{z}_i$ are the current query scan and a virtual scan of one (x, y) location respectively. Note that no additional hyperparameter is needed to formulate our observation model for localization.

For illustration purposes, Fig. 2 shows the probabilities of all grid cells in a local area calculated by the overlap observation model. The blue car in the figure shows the current location of the car. The probabilities calculated by the overlap observation model can represent well the hypotheses of the current location of the car.

Typically, a large number of particles are used, especially when the environment is large. However, a large amount of particles will increase the computation time linearly. When applying the overlap observation model, particles could still obtain relatively large weights as long as they are close to the actual pose, even if not in the exact same position. This allows us to use fewer particles to achieve a high success rate of global localization.

Furthermore, the overlap estimation only encodes the location hypotheses. Therefore, if multiple particles locate in the same grid area, only a single inference against the nearest virtual scan of the map needs to be done, which can further reduce the computation time.

E. Yaw Angle Observation Model

Given a particle i with the state vector (x_i, y_i, θ_i), the yaw angle estimates encode the orientation agreement between the query LiDAR scan and virtual scans of the corresponding grids where particles locate. We formulate the orientation probability as follows:

$$p_O\left(\mathbf{z}_t \mid \mathbf{x}_t\right) \propto \exp\left(-\frac{1}{2}\frac{\left(g\left(\mathbf{z}_t, \mathbf{z}_i; \mathbf{w}\right) - \theta_i\right)^2}{\sigma_\theta^2}\right), \tag{4}$$

where g corresponds to the neural network providing the yaw angle estimation between the input scans $\mathbf{z}_t, \mathbf{z}_i$ and $\mathbf{w}$ is the pre-trained weights of the network. $\mathbf{z}_t$ and $\mathbf{z}_i$ are the current query scan and a virtual scan of one particle respectively.

When generating the virtual scans of the grid map, all virtual scans will be set facing the absolute 0° yaw angle direction. By doing this, the estimated relative yaw angle between the query scan and the virtual scan indicates the absolute yaw angle of the current query scan. Eq. (4) assumes a Gaussian measurement error in the heading.

By combining overlap and yaw angle estimation, the proposed observation model will correct the weights of particles considering agreements between the query scan and the map with the full pose (x, y, θ).



Fig. 3. Sensor setup used for data recording: Ouster OS1-64 LiDAR sensor plus GNSS information from a Emilid Reach RS2.

IV. Experimental Evaluation

In this paper, we use a grid representation and generate virtual 360° scans for each grid point. The resolution of the grid is $\gamma = 20$ cm. When generating the virtual scans, we set the yaw angle to 0°. Therefore, when estimating the relative yaw angle between the query frame and the virtual frames, the result will indicate the absolute yaw of the query frame. For the yaw angle observation model in Eq. (4), we set $\sigma_\theta = 5^\circ$. To achieve global localization, we train a new model only based on the map scans and the generated virtual scans.

The main focus of this work is a new observation model for LiDAR-based localization. Therefore, when comparing different methods, we only change the observation model $p(\mathbf{z}_t|\mathbf{x}_t)$ of the MCL framework and keep the particle filter-based localization the same. The motion model is the typical odometry model [27].

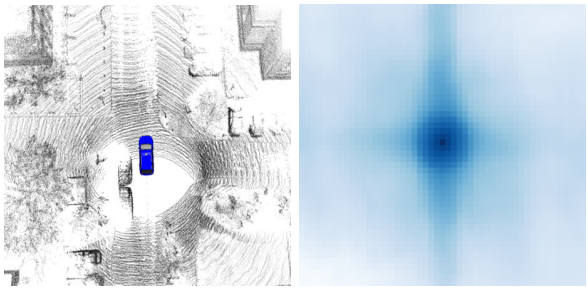
A. Car Dataset

The dataset used in this paper was collected using a self-developed sensor platform illustrated in Fig. 3. To test LiDAR-based global localization, a large-scale dataset has been collected in different seasons with multiple sequences repeatedly exploring the same crowded urban area. For our car dataset, we performed a 3D LiDAR SLAM [3] combined with GPS information to create near ground truth poses. During localization, we only use LiDAR scans for global localization without using GPS.

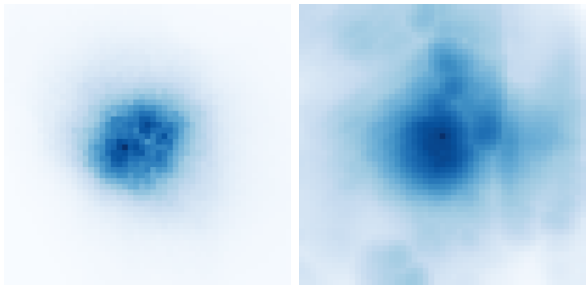
The dataset has three sequences that were collected at different times of the year, sequence 00 in September 2019, sequence 01 in November 2019, and sequence 02 in February 2020. The whole dataset covers a distance of over 10 km. We use LiDAR scans from sequence 02 to build the virtual scans and use sequence 00 and 01 for localization. As can be seen from Fig. 1, the appearance of the environment changes quite significantly since the dataset was collected in different seasons and in crowded urban environments, including changes in vegetation, but also cars at different locations and moving people.

B. Different Observation Models

In the following experiments, we use the same MCL framework and only exchange the observation models. We compare our observation model with two baseline observation models: the typical beam-end model [27] and a

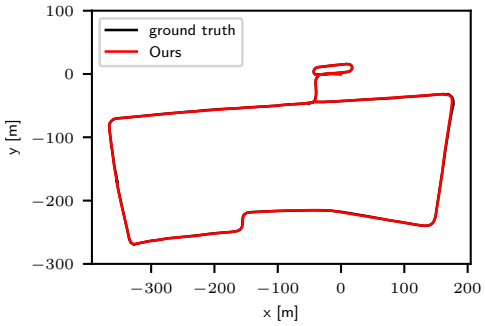


(a) Corresponding submap (b) Overlap-based $p(z|x)$

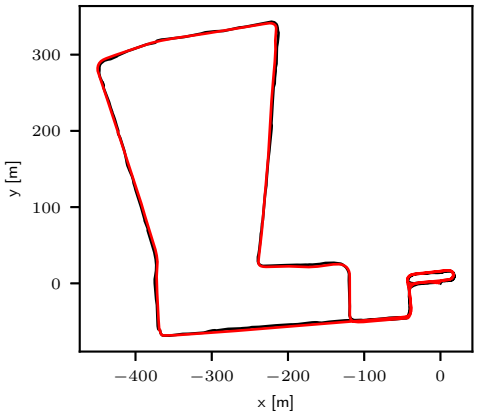


(c) Histogram-based $p(z|x)$ (d) Beam-end $p(z|x)$

Fig. 4. Heatmaps of different observation models used in the experiments generated for the same query scan and map.



(a) Trajectory from sequence 00



(b) Trajectory from sequence 01

Fig. 5. Localization results of our method with $10,000$ particles on two sequences recorded with the setup depicted in Fig. 3. Shown are the ground truth trajectory (black) and our estimated trajectory using our observation model (red).

histogram-based model derived from the work of Röhling *et al*. [22].

The beam-end observation model is often used for 2D LiDAR data. For 3D LiDAR scans, it needs much more particles to make sure that it converges to the correct pose, which causes the computation time to increase substantially. In this paper, we implement the beam-end model with a down-sampled point cloud map using voxelization with a resolution of 10 cm.

Our second baseline for comparison is inspired by Röhling *et al*. [22], which proposed a fast method to detect loop closures through the use of similarity measures on histograms extracted from 3D LiDAR data. The histogram contains the range information. We use a similar idea, but integrate it into the MCL framework as a baseline observation model. We employ the same grid map and virtual frames as used for our method with the histogram-based observation model. When updating the weights of particles, we will first generate the histograms of the current query scan and the virtual scans of grids where the particles locate. Then, we use the same Wasserstein distance to measure the similarity between them and update the weights of particles as follows:

$$p(\mathbf{z}_t|\mathbf{x}_t) \propto d\left(h(\mathbf{z}_t), h(\mathbf{z}_i)\right), \tag{5}$$

where d represents the Wasserstein distance between histograms $h(\mathbf{z}_t), h(\mathbf{z}_i)$ of LiDAR scan $\mathbf{z}_t, \mathbf{z}_i$.

Fig. 4 shows the 2D heatmaps in x and y calculated for the different observation models. As can be seen, the proposed observation model tends to give higher weights to the positions along the road, which leads to a higher success rate when the vehicle aims to localize in an urban environment. We will show the numerical results which verify that our method can achieve a high success rate with much fewer particles in the next sections.

C. Localization Performance

The experiment presented in this section is designed to show the performance of our approach and to support the claim that it is well suited for global localization.

First of all, we show the general localization results tested with two sequences in Fig. 5. The qualitative results show that, after applying our sensor-model, the proposed method can well localize in the map with only LiDAR data collected in highly dynamic environments at different times.

For quantitative results, we calculate the success rate for different methods with different particle numbers comparing our approach to different methods, as shown in Fig. 6. The x-axis represents the number of particles used during localization, while the y-axis is the success rate of different setups. The success rate for a specific setup of one method is calculated using the number of success cases divided by the total numbers of the tests. To decide whether one test is successful or not, we check the location error by every 100 frames after converging. If the location error is smaller than a certain threshold, we count this run as a success case.

TABLE I

LOCATION RESULTS

Sequence	Location error [meter]		
	Beam-end	Histogram-based	Ours
0	0.92 ± 0.27	1.85 ± 0.34	$\mathbf{0.81 \pm 0.13}$
1	$\mathbf{0.67 \pm 0.11}$	1.86 ± 0.34	0.88 ± 0.07

TABLE II

YAW ANGLE RESULTS

Sequence	Yaw angle error [degree]		
	Beam-end	Histogram-based	Ours
0	1.87 ± 0.47	3.10 ± 3.07	$\mathbf{1.74 \pm 0.11}$
1	2.10 ± 0.59	3.11 ± 3.08	$\mathbf{1.88 \pm 0.09}$

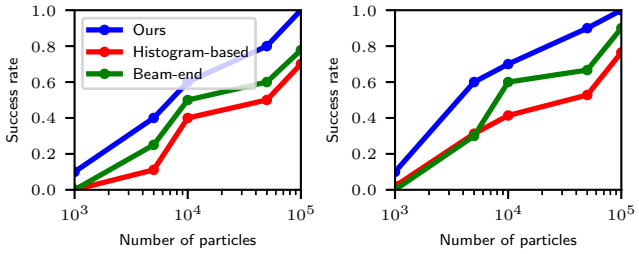
We test our method together with two baselines using five different numbers of particles $N = \{1,000\,;\, 5,000\,;\, 10,000\,;\, 50,000\,;\, 100,000\}$. For each setup, we sample 10 trajectories and perform global localization.

Quantitative results of localization accuracy are shown in Tab. I and Tab. II. Tab. I shows the location error of all methods tested with both sequences. The location error is defined as the root mean square error (RMSE) of each test in terms of (x, y) Euclidean error with respect to the ground truth poses. Tab. I shows the mean and the standard deviation of the error for each observation model. Note that the location error is only calculated for success cases.

Tab. II shows the yaw angle error. It is the RMSE of each test in terms of yaw angle error with respect to the ground truth poses. The table shows the mean and the standard deviation of the error for each observation model. As before, the yaw angle error is also only calculated for cases in which the global localization converged.

As can be seen from the results, our method achieves higher success rates with a smaller number of particles compared to the baseline methods, which also makes the proposed method faster than baseline methods. Furthermore, our method converges already with $100,000$ particles in all cases, whereas the other observation models still need more particles to sufficiently cover the state space. Moreover, the proposed method gets similar performance in location error comparing to the baseline methods but it achieves better results in yaw angle estimation. This is because the proposed method decouples the location and yaw angle estimation and, therefore, can exploit more constraint in yaw angle corrections.

To sum up, the proposed method outperforms the baselines method in terms of success rate, while getting similar results in terms of location error. Moreover, our method outperforms baseline methods in yaw angle estimation, because of the proposed de-coupled observation model. Furthermore, our method is faster than the baseline method. The runtime details will be shown in the next experiment.



(a) sequence 00 (b) sequence 01

Fig. 6. Success rate of the different observation models for 10 runs with $N = \{1,000\,;\, 5,000\,;\, 10,000\,;\, 50,000\,;\, 100,000\}$ particles. Here, we use sequence 00 and sequence 01 to localize in sequence 02. We count runs as success if they converge to the ground truth location within $5\,\mathrm{m}$.

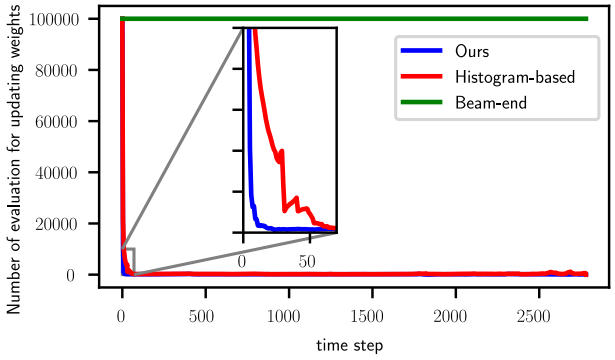


Fig. 7. Number of observation model evaluations for updating the weights at each timestep with $100,000$ particles. The beam end model needs to be evaluated for each and every particle individually. The histogram-based method is more computationally efficient, while our proposed method still needs the fewest evaluations.

D. Runtime

In this experiment, we show the *number* of observation model evaluations necessary for updating the weights at each time step in Fig. 7. This is a fairer way to compare the computational cost of different methods, since our neural network-based method uses GPU to concurrently updating the weights of particles, while the other methods only use CPU. As can be seen, our method needs a smaller number of observation model evaluations to update the weights for all particles. This is because we only need to perform the network inference for all particles which are localized in the same grid cell once. For one incoming frame and the virtual frame of that grid cell, the inputs of the network and thus the outputs are the same for all particles in that cell.

We tested our method on a system equipped with an Intel i7-8700 with 3.2 GHz and an Nvidia GeForce GTX 1080 Ti with 11 GB of memory. For initializing in the large-scale map, the worst case will take around $43\,\mathrm{s}$ to process one frame. However, after converging, the proposed method takes only $1\,\mathrm{s}$ on average to process one frame with $10,000$ particles.

V. Conclusion

In this paper, we presented a novel observation model and integrated it into an MCL framework to solve the global localization problem. Our method exploits OverlapNet to estimate the overlap and yaw angle between the current frame and the virtual frames generated at each particle using the pre-built map. This allows us to successfully update the weights of particles with agreements of both location and orientation. We implemented and evaluated our approach on an urban dataset collected with a car in different seasons and provided comparisons to other existing techniques. The experiments suggest that our approach can achieve a similar performance as other approaches in global localization while obtaining a higher success rate and lower computational time.

In future work, we will test our method on more datasets with different types of LiDAR sensors. We also plan to test our method with high definition maps and want to exploit semantic information.

References

[1] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun. Learning to Localize Using a LiDAR Intensity Map. In *Proc. of the Conference on Robot Learning (CoRL)*, 2018.

[2] 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 Intl. Conf. on Computer Vision (ICCV)*, 2019.

[3] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[4] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proc. of Robotics: Science and Systems (RSS)*, 2020.

[5] X. Chen, A. Milioto, E. Palazzolo, P. Gigure, J. Behley, and C. Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[6] X. Chen, H. Zhang, H. Lu, J. Xiao, Q. Qiu, and Y. Li. Robust slam system based on monocular vision and lidar for robotic urban search and rescue. In *Proc. of the IEEE Intl. Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 41–47, 2017.

[7] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. Ang, E. Frazzoli, and D. Rus. Synthetic 2D Lidar for Precise Vehicle Localization in 3D Urban Environment. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[8] K.P. Cop, P.V.K. Borges, and R. Dub. Delight: An efficient descriptor for global localisation using lidar intensities. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[9] I.J Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. on Robotics and Automation*, 7(2):193–204, 1991.

[10] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 1999.

[11] R. Dube, M.G. Gollub, H. Sommer, I. Gilitschenski, R. Siegwart, C.C. Lerma, and J. Nieto. Incremental segment-based localization in 3d point clouds. *IEEE Robotics and Automation Letters (RA-L)*, 2018.

[12] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, 1999(343-349), 1999.

[13] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In *Sequential Monte Carlo methods in practice*, pages 401–428. Springer, 2001.

[14] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Trans. on Robotics (TRO)*, 23(1):34–46, 2007.

[15] M. Hussain and J. Bethel. Project and mission planing. In C. McGlone, E. Mikhail, J. Bethel, and R. Mullen, editors, *Manual of Photogrammetry*, chapter 15.1.2.6, pages 1109–1111. American Society for Photogrammetry and Remote Sensing, 2004.

[16] G. Kim, B. Park, and A. Kim. 1-day learning, 1-year localization: Long-term lidar localization using scan context image. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1948–1955, 2019.

[17] J. Levinson, M. Montemerlo, and S. Thrun. Map-Based Precision Vehicle Localization in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[18] J. Levinson and S. Thrun. Robust Vehicle Localization in Urban Environments Using Probabilistic Maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 4372–4378, 2010.

[19] W. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mattyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky, and R. Urtasun. Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[20] Ch. Merfels and C. Stachniss. Pose fusion with chain pose graphs for automated driving. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[21] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[22] T. Röhling, J. Mack, and D. Schulz. A Fast Histogram-Based Similarity Measure for Detecting Loop Closures in 3-D LIDAR Data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 736–741, 2015.

[23] A. Schaefer, D. Büscher, J. Vertens, L. Luft, and W. Burgard. Long-term urban vehicle localization using pole landmarks extracted from 3-d lidar scans. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, pages 1–7, 2019.

[24] R.C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *Intl. Journal of Robotics Research (IJRR)*, 5(4):56–68, 1986.

[25] C. Stachniss, J. Leonard, and S. Thrun. *Springer Handbook of Robotics, 2nd edition*, chapter Chapt. 46: Simultaneous Localization and Mapping. Springer Verlag, 2016.

[26] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place Recognition in 3D Scans Using a Combination of Bag of Words and Point Feature Based Relative Pose Estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[27] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[28] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128(1-2), 2001.

[29] G. Tinchev, A. Penate-Sanchez, and M. Fallon. Learning to see the wood for the trees: Deep laser localization in urban and natural environments on a cpu. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1327–1334, 2019.

[30] X. Wei, I. A. Bârsan, S. Wang, J. Martinez, and R. Urtasun. Learning to Localize Through Compressed Binary Maps. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10316–10324, 2019.

[31] D. Wilbers, Ch. Merfels, and C. Stachniss. Localization with Sliding Window Factor Graphs on Third-Party Maps for Automated Driving. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[32] R. W. Wolcott and R. M. Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2814–2821, 2015.

[33] F. Yan, O. Vysotska, and C. Stachniss. Global Localization on OpenStreetMap Using 4-bit Semantic Descriptors. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2019.

[34] H. Yin, Y. Wang, X. Ding, L. Tang, S. Huang, and R. Xiong. 3d lidar-based global localization using siamese neural network. *IEEE Trans. on Intelligent Transportation Systems (ITS)*, 2019.

[35] C. Zhang, M. H. Ang, and D. Rus. Robust lidar localization for autonomous driving in rain. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3409–3415, 2018.

[N] A. Milioto, J. Behley, C. McCool, *et al.*, "LiDAR Panoptic Segmentation for Autonomous Driving," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020. DOI: 10.1109/IROS45743.2020.9340837

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

LiDAR Panoptic Segmentation for Autonomous Driving

Andres Milioto Jens Behley Chris McCool Cyrill Stachniss

***Abstract*— Truly autonomous driving without the need for human intervention can only be attained when self-driving cars fully understand their surroundings. Most of these vehicles rely on a suite of active and passive sensors. LiDAR sensors are a cornerstone in most of these hardware stacks, and leveraging them as a complement to other passive sensors such as RGB cameras is an enticing goal. Understanding the semantic class of each point in a LiDAR sweep is important, as well as knowing to which instance of that class it belongs to. To this end, we present a novel, single-stage, and real-time capable panoptic segmentation approach using a shared encoder with a semantic and instance decoder. We leverage the geometric information of the LiDAR scan to perform a novel, distance-aware tri-linear upsampling, which allows our approach to use larger output strides than using transpose convolutions leading to substantial savings in computation time. Our experimental evaluation and ablation studies for each module show that combining our geometric and semantic embeddings with our learned, variable instance thresholds, a category-specific loss, and the novel trilinear upsampling module leads to higher panoptic quality. We will release the code of our approach in our LiDAR processing library LiDAR-Bonnetal [27].**

I. Introduction

Perception and scene understanding are key components for building fully autonomous cars that can drive safely even in unknown parts of the world. A multitude of sensors such as cameras, LiDARs, and radars offering redundant views of the world are part of the hardware stack of these vehicles. Image-based perception has been steadily becoming more capable due to advances in deep learning [21] and convolutional neural networks (CNN). LiDAR sensors are often used because they produce accurate distance measurements, even in scenarios where other sensors fail, like at night. However, challenges caused by the characteristics of the LiDAR data, such as its distance-dependent sparsity, consequently call for different solutions.

Panoptic segmentation [16] is a recent task unifying semantic segmentation of so-called *stuff* classes and instance-specific *thing* classes jointly. Specifically, *stuff* refers to uncountable classes, such as *vegetation*, or *road*, but also countable classes that are not critical to distinguish individually when performing a specific task, such as is the case for *buildings* while driving. Opposite to this, *things* represent interesting countable classes for the task the robot performs, such as driving participants (i.e. *cars*, *pedestrians*, etc). Therefore, this task provides a unified understanding of the scene components, leading towards scene understanding capturing the complete picture.

All authors are with the University of Bonn, Germany. This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germanys Excellence Strategy EXC 2070 390732324, as well as grant number BE 5996/1-1.

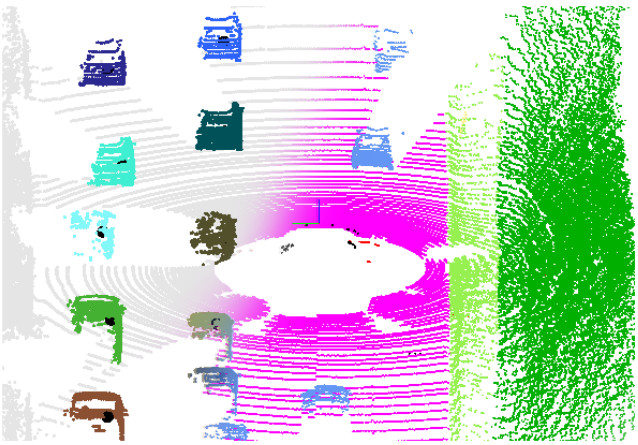


Fig. 1: Our approach provides a panoptic segmentation for point clouds from a rotating automotive LiDAR. Thus, we assign each point a semantic label (right part), and instance IDs (left part).

Recently, many approaches to this task using images were proposed [16], [22], [24], [31], [47]. In this paper, we investigate the task of panoptic segmentation using solely LiDAR scans and present an approach for solving this task, as shown in Fig. 1. We propose a single-stage approach that jointly solves semantic and instance segmentation. This architecture learns both instance and semantic embeddings using a shared encoder. The instance decoder provides an offset prediction for each point that points towards the object center, which allows us to segment individual instances. In turn, the semantic embeddings are used to extract point-wise semantic labels and aid the clustering of object centers. Furthermore, we propose a novel upsampling method that allows us to use large output strides, enabling better runtime performance. Combined with a category-based loss, we achieve high panoptic quality for the panoptic task [2] of SemanticKITTI [1] than an approach combining state-of-the-art projective semantic segmentation [27] and state-of-the-art object detection [20]. Lastly, our experiments show that our single-stage approach runs at a fraction of the time compared to two two-stage approaches, which is paramount for moving vehicles. In summary, our contributions are: (i) a single-stage approach for LiDAR panoptic segmentation that achieves state-of-the-art performance at a fraction of the processing time of two-stage approaches, (ii) a novel upsampling strategy exploiting the distance information provided by the LiDAR point clouds leading to better panoptic quality and increased runtime efficiency, and (iii) the novel combination of semantic and geometric embeddings with learned point-wise radii for metric learning-based instance clustering.

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

II. Related Work

We aim to provide a broad overview of closely related instance and semantic segmentation approaches for point clouds, but we also discuss closely related RGBD and image-based approaches.

Semantic Segmentation. For semantic segmentation of point clouds, a variety of approaches have been proposed. Voxel-based methods transform the point cloud into a voxel-grid and apply convolutional neural networks with 3D convolutions for object classification [25] and semantic segmentation [14]. Both approaches were among the first investigating such models and allow for directly exploiting architectures and insights known from image-based methods.

To overcome the limitations of the voxel-based representation, such as the inherent higher memory consumption with increasing voxel grid resolution, approaches either upsample voxel-predictions [39] using a conditional random field (CRF) or use a different representation, such as more efficient spatial subdivisions [9], [17], [37], [44], [50], graphs [19], [40], splats [38], or points directly [7], [10], [13], [15], [32], [34], [36], [41]. Opposite to these spatial partition approaches, methods exploiting the organization of the generated measurements by a rotating automotive LiDAR sensor [45], [46] or approaches using a cylindrical or spherical projection [27] of the point cloud showed promising results on the KITTI Vision Benchmark and its extension SemanticKITTI [1]. Compared to the aforementioned point cloud-based approaches, these techniques can use larger backbones [35] and realize a more efficient neighbor search by exploiting the organization of the data from the sensor directly [27].

We base our single-stage approach on the latter style, and we present a novel range-image-based tri-linear upsampling method in our decoders that exploits the image representation of neighbor information but uses the actual distances between points from the point cloud pyramid to upsample features spatially. Furthermore, we exploit a category loss that exploits the knowledge of a useful dataset ontology to improve the accuracy of the results.

Instance Segmentation. For image-based instance segmentation, there are mainly two types of approaches: detection-based [11], [12], [48] and clustering-based [5], [26], [29], [3]. Detection-based approaches, pioneered by Mask R-CNN [11], first locate objects using an object detector and then segment the object inside the bounding box. Clustering-based approaches, pioneered by Brabandere *et al*. [3], use metric learning to find an embedding, which facilitates the clustering of pixels from an instance. Often, this also involves the prediction of a seed pixel or point, like the center of an object [5], [26], [29], from which the clustering is seeded.

Also, point cloud instance segmentation has been explored. The approach of Wang *et al*. [42] extracts point-wise features using a PointNet, which are used to generate a similarity matrix, a confidence map, and a semantic segmentation then used to cluster instances by virtue of the similarity scores. The two-stage approach of Hou *et al*. [12] regresses bounding boxes for objects and uses then information from point clouds, but also image information to generate an instance mask for each bounding box. In contrast, the single-stage approach of Yang *et al*. [48] directly estimates a fixed number of bounding boxes and associates each bounding box with a point-wise mask separating the object from the background. Yi *et al*. [49] use object-like proposals instead of bounding boxes, which are then used to generate bounding boxes, segmentation masks, and classification into object classes.

Panoptic Segmentation. Recently, the task of panoptic segmentation [16], i.e., jointly predicting a semantic segmentation of *stuff* classes and instance segmentations for *things* gained significant interest using images [5] or RGBD data [12], [30], [43]. Panoptic segmentation metrics were also adopted by several of the major image datasets [6], [23], [28].

The approach of Pham *et al*. [30] uses a PointNet-based network to provide semantic class probabilities, but also instance embeddings. These are then used by a conditional random field [18] to predict instance labels and semantic labels for an RGBD scan. Similarly, Wang *et al*. [43] use an encoder with two decoders to generate semantic and instance features using PointNets for RGBD data, combining these with an associative segmentation module that uses semantics to generate instance IDs and vice versa.

In contrast to prior work, we propose a single-stage, end-to-end trainable, and real-time capable approach using point clouds generated by a rotating automotive LiDAR. Our approach combines a suite of practices that improve panoptic quality. This includes the combination of semantics and geometry for instance clustering, a learned point-wise threshold for said clustering, a new trilinear upsampling for range images in the decoders, and a joint class plus category loss.

III. Our Approach

Fig. 2 illustrates our network architecture. First, the point cloud obtained by the LiDAR scanner is projected to a range-image-like representation containing the range, (x, y, z) point coordinates, and remission of each point by virtue of a spherical projection of the de-skewed scans caused by the ego-motion of the vehicle. Then, we extract features at different resolutions, or output strides (OS), using a shared backbone, which is trained with all the losses through back-propagation. At the same time, we construct a point cloud pyramid which samples points from the image representation of the latter at exactly the location where the downsampling, stride 2 convolutions are applied in the backbone. This helps us recover the features with higher accuracy during the upsampling process.

Following the backbone and image pyramid, we use two separate decoders, bringing back the backbone features to the original image resolution, which also contain convolutional layers that learn task-specific functions. The first decoder extracts a semantic embedding $\hat{\mathbf{e}}_p$ that allows us to predict classes and categories, as well as an error estimate for the embeddings, used in the clustering. The second decoder

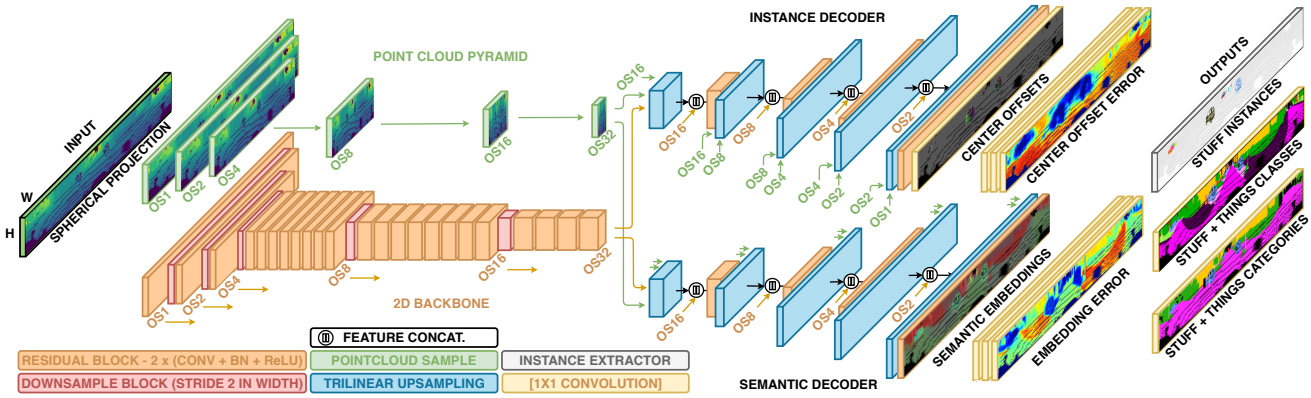


Fig. 2: Architecture layout for the single-stage, projective panoptic segmentation approach. For detailed samples of inputs, intermediate representations, and outputs see Fig. 3

extracts, for each point, an offset to the center of the instance. It also predicts an error estimate for the offsets, used later in the clustering (Sec. III-D). Both decoders use a novel range-image-based trilinear upsampling, which we explain in detail in Sec. III-C. Finally, the instance extractor uses the center offsets and the semantic embeddings to assign an instance id to each point in the *thing* classes and categories, before unprojecting the points to 3D. Fig. 3 shows an example of the input range image, the semantic embedding $\hat{\mathbf{e}}_p$ as a random projection from 32 dimensions to 3 displayed as RGB, the center offsets $\hat{\mathbf{o}}_p$ showing each one of the offsets in x, y, and z as RGB colors, and the final outputs based on semantic and geometric embeddings.

A. Projection

The first step in the projective panoptic segmentation pipeline is to project the points using a spherical projection (Fig. 3, top). To this end, we transform all (x, y, z) 3D points into a set of (u, v) 2D image coordinates using the formula:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\,\pi^{-1}\right] & W \\ \left[1 - \left(\arcsin(z\,r^{-1}) + \mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right] & H \end{pmatrix} \quad (1)$$

where $r = \sqrt{x^2 + y^2 + z^2}$ is the distance from the point to the sensor, and $f = f_{\mathrm{up}} + f_{\mathrm{down}}$ is the vertical sensor field of view. This generates a $(5, H, W)$ volume which represents the point cloud as an image with channels $(\mathrm{range}, x, y, z, \mathrm{remission})$. For a point cloud of size N, we generate an index matrix of size $(N, 2)$ containing all the (u, v) image coordinate pairs, which we use later when transferring back the predictions to 3D (see Sec. III-E).

B. Backbone and Point Cloud Pyramid

Relevant works for projection-based LiDAR semantic segmentation are SqueezeSeg [45], [46] and RangeNet++ [27], which is the first approach of this type for SemanticKITTI. These approaches exploit the way the sensor acquires the points using a rotating array of laser beams and use a 2D segmentation CNN on a spherical projection of the input point cloud.

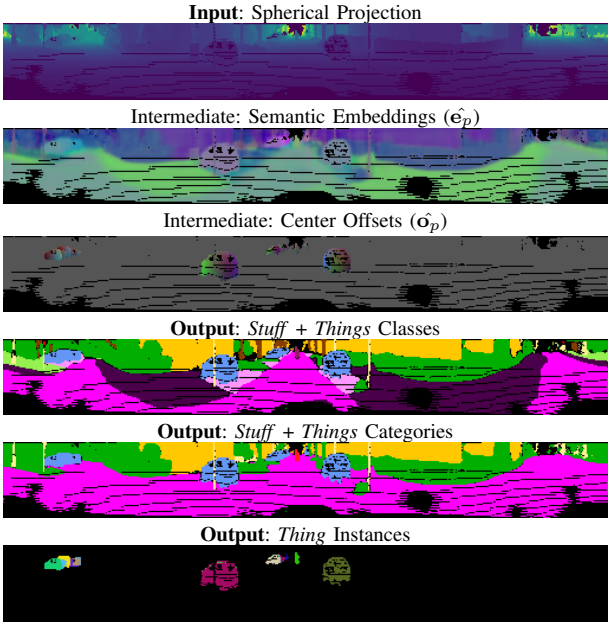


Fig. 3: Example input, intermediate results, and outputs of our panoptic segmentation.

Both of these approaches downsample periodically on the width dimension by using strided convolutions. This generates feature volumes of OS 1, 2, 4, 8, 16, and 32, which are later skipped to the decoder to recover high-frequency signals lost in downsampling, aiding the upsampling process, which uses either bilinear upsampling or transposed convolutions. However, neither of these approaches exploit the fact that the inputs contain useful metric information as they downsample, to aid the upsampling. In our work, each time we apply strided convolutions, we store a downsampled version of the point cloud which contains the points at the centers of the locations where the kernels were applied. This can be used to recover useful geometric information during the upsampling in the decoder that allows us to improve the accuracy of the final output. The foundation of our architecture and shared feature extractor for both the instance head and the semantic head branch is DarkNet53 [27], [35].

C. Decoders

After the point cloud range image is processed by the backbone and the sampled pyramid is generated, we pass the features through two different decoders that complement each other to solve the task of panoptic segmentation. One decoder predicts the instance centers, and the other one the semantics of the scene. Both decoders follow the same structure, which is illustrated in Fig. 2.

In contrast to prior works [27], [45], [46], which upsample the backbone features using transposed convolutions or bilinear upsampling, exploiting closeness in the image, we implement a differentiable trilinear upsampling layer, which exploits the fact that our inputs are 3D point clouds and not camera images. As in bilinear upsampling, for each point in the resolution that is currently upsampled, we find the 4 corresponding points in the coarser grid using the point cloud pyramid. However, since the real information of vicinity between the points and its coarser corresponding points is known to us through their real (x, y, z) coordinates, besides their image (u, v) coordinate. This allows us to approximate a trilinear upsampling by using the real 3D Euclidean distances rather than the 2D image ones. Even though this is technically not a strict trilinear upsampling due to the absence of a cubic lattice, our approach uses the real 3D distances. Thus, it more closely resembles the trilinear upsampling than bilinear interpolation, as shown in Fig. 4. After upsampling the feature volume, we concatenate the upsampled features with the matching resolution from the backbone as a skip connection followed by a convolutional block. These blocks also learn the task-dependent weights that separate the tasks. This is done recursively until the input resolution is met, and the task-dependent heads are applied. In the remainder, we describe each decoder head individually.

1) Instance decoder: This decoder has the responsibility to predict a 3D offset $\hat{\mathbf{o}}_p = [\Delta x, \Delta y, \Delta z]^T$ for each point $\mathbf{x}_p = [x_p, y_p, z_p]^T$ on the range image belonging to one of the *things*, from its center $\mathbf{c} = [x_c, y_c, z_c]^T$. This is similar to [29], which predicts a 2D offset to the center for instance segmentation, and [33], which predicts 3D offsets for point cloud object detection. Since we are using range image representations of 3D point clouds, our approach sits in the middle, predicting 3D offsets given an image representation. After the offsets are predicted, each point in the *thing* mask (from the semantic segmentation) predicts a center coordinate, which is used to cluster the instances. This is done using the region-based clustering method described in Sec. III-D.

The offsets $\hat{\mathbf{o}}_p$ are learnt through an L_2 loss of the form:

$$\mathcal{L}_{CENTER} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{P_i}\sum_{p=1}^{P_i}[\hat{\mathbf{o}}_p - (\mathbf{x}_p - \mathbf{c}_i)]^2, \quad (2)$$

where I is the number of instances in the batch, and P_i is the number of points in the instance i. During inference, the predicted center for each point can then be calculated as $\hat{\mathbf{c}}_p = \hat{\mathbf{x}}_p - \hat{\mathbf{o}}_p$.

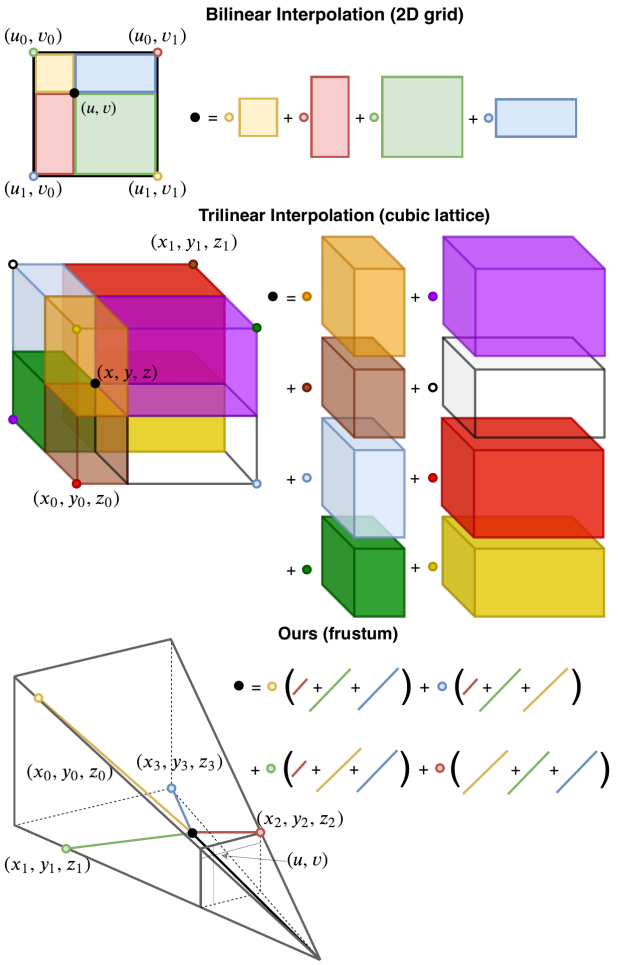


Fig. 4: Upsampling methods graphically. The black dot corresponds to the desired interpolated feature value, while the colored dots represent the values to interpolate from. Top: Bilinear upsampling. Center: Trilinear Interpolation. Bottom: Ours. The vertex values are obtained in the range image domain, but their distances to the query point are calculated in 3D as their real euclidean distance. Then the interpolated feature value is a linear combination of four the closest feature values in the lower resolution grid (which define the frustum), with the coefficients calculated as an inverse of the distance to the point, normalized by the total distance.

To use a region-based clustering method, an intra-cluster radius needs to be defined. For the center embedding $\hat{\mathbf{c}}_p$, the radius is the maximum Euclidean distance that we allow two points in the same instance to predict. In theory, this radius could be fixed, and chosen by cross-validation, but research shows that different points have different levels of accuracy [29], resulting in either over- or under-segmentation. Using a learned, adaptive radius for each point instead can help solve this problem, as shown in Fig. 5. Therefore, we add three convolutional blocks on top of the offset encoder that predict this radius $\hat{\epsilon}_p$ for each point, estimating the radius as the Euclidean distance between the offset prediction and its ground truth, for each point, during the training, i.e,

$$\mathcal{L}_{CENT\ ERR} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{P_i}\sum_{p=1}^{P_i}\left[\hat{\epsilon}_p - \|\hat{\mathbf{o}}_p - (\mathbf{x}_p - \mathbf{c}_i)\|\right]^2 \quad (3)$$

The intuition behind using the training offset error as the ground truth for this variable radii is not to estimate the uncertainty of each prediction, which would not be possible, since the training error is usually significantly lower than the test time error. Instead, we are interested in learning the *relative difficulty* of different points belonging to the same instance. Fig. 5 shows an example where the point in the fender is significantly more inaccurate than the point in the center of the car, requiring a larger radius to be properly assigned to this car's cluster. This is fixable by simply using a larger radius for all points, but this leads to under-segmentation in many cases in our dataset, preventing us from a proper understanding of the scene. Therefore, at inference time we use this learned tolerances as relative thresholds for each point, but adjust the overall scale by a constant factor to account for the difference between training and validation error. This factor is the ratio between the latter.

2) Semantic decoder: This decoder predicts a 32-dimensional semantic embedding for each point in the range image representation. From this embedding, two $[1 \times 1]$ convolutional heads are used to predict classes and categories. In order to train the network to predict both, we use a cross-entropy loss for the classes and an analogous one for the categories, of the form:

$$\mathcal{L}_{SEM} = -\sum_{c=1}^{C} w_c \, y_c \, \log\left(\hat{y}_c\right), \tag{4}$$

where $w_c = \frac{1}{\log\left(f_c + \epsilon\right)}$ is a class-wise weight calculated as a function of the inverse class-frequency f_c, and ϵ limits the largest possible weight for a class. In practice, the category prediction does not need its own head or loss. However, our experiments show that adding a separate head, along with a category loss, improves the category segmentation, effectively mapping semantically similar classes together in embedding space.

To aid the instance segmentation head center predictions, we also add an auxiliary loss that applies directly to the embeddings from the semantic head. Fig. 3 shows a random projection from the 32-dimensional embeddings to 3 dimensions, plotted as RGB values. It is possible to see that not only the embeddings are different from class to class, but also between different instances of the same class. This loss uses metric learning to cluster all pixel embeddings $\hat{\mathbf{e}}_p$ of the same instance close to each other, and pushing all mean embeddings of different instances away from each other. This is done through an attraction and a repulsion loss, of the form:

$$L_{ATTRACT} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{P_i}\sum_{p=1}^{P_i}\left(\hat{\mathbf{e}}_{i,p} - \hat{\bar{\mathbf{e}}}_i\right)^2 \tag{5}$$

$$L_{REPEL} = \frac{1}{I(I-1)}\sum_{\substack{i_A=1 \\ i_A \neq i_B}}^{I}\sum_{i_B=1}^{I}\frac{1}{\left(\hat{\bar{\mathbf{e}}}_{i_A} - \hat{\bar{\mathbf{e}}}_{i_B}\right)^2}, \tag{6}$$

where $\hat{\mathbf{e}}_{i,p}$ is the embedding for pixel p of instance i, and $\hat{\bar{\mathbf{e}}}_i$ is the mean embedding of all points in instance i.

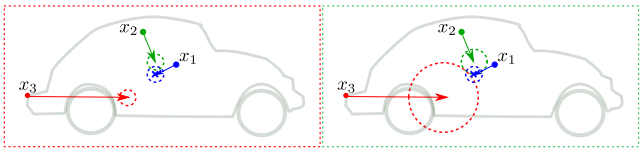


Fig. 5: Influence of clustering radius. Left: Wrong clustering due to fixed radii. Right: Correct clustering using the learned radii depending on point difficulty.

Analogously to the center offsets, we also predict an error radius for each pixel embedding that is later used for the adaptive clustering, following an analogous loss to the center offset error loss:

$$\mathcal{L}_{EMBED\ ERR} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{P_i}\sum_{p=1}^{P_i}\left[\hat{\epsilon}_p - \left\|\hat{\mathbf{e}}_{i,p} - \hat{\bar{\mathbf{e}}}_i\right\|\right]^2, \tag{7}$$

D. Instance Extractor

To obtain all individual instance IDs, we employ an iterative procedure. First, we sample a *thing* point from the semantic prediction and obtain its distance in feature space to all other points in the same class. We propose two alternative features to do this through the instance and the semantic decoders. Using the instance decoder, the features represent the prediction of the center of the instance $\hat{\mathbf{c}}_i = \mathbf{x}_p - \hat{\mathbf{o}}_p$. Using the semantic decoders, the features used are the embeddings of each point $\hat{\mathbf{e}}_{i,p}$. In both cases, we then use the predictions of the error for each pixel as the clustering radius to consider points as belonging to the same instance as the initially sampled point. We then assign the instance ID to all points within the radius, remove these points from the pool, and start over with a new sampled point, until all points are consumed. In our ablation study, we compare all features and the usage of the learned radius vs. a statically defined threshold chosen by cross-validation.

E. Point Cloud Extraction and Post-Processing

After the segmentation of the point cloud as a range image, recovering all labels for the N original points in the point cloud is desired. However, if $N > HW$, unprojecting the image using the intrinsic calibration of the sensor does not reconstruct all points. This is why the reprojection of the labeled points to the 3D world is, instead, performed by keeping an $(N, 2)$ shaped list of (u, v) image indexes that can use the label image as a lookup table to recover the labels and IDs for all N points.

IV. Experimental Evaluation

In the first part of the experiments, we compare our panoptic segmentation approach with state-of-the-art approaches on SemanticKITTI [1], [2], a large-scale LiDAR dataset providing instance and semantic segmentation annotations, as well as a panoptic segmentation benchmark. We then provide ablations studies to show the importance of different design decisions of our approach, such as the use of semantic and center embeddings jointly, the inclusion of the novel trilinear upsampling module, and the category loss.

Implementation details. In all the following experiments, we use the following parameters, when not otherwise stated. All networks in the single-stage approach were trained following the same training schedule, using Adam optimizer with a learning-rate of $1 \cdot 10^{-4}$, a warm-up ramp of 1 epoch, momentums $(0.9, 0.99)$, a learning-rate decay of 0.99 per epoch, training for 200 epochs. To integrate all losses, we tested GradNorm [4], but yielded no significant improvement over the simple addition of all losses.

Dataset. We evaluate our approach on SemanticKITTI [1], which provides point-wise semantic annotations for all scans of the KITTI odometry split [8]. Recently, we extended the dataset by providing temporally consistent instance annotations for all scans in the dataset [2]. The dataset provides 23,201 scans for training and the remaining 20,351 scans are used for evaluation on a benchmark server. We use sequence 08 from the training data comprised of 4,071 scans for validation purposes. The dataset contains overall 28 classes from which the vehicle classes and classes representing humans have point-wise instance annotations.

Evaluation Metrics. In order to compare the semantic segmentation branch with other approaches in the semantic segmentation benchmark, we calculate the mean intersection over union (mIoU) over all classes, defined as:

$$\text{mIoU} = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} \frac{|\{i \mid y_i = c\} \cap \{j \mid \hat{y}_j = c\}|}{|\{i \mid y_i = c\} \cup \{j \mid \hat{y}_j = c\}|}, \quad (8)$$

where y_i corresponds to the ground truth label of point $\vec{p}_i$ and $\hat{y}_i$ to the prediction.

To measure the quality of the joint semantic and instance segmentation for each class, we use the recently proposed panoptic quality (PQ) [16], [2]:

$$\text{PQ} = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} \frac{\sum_{(\mathcal{S},\hat{\mathcal{S}}) \in \text{TP}_c} \text{IoU}(\mathcal{S}, \hat{\mathcal{S}})}{|\text{TP}_c| + \frac{1}{2}|\text{FP}_c| + \frac{1}{2}|\text{FN}_c|}, \quad (9)$$

where $\text{IoU}(\mathcal{S}, \hat{\mathcal{S}})$ is the intersection over union between the prediction segment $\hat{\mathcal{S}}$ and the ground truth segment $\mathcal{S}$, TP_c are the pairs of predicted segments $\hat{\mathcal{S}}$ that present over 50% intersection over union (IoU) with a ground truth segment $\mathcal{S}$, FP_c the set of unmatched predicted segments $\hat{\mathcal{S}}$, and FN_c the set of unmatched ground truth segments $\mathcal{S}$. The overall panoptic quality (PQ) metric averaged over all classes, which makes the metric insensitive to class imbalance.

For *stuff* classes, Porzi *et al*. [31] use an alternative metric that considers the specific case of only one segment per image, where the segment- and IoU-based criterion often leads to unmatched predictions (FP):

$$\text{PQ}_c^{\dagger} = \begin{cases} \text{IoU}(\mathcal{S}, \hat{\mathcal{S}}) & \text{, if c is a } \textit{stuff} \text{ class} \\ \text{PQ}_c & \text{, otherwise.} \end{cases} \quad (10)$$

Lastly, SemanticKITTI defines an ontology assigning each class to a category, e.g., *truck*, *car*, *other-vehicle* belong to the category *vehicle* [2]. These category definitions are useful for autonomous driving, e.g. identifying *humans* as an alternative to the more fine-grained *person* or *bicyclist* classes. Therefore, alternatively to the class-wise metrics, we also evaluate all approaches with respect to categories.

A. Comparison with the State of the Art

The first experiment evaluates the performance of our single-stage approach in comparison with the two-stage approaches proposed as baselines for the panoptic task in [2].

Tab. I shows the results in terms of class-wise performance on the test set. Likewise, Tab. II shows the performance in respect to the categories. Our proposed single-stage approach gets superior performance in comparison with the two-stage approach using RangeNet++ [27]. However, it is worse than the best performing KPConv [41], which achieves 44.5 panoptic quality and can be mainly attributed to better semantic segmentation, albeit at a higher computational cost.

We also show in the results the difference between using the category loss and head, vs a lookup table between class prediction and corresponding category. Interestingly, these results show that the category loss helps to improve the panoptic quality performance for *thing* classes, but leads to worse results with respect to the semantic segmentation quality as shown by a drop in mIoU.

Nevertheless, the main motivation for our single-stage approach is the improved computational efficiency in comparison to the aforementioned two-stage approaches. Instead of using multiple different networks, we can use a single multi-task network, which also profits from sharing the encoder between different tasks. Fig. 6 shows the runtime performance in relation to the panoptic quality. Our single-stage approach is considerably faster than the two-stage approaches. Here, we assume that the separate object detectors runs in parallel (314 ms for *pedestrian/cyclist* and 105 ms for *car*) after the semantic segmentation (200 ms for KPConv and 95 ms for RangeNet++) resulting in 514 ms and 409 ms respectively. Our single-stage approach with trilinear upsampling takes 85 ms on average.

B. Ablation Studies

We also validate that our contributions lead to an increase in performance with respect to panoptic quality through ablation studies. Note that here we evaluate all approaches on the validation set.

The first experiment, cf. Tab. III, shows the influence of the upsampling method to regain spatial resolution in the decoder after backbone downsampling. We can see that using our trilinear evaluation leads to considerable gains in class and computational performance, allowing for more downsampling without sacrificing in accuracy or speed.

The next experiment, cf. Tab. IV, shows the influence of features used for clustering instances. We can see that the best method using a single head is through the prediction of the center instances, rather than the semantic embeddings. However, combining both yields an increase in the performance of the clustering, which is appreciated by an increase in the recognition quality of *things*. Furthermore, we compare the clustering using each feature with the learned radii vs the best static threshold, found by cross-validation, and we show that learning a point-wise radius helps the performance of the approach.

Method	FPS	mIoU	PQ	PQ†	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}
KPConv [41] + PointPillars [20]	1.9	58.8	44.5	52.5	54.4	80.0	32.7	38.7	81.5	53.1	65.9	79.0
RangeNet++ [27] + PointPillars [20]	2.4	52.4	37.1	45.9	47.0	75.9	20.2	25.2	75.2	49.3	62.8	76.5
Ours (**without** category loss)	11.8	51.0	35.3	44.3	45.0	76.5	19.1	24.1	76.7	47.2	60.2	76.4
Ours (**with** category loss)	11.8	50.9	38.0	47.0	48.2	76.5	25.6	31.8	76.8	47.1	60.1	76.2

TABLE I: Comparison of test set results on SemanticKITTI using *stuff*(St) and *thing*(Th) *classes*. All results in [%].

Method	FPS	mIoU	PQ	PQ†	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}
KPConv [41] + PointPillars [20]	1.9	84.8	70.0	71.5	79.2	87.3	54.3	62.1	86.7	77.8	87.8	87.7
RangeNet++ [27] + PointPillars [20]	2.4	78.8	63.6	65.9	74.3	83.8	43.8	52.3	82.2	73.5	85.3	84.6
Ours (**without** category loss)	11.8	77.4	59.9	62.6	71.4	81.7	37.7	47.2	78.0	71.0	83.4	83.5
Ours (**with** category loss)	11.8	77.8	65.8	68.1	77.2	83.5	53.6	63.8	82.5	71.9	84.0	84.0

TABLE II: Comparison of test set results on SemanticKITTI using *stuff*(St) and *thing*(Th) *categories*. All results in [%].

OS	Method	mIoU	PQ	PQ†	FPS
32	Transpose convolution	46.4	30.8	41.9	11.1
8	Transpose convolution	48.5	31.2	42.6	4.2
32	Ours (trilinear)	50.7	36.5	46.1	11.8

TABLE III: Influence of upsampling method evaluated on validation set with respect to panoptic quality and runtime.

Center	Embedding	Learnt Radius	RQ^{Th}	PQ
✓			23.7	34.4
✓		✓	26.7	35.8
	✓		18.0	32.8
	✓	✓	20.0	33.3
✓	✓	✓	28.2	36.5

TABLE IV: Features used for clustering of *things* on validation set with respect to recognition and panoptic quality.

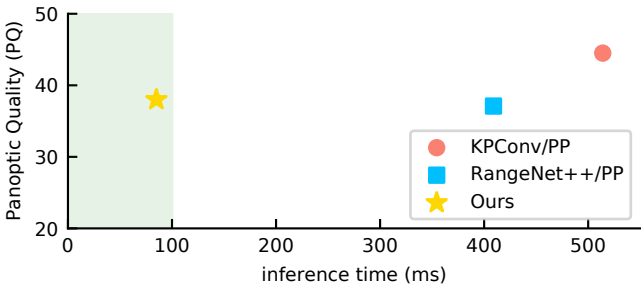


Fig. 6: Runtime of the evaluated approaches. Green area represents the zone of approaches faster than the rate of the sensor.

V. Conclusion

In this paper, we propose a novel approach for single-stage LiDAR-based panoptic segmentation which achieves high panoptic quality while still running over the frame rate of the sensor. Our experiments show that our approach achieves results that are on par with, but faster than the best performing real-time approach on SemanticKITTI. Our ablations studies also show that the addition of the novel range-image-based trilinear upsampling module allows our approach to use larger output strides than approaches using transpose convolutions, resulting in faster runtime without sacrificing accuracy. Furthermore, we show that the combination of our geometric and semantic feature embeddings helps increase the performance of the approach in terms of recognition quality. This is also the case for our learned point-wise radii, which adapts the clustering threshold for points of different difficulty. Finally, the addition of a category loss makes category-based results more robust. This means that if we get a class wrong, we will more likely confuse it with an instance of the same category, which is desired behavior.

References

[1] 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.

[2] J. Behley, A. Milioto, and Cyrill Stachniss. A Benchmark for LiDAR-based Panoptic Segmentation based on KITTI. *arXiv preprint*, 2020.

[3] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. In *Deep Learning for Robotic Vision workshop, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[4] Z. Chen, V. Badrinarayanan, C.Y. Lee, and A. Rabinovich. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. *arXiv preprint*, 2017. Proceedings of the 35th International Conference on Machine Learning (2018), 793-802.

[5] B. Cheng, M.D. Collins, Y. Zhu, T. Liu, T.S. Huang, H. Adam, and L. Chen. Panoptic-DeepLab. In *Proc. of the ICCV Workshop: Joint COCO and Mapillary Recognition Challlenge Workshop*, 2019.

[6] M. Cordts, S. Mohamed Omran, Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] F. Engelmann, T. Kontogianni, J. Schult, and B. Leibe. Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds. *arXiv preprint*, 2018.

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[9] B. Graham, M. Engelcke, and L. van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] F. Groh, P. Wieschollek, and H. Lensch. Flex-Convolution (Million-Scale Pointcloud Learning Beyond Grid-Worlds). In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, Dezember 2018.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[12] J. Hou, A. Dai, and M. Niessner. 3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[13] B. Hua, M. Tran, and S. Yeung. Pointwise Convolutional Neural Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[14] J. Huang and S. You. Point Cloud Labeling using 3D Convolutional Neural Network. In *Proc. of the Internation Conference on Pattern Recognition (ICPR)*, 2016.

[15] M. Jiang, Y. Wu, and C. Lu. PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation. *arXiv preprint*, 2018.

[16] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] R. Klukov and V. Lempitsky. Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[18] J.D. Lafferty, A. McCallum, and F.C.N. Pereira. Conditional Random Fields: Probabilistic Models for Segmentating and Labeling Sequence Data. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2001.

[19] L. Landrieu and M. Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] A.H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[21] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521:436–444, 2015.

[22] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang. Attention-Guided Unified Network for Panoptic Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[23] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 740–755, 2014.

[24] H. Liu, C. Peng, C. Yu, J. Wang, X. Liu, G. Yu, and W. Jiang. An End-To-End Network for Panoptic Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[26] A. Milioto, L. Mandtler, and C. Stachniss. Fast Instance and Semantic Segmentation Exploiting Local Connectivity, Metric Learning, and One-Shot Detection for Robotics. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[27] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[28] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[29] D. Neven, B. De Brabandere, M. Proesmans, and L. Van Gool. Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] Q.H. Pham, D.T. Nguyen, B.S. Hua, G. Roig, and S.K. Yeung. JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds With Multi-Task Pointwise Networks and Multi-Value Conditional Random Fields. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[31] L. Porzi, S. Rota Bulo, A. Colovic, and P. Kontschieder. Seamless Scene Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[32] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J .Guibas. Deep Hough Voting for 3D Object Detection in Point Clouds. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2019.

[34] C.R. Qi, K. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2017.

[35] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint*, 2018.

[36] D. Rethage, J. Wald, J. Sturm, N. Navab, and F. Tombari. Fully-Convolutional Point Networks for Large-Scale Point Clouds. *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2018.

[37] G. Riegler, A. Ulusoy, and A. Geiger. OctNet: Learning Deep 3D Representations at High Resolutions. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[38] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M-H. Yang, and J. Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese. SEGCloud: Semantic Segmentation of 3D Point Clouds. In *Proc. of the International Conference on 3D Vision (3DV)*, 2017.

[40] G. Te, W. Hu, Z. Guo, and A. Zheng. RGCNN: Regularized Graph CNN for Point Cloud Segmentation. *arXiv preprint*, 2018.

[41] H. Thomas, C.R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L.J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. *arXiv preprint*, 2019.

[42] W. Wang, R. Yu, Q. Huang, and U. Neumann. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[43] X. Wang, S. Liu, X. Shen, C. Shen, and J. Jia. Associatively Segmenting Instances and Semantics in Point Clouds. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[44] Z. Wang and F. Lu. VoxSegNet: Volumetric CNNs for Semantic Part Segmentation of 3D Shapes. *arXiv preprint*, 2018.

[45] 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 *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[46] 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. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[47] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun. UPSNet: A Unified Panoptic Segmentation Network. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[48] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[49] L. Yi, W. Zhao, H. Wang, M. Sung, and L. Guibas. GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] W. Zeng and T. Gevers. 3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local and Global Contextual Cues. *arXiv preprint*, 2017.

[O] J. Behley, M. Garbarde, A. Milioto, *et al.*, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019. DOI: 10.1109/ICCV.2019.00939

This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences

Jens Behley* Martin Garbade* Andres Milioto Jan Quenzel
Sven Behnke Cyrill Stachniss Juergen Gall
University of Bonn, Germany

www.semantic-kitti.org

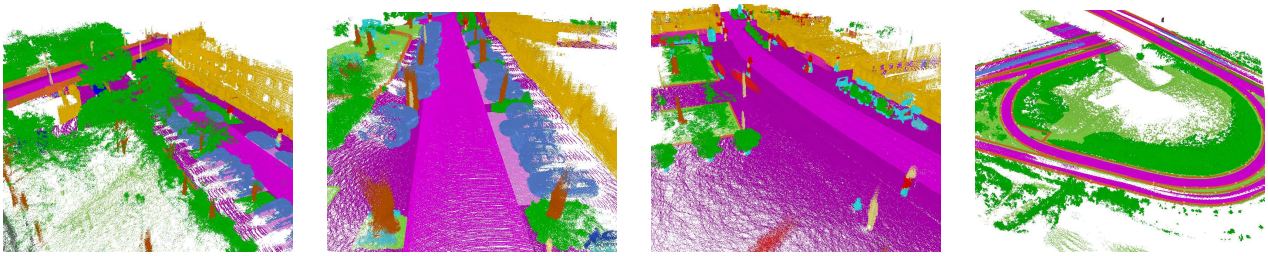


Figure 1: Our dataset provides dense annotations for each scan of all sequences from the KITTI Odometry Benchmark [19]. Here, we show multiple scans aggregated using pose information estimated by a SLAM approach.

Abstract

Semantic scene understanding is important for various applications. In particular, self-driving cars need a fine-grained understanding of the surfaces and objects in their vicinity. Light detection and ranging (LiDAR) provides precise geometric information about the environment and is thus a part of the sensor suites of almost all self-driving cars. Despite the relevance of semantic scene understanding for this application, there is a lack of a large dataset for this task which is based on an automotive LiDAR.

In this paper, we introduce a large dataset to propel research on laser-based semantic segmentation. We annotated all sequences of the KITTI Vision Odometry Benchmark and provide dense point-wise annotations for the complete 360^o field-of-view of the employed automotive LiDAR. We propose three benchmark tasks based on this dataset: (i) semantic segmentation of point clouds using a single scan, (ii) semantic segmentation using multiple past scans, and (iii) semantic scene completion, which requires to anticipate the semantic scene in the future. We provide baseline experiments and show that there is a need for more sophisticated models to efficiently tackle these tasks. Our dataset opens the door for the development of more advanced methods, but also provides plentiful data to investigate new research directions.

1. Introduction

Semantic scene understanding is essential for many applications and an integral part of self-driving cars. Particularly, fine-grained understanding provided by semantic segmentation is necessary to distinguish drivable and non-drivable surfaces and to reason about functional properties, like parking areas and sidewalks. Currently, such understanding, represented in so-called high definition maps, is mainly generated in advance using surveying vehicles. However, self-driving cars should also be able to drive in unmapped areas and adapt their behavior if there are changes in the environment.

Most self-driving cars currently use multiple different sensors to perceive the environment. Complementary sensor modalities enable to cope with deficits or failures of particular sensors. Besides cameras, light detection and ranging (LiDAR) sensors are often used as they provide precise distance measurements that are not affected by lighting.

Publicly available datasets and benchmarks are crucial for empirical evaluation of research. They mainly fulfill three purposes: (i) they provide a basis to measure progress, since they allow to provide results that are reproducible and comparable, (ii) they uncover shortcomings of the current state of the art and therefore pave the way for novel approaches and research directions, and (iii) they make it possible to develop approaches without the need to first painstakingly collect and label data. While multiple

* indicates equal contribution

	#scans[1]	#points[2]	#classes[3]	sensor	annotation	sequential
SemanticKITTI (Ours)	**23201/20351**	**4549**	**25** (28)	Velodyne HDL-64E	point-wise	✓
Oakland3d [36]	17	1.6	5 (44)	SICK LMS	point-wise	✗
Freiburg [50, 6]	77	1.1	4 (11)	SICK LMS	point-wise	✗
Wachtberg [6]	5	0.4	5 (5)	Velodyne HDL-64E	point-wise	✗
Semantic3d [23]	15/15	4009	8 (8)	Terrestrial Laser Scanner	point-wise	✗
Paris-Lille-3D [47]	3	143	9 (50)	Velodyne HDL-32E	point-wise	✗
Zhang et al. [65]	140/112	32	10 (10)	Velodyne HDL-64E	point-wise	✗
KITTI [19]	7481/7518	1799	3	Velodyne HDL-64E	bounding box	✗

Table 1: Overview of other point cloud datasets with semantic annotations. Ours is by far the largest dataset with sequential information. [1]Number of scans for train and test set, [2]Number of points is given in millions, [3]Number of classes used for evaluation and number of classes annotated in brackets.

large datasets for *image-based* semantic segmentation exist [10, 39], publicly available datasets with point-wise annotation of three-dimensional point clouds are still comparably small, as shown in Table 1.

To close this gap we propose *SemanticKITTI*, a large dataset showing unprecedented detail in point-wise annotation with 28 classes, which is suited for various tasks. In this paper, we mainly focus on *laser-based* semantic segmentation, but also semantic scene completion. The dataset is distinct from other laser datasets as we provide accurate scan-wise annotations of sequences. Overall, we annotated all 22 sequences of the odometry benchmark of the *KITTI Vision Benchmark* [19] consisting of over 43 000 scans. Moreover, we labeled the complete horizontal 360° field-of-view of the rotating laser sensor. Figure 1 shows example scenes from the provided dataset. In summary, our main contributions are:

- We present a point-wise annotated dataset of point cloud sequences with an unprecedented number of classes and unseen level-of-detail for each scan.
- We furthermore provide an evaluation of state-of-the-art methods for semantic segmentation of point clouds.
- We investigate the usage of sequence information for semantic segmentation using multiple scans.
- Based on the annotation of sequences of a moving car, we furthermore introduce a real-world dataset for semantic scene completion and provide baseline results.
- Together with a benchmark website, the point cloud labeling tool is also publicly available, enabling other researchers to generate other labeled datasets in future.

This large dataset will stimulate the development of novel algorithms, make it possible to investigate new research directions, and puts evaluation and comparison of these novel algorithms on a more solid ground.

2. Related Work

The progress of computer vision has always been driven by benchmarks and datasets [55], but the availability of especially large-scale datasets, such as *ImageNet* [13], was even a crucial prerequisite for the advent of deep learning.

More task-specific datasets geared towards self-driving cars were also proposed. Notable is here the *KITTI Vision Benchmark* [19] since it showed that off-the-shelf solutions are not always suitable for autonomous driving. The *Cityscapes* dataset [10] is the first dataset for self-driving car applications that provides a considerable amount of pixel-wise labeled images suitable for deep learning. The *Mapillary Vistas* dataset [39] surpasses the amount and diversity of labeled data compared to *Cityscapes*.

Also in point cloud-based interpretation, *e.g.*, semantic segmentation, RGB-D based datasets enabled tremendous progress. *ShapeNet* [8] is especially noteworthy for point clouds showing a single object, but such data is not directly transferable to other domains. Specifically, LiDAR sensors usually do not cover objects as densely as an RGB-D sensor due to their lower angular resolution, in particular in vertical direction.

For indoor environments, there are several datasets [48, 46, 24, 3, 11, 35, 32, 12] available, which are mainly recorded using RGB-D cameras or synthetically generated. However, such data shows very different characteristics compared to outdoor environments, which is also caused by the size of the environment, since point clouds captured indoors tend to be much denser due to the range at which objects are scanned. Furthermore, the sensors have different properties regarding sparsity and accuracy. While laser sensors are more precise than RGB-D sensors, they usually only capture a sparse point cloud compared to the latter.

For outdoor environments, datasets were recently proposed that are recorded with a terrestrial laser scanner (TLS), like the *Semantic3d* dataset [23], or using automotive LiDARs, like the *Paris-Lille-3D* dataset [47]. However, the *Paris-Lille-3D* provides only the aggregated scans with

point-wise annotations for 50 classes from which 9 are selected for evaluation. Another recently used large dataset for autonomous driving [57], but with fewer classes, is not publicly available.

The *Virtual KITTI* dataset [17] provides synthetically generated sequential images with depth information and dense pixel-wise annotation. The depth information can also be used to generate point clouds. However, these point clouds do not show the same characteristics as a real rotating LiDAR, including defects like reflections and outliers.

In contrast to these datasets, our dataset combines a large amount of labeled points, a large variety of classes, and sequential scans generated by a commonly employed sensor used in autonomous driving, which is distinct from all publicly available datasets, also shown in Table 1.

3. The SemanticKITTI Dataset

Our dataset is based on the odometry dataset of the KITTI Vision Benchmark [19] showing inner city traffic, residential areas, but also highway scenes and countryside roads around Karlsruhe, Germany. The original odometry dataset consists of 22 sequences, splitting sequences 00 to 10 as training set, and 11 to 21 as test set. For consistency with the original benchmark, we adopt the same division for our training and test set. Moreover, we do not interfere with the original odometry benchmark by providing labels only for the training data. Overall, we provide 23 201 full 3D scans for training and 20 351 for testing, which makes it by a wide margin the largest dataset publicly available.

We decided to use the KITTI dataset as a basis for our labeling effort, since it allowed us to exploit one of the largest available collections of raw point cloud data captured with a car. We furthermore expect that there are also potential synergies between our annotations and the existing benchmarks and this will enable the investigation and evaluation of additional research directions, such as the usage of semantics for laser-based odometry estimation.

Compared to other datasets (cf. Table 1), we provide labels for sequential point clouds generated with a commonly used automotive LiDAR, *i.e.*, the Velodyne HDL-64E. Other publicly available datasets, like *Paris-Lille-3D* [47] or *Wachtberg* [6], also use such sensors, but only provide the aggregated point cloud of the whole acquired sequence or some individual scans of the whole sequence, respectively. Since we provide the individual scans of the whole sequence, one can also investigate how aggregating multiple consecutive scans influences the performance of the semantic segmentation and use the information to recognize moving objects.

We annotated 28 classes, where we ensured a large overlap of classes with the *Mapillary Vistas* dataset [39] and *Cityscapes* dataset [10] and made modifications where necessary to account for the sparsity and vertical field-of-view. More specifically, we do not distinguish between persons riding a vehicle and the vehicle, but label the vehicle and the person as either *bicyclist* or *motorcyclist*.

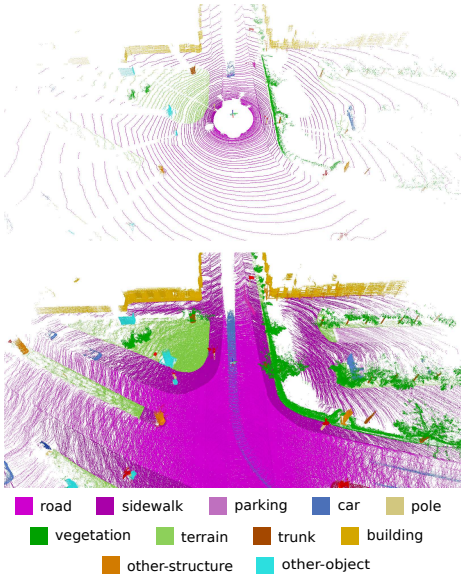


Figure 2: Single scan (top) and multiple superimposed scans with labels (bottom). Also shown is a moving car in the center of the image resulting in a trace of points.

We furthermore distinguished between moving and non-moving vehicles and humans, *i.e.*, vehicles or humans gets the corresponding moving class if they moved in some scan while observing them, as shown in the lower part of Figure 2. All annotated classes are listed in Figure 3 and a more detailed discussion and definition of the different classes can be found in the supplementary material. In summary, we have 28 classes, where 6 classes are assigned the attribute moving or non-moving, and one *outlier* class is included for erroneous laser measurements caused by reflections or other effects.

The dataset is publicly available through a benchmark website and we provide only the training set with ground truth labels and perform the test set evaluation online. We furthermore will also limit the number of possible test set evaluations to prevent overfitting to the test set [55].

3.1. Labeling Process

To make the labeling of point cloud sequences practical, we superimpose multiple scans above each other, which conversely allows us to label multiple scans consistently. To this end, we first register and loop close the sequences using an off-the-shelf laser-based SLAM system [5]. This step is needed as the provided information of the inertial navigation system (INS) often results in map inconsistencies, *i.e.*, streets that are revisited after some time have differ-

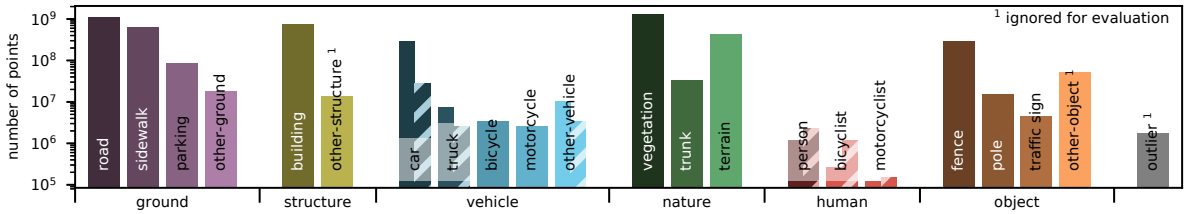


Figure 3: Label distribution. The number of labeled points per class and the root categories for the classes are shown. For movable classes, we also show the number of points on non-moving (solid bars) and moving objects (hatched bars).

ent height. For three sequences, we had to manually add loop closure constraints to get correctly loop closed trajectories, since this is essential to get consistent point clouds for annotation. The loop closed poses allow us to load all overlapping point clouds for specific locations and visualize them together, as depicted in Figure 2.

We subdivide the sequence of point clouds into tiles of 100 m by 100 m. For each tile, we only load scans overlapping with the tile. This enables us to label all scans consistently even when we encounter temporally distant loop closures. To ensure consistency for scans overlapping with more than one tile, we show all points inside each tile and a small boundary overlapping with neighboring tiles. Thus, it is possible to continue labels from a neighboring tile.

Following best practices, we compiled a labeling instruction and provided instructional videos on how to label certain objects, such as cars and bicycles standing near a wall. Compared to image-based annotation, the annotation process with point clouds is more complex, since the annotator often needs to change the viewpoint. An annotator needs on average 4.5 hours per tile, when labeling residential areas corresponding to the most complex encountered scenery, and needs on average 1.5 hours for labeling a highway tile.

We explicitly did not use bounding boxes or other available annotations for the KITTI dataset, since we want to ensure that the labeling is consistent and the point-wise labels should only contain the object itself.

We provided regular feedback to the annotators to improve the quality and accuracy of labels. Nevertheless, a single annotator also verified the labels in a second pass, *i.e.*, corrected inconsistencies and added missing labels. In summary, the whole dataset comprises 518 tiles and over 1 400 hours of labeling effort have been invested with additional $10 - 60$ minutes verification and correction per tile, resulting in a total of over 1 700 hours.

3.2. Dataset Statistics

Figure 3 shows the distribution of the different classes, where we also included the root categories as labels on the x-axis. The ground classes, *road*, *sidewalk*, *building*, *vegetation*, and *terrain* are the most frequent classes. The class *motorcyclist* only occurs rarely, but still more than 100 000 points are annotated.

The unbalanced count of classes is common for datasets captured in natural environments and some classes will be always under-represented, since they do not occur that often. Thus, an unbalanced class distribution is part of the problem that an approach has to master. Overall, the distribution and relative differences between the classes is quite similar in other datasets, *e.g.* *Cityscapes* [10].

4. Evaluation of Semantic Segmentation

In this section, we provide the evaluation of several state-of-the-art methods for semantic segmentation of a single scan. We also provide experiments exploiting information provided by sequences of multiple scans.

4.1. Single Scan Experiments

Task and Metrics. In semantic segmentation of point clouds, we want to infer the label of each three-dimensional point. Therefore, the input to all evaluated methods is a list of coordinates of the three-dimensional points along with their remission, *i.e.*, the strength of the reflected laser beam which depends on the properties of the surface that was hit. Each method should then output a label for each point of a scan, *i.e.*, one full turn of the rotating LiDAR sensor.

To assess the labeling performance, we rely on the commonly applied mean Jaccard Index or mean intersection-over-union (mIoU) metric [15] over all classes, given by

$$\frac{1}{C} \sum_{c=1}^{C} \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c + \mathrm{FN}_c}, \tag{1}$$

where TP_c, FP_c, and FN_c correspond to the number of true positive, false positive, and false negative predictions for class c, and C is the number of classes.

As the classes *other-structure* and *other-object* have either only a few points and are otherwise too diverse with a high intra-class variation, we decided to not include these classes in the evaluation. Thus, we use 25 instead of 28 classes, ignoring *outlier*, *other-structure*, and *other-object* during training and inference.

Furthermore, we cannot expect to distinguish moving from non-moving objects with a single scan, since this Velodyne LiDAR cannot measure velocities like radars exploiting the Doppler effect. We therefore combine the moving classes with the corresponding non-moving class resulting in a total number of 19 classes for training and evaluation.

State of the Art. Semantic segmentation or point-wise classification of point clouds is a long-standing topic [2], which was traditionally solved using a feature extractor, such as Spin Images [29], in combination with a traditional classifier, like support vector machines [1] or even semantic hashing [4]. Many approaches used Conditional Random Fields (CRF) to enforce label consistency of neighboring points [56, 37, 36, 38, 62].

With the advent of deep learning approaches in image-based classification, the whole pipeline of feature extraction and classification has been replaced by end-to-end deep neural networks. Voxel-based methods transforming the point cloud into a voxel-grid and then applying convolutional neural networks (CNN) with 3D convolutions for object classification [34] and semantic segmentation [26] were among the first investigated models, since they allowed to exploit architectures and insights known for images.

To overcome the limitations of the voxel-based representation, such as the exploding memory consumption when the resolution of the voxel grid increases, more recent approaches either upsample voxel-predictions [53] using a CRF or use different representations, like more efficient spatial subdivisions [30, 44, 63, 59, 21], rendered 2D image views [7], graphs [31, 54], splats [51], or even directly the points [41, 40, 25, 22, 43, 28, 14].

Baseline approaches. We provide the results of six state-of-the-art architectures for the semantic segmentation of point clouds in our dataset: PointNet [40], PointNet++ [41], Tangent Convolutions [52], SPLATNet [51], Superpoint Graph [31], and SqueezeSeg (V1 and V2) [60, 61]. Furthermore, we investigate two extensions of SqueezeSeg: DarkNet21Seg and DarkNet53Seg.

PointNet [40] and PointNet++ [41] use the raw unordered point cloud data as input. Core of these approaches is max pooling to get an order-invariant operator that works surprisingly well for semantic segmentation of shapes and several other benchmarks. Due to this nature, however, PointNet fails to capture the spatial relationships between the features. To alleviate this, PointNet++ [41] applies individual PointNets to local neighborhoods and uses a hierarchical approach to combine their outputs. This enables it to build complex hierarchical features that capture both local fine-grained and global contextual information.

Tangent Convolutions [52] also handles unstructured point clouds by applying convolutional neural networks directly on surfaces. This is achieved by assuming that the data is sampled from smooth surfaces and defining a tangent convolution as a convolution applied to the projection of the local surface at each point into the tangent plane.

SPLATNet [51] takes an approach that is similar to the aforementioned voxelization methods and represents the point clouds in a high-dimensional sparse lattice. As with voxel-based methods, this scales poorly both in computation and in memory cost and therefore they exploit the sparsity of this representation by using bilateral convolutions [27], which only operates on occupied lattice parts.

Similarly to PointNet, Superpoint Graph [31], captures the local relationships by summarizing geometrically homogeneous groups of points into superpoints, which are later embedded by local PointNets. The result is a superpoint graph representation that is more compact and rich than the original point cloud exploiting contextual relationships between the superpoints.

SqueezeSeg [60, 61] also discretizes the point cloud in a way that makes it possible to apply 2D convolutions to the point cloud data exploiting the sensor geometry of a rotating LiDAR. In the case of a rotating LiDAR, all points of a single turn can be projected to an image by using a spherical projection. A fully convolutional neural network is applied and then finally filtered with a CRF to smooth the results. Due to the promising results of SqueezeSeg and the fast training, we investigated how the labeling performance is affected by the number of model parameters. To this end, we used a different backbone based on the Darknet architecture [42] with 21 and 53 layers, and 25 and 50 million parameters respectively. We furthermore eliminated the vertical downsampling used in the architecture.

We modified the available implementations such that the methods could be trained and evaluated on our large-scale dataset. Note that most of these approaches have so far only been evaluated on shape [8] or RGB-D indoor datasets [48]. However, some of the approaches [40, 41] were only possible to run with considerable downsampling to 50 000 points due to memory limitations.

Results and Discussion. Table 2 shows the results of our baseline experiments for various approaches using either directly the point cloud information [40, 41, 51, 52, 31] or a projection of the point cloud [60]. The results show that the current state of the art for point cloud semantic segmentation falls short for the size and complexity of our dataset.

We believe that this is mainly caused by the limited capacity of the used architectures (see Table 3), because the number of parameters of these approaches is much lower than the number of parameters used in leading image-based semantic segmentation networks. As mentioned above, we added DarkNet21Seg and DarkNet53Seg to test this hypothesis and the results show that this simple modification improves the accuracy from 29.5 % for SqueezeSeg to 47.4 % for DarkNet21Seg and to 49.9 % for DarkNet53Seg.

Approach	mIoU	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign
PointNet [40]	14.6	61.6	35.7	15.8	1.4	41.4	46.3	0.1	1.3	0.3	0.8	31.0	4.6	17.6	0.2	0.2	0.0	12.9	2.4	3.7
SPGraph [31]	17.4	45.0	28.5	0.6	0.6	64.3	49.3	0.1	0.2	0.2	0.8	48.9	27.2	24.6	0.3	2.7	0.1	20.8	15.9	0.8
SPLATNet [51]	18.4	64.6	39.1	0.4	0.0	58.3	58.2	0.0	0.0	0.0	0.0	71.1	9.9	19.3	0.0	0.0	0.0	23.1	5.6	0.0
PointNet++ [41]	20.1	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9
SqueezeSeg [60]	29.5	85.4	54.3	26.9	4.5	57.4	68.8	3.3	16.0	4.1	3.6	60.0	24.3	53.7	12.9	13.1	0.9	29.0	17.5	24.5
SqueezeSegV2 [61]	39.7	88.6	67.6	45.8	17.7	73.7	81.8	13.4	18.5	17.9	14.0	71.8	35.8	60.2	20.1	25.1	3.9	41.1	20.2	36.3
TangentConv [52]	40.9	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1	79.5	49.3	58.1	23.0	28.4	8.1	49.0	35.8	28.5
DarkNet21Seg	47.4	91.4	74.0	57.0	26.4	81.9	85.4	18.6	26.2	26.5	15.6	77.6	48.4	63.6	31.8	33.6	4.0	52.3	36.0	50.0
DarkNet53Seg	49.9	91.8	74.6	64.8	27.9	84.1	86.4	25.5	24.5	32.7	22.6	78.3	50.1	64.0	36.2	33.6	4.7	55.0	38.9	52.2

Table 2: Single scan results (19 classes) for all baselines on sequences 11 to 21 (test set). All methods were trained on sequences 00 to 10, except for sequence 08 which is used as validation set.

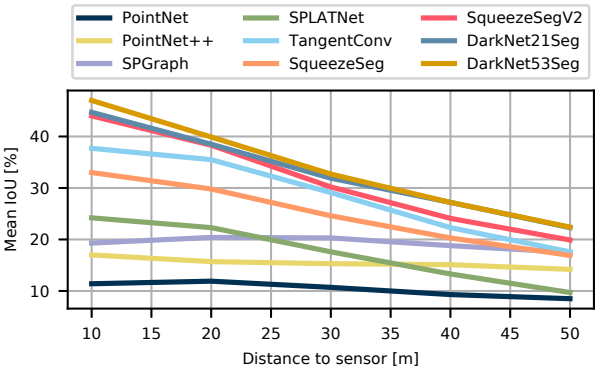


Figure 4: IoU vs. distance to the sensor.

Approach	num. parameters (million)	train time $\left(\frac{\text{GPU hours}}{\text{epoch}}\right)$	inference time $\left(\frac{\text{seconds}}{\text{point cloud}}\right)$
PointNet	3	4	0.5
PointNet++	6	16	5.9
SPGraph	0.25	6	5.2
TangentConv	0.4	6	3.0
SPLATNet	0.8	8	1.0
SqueezeSeg	1	0.5	0.015
SqueezeSegV2	1	0.6	0.02
DarkNet21Seg	25	2	0.055
DarkNet53Seg	50	3	0.1

Table 3: Approach statistics.

Another reason is that the point clouds generated by LiDAR are relatively sparse, especially as the distance to the sensor increases. This is partially solved in SqueezeSeg, which exploits the way the rotating scanner captures the data to generate a dense range image, where each pixel corresponds roughly to a point in the scan.

These effects are further analyzed in Figure 4, where the mIoU is plotted w.r.t. the distance to the sensor. It shows that results of all approaches get worse with increasing distance. This further confirms our hypothesis that the sparsity is the main reason for worse results at large distances. However, the results also show that some methods, like SPGraph, are less affected by the distance-dependent sparsity and this might be a promising direction for future research to combine the strength of both paradigms.

Especially classes with few examples, like motorcyclists and trucks, seem to be more difficult for all approaches. But also classes with only a small number of points in a single point cloud, like bicycles and poles, are hard classes.

Finally, the best performing approach (DarkNet53Seg) with 49.9% mIoU is still far from achieving results that are on par with image-based approaches, *e.g.*, 80% on the *Cityscapes* benchmark [10].

4.2. Multiple Scan Experiments

Task and Metrics. In this task, we allow methods to exploit information from a sequence of multiple past scans to improve the segmentation of the current scan. We furthermore want the methods to distinguish moving and non-moving classes, *i.e.*, all 25 classes must be predicted, since this information should be visible in the temporal information of multiple past scans. The evaluation metric for this task is still the same as in the single scan case, *i.e.*, we evaluate the mean IoU of the current scan no matter how many past scans were used to compute the results.

Baselines. We exploit the sequential information by combining 5 scans into a single, large point cloud, *i.e.*, the current scan at timestamp t and the 4 scans before at timestamps $t-1, \ldots, t-4$. We evaluate DarkNet53Seg and TangentConv, since these approaches can deal with a larger number of points without downsampling of the point clouds and could still be trained in a reasonable amount of time.

Results and Discussion. Table 4 shows the per-class results for the movable classes and the mean IoU (mIoU) over all classes. For each method, we show in the upper part of the row the IoU for non-moving (unshaded) and in the lower part of the row the IoU for moving objects (shaded). The

Approach	car	truck	other-vehicle	person	bicyclist	motorcyclist	mIoU
TangentConv [52]	84.9	21.1	18.5	1.6	0.0	0.0	34.1
	40.3	42.2	30.1	6.4	1.1	1.9	
DarkNet53Seg	84.1	20.0	20.7	7.5	0.0	0.0	41.6
	61.5	37.8	28.9	15.2	14.1	0.2	

Table 4: IoU results using a sequence of multiple past scans (in %). Shaded cells correspond to the IoU of the moving classes, while unshaded entries are the non-moving classes.

performance of the remaining static classes is similar to the single scan results and we refer to the supplement for a table containing all classes.

The general trend that the projective methods perform better than the point-based methods is still apparent, which can be also attributed to the larger amount of parameters as in the single scan case. Both approaches show difficulties in separating moving and non-moving objects, which might be caused by our design decision to aggregate multiple scans into a single large point cloud. The results show that especially bicyclist and motorcyclist never get correctly assigned the non-moving class, which is most likely a consequence from the generally sparser object point clouds.

We expect that new approaches could explicitly exploit the sequential information by using multiple input streams to the architecture or even recurrent neural networks to account for the temporal information, which again might open a new line of research.

5. Evaluation of Semantic Scene Completion

After leveraging a sequence of past scans for semantic point cloud segmentation, we now show a scenario that makes use of future scans. Due to its sequential nature, our dataset provides the unique opportunity to be extended for the task of 3D semantic scene completion. Note that this is the first real world outdoor benchmark for this task. Existing point cloud datasets cannot be used to address this task, as they do not allow for aggregating labeled point clouds that are sufficiently dense in both space and time.

In semantic scene completion, one fundamental problem is to obtain ground truth labels for real world datasets. In case of NYUv2 [48], CAD models were fit into the scene [45] using an RGB-D image captured by a Kinect sensor. New approaches often resort to prove their effectiveness on the larger, but synthetic SUNCG dataset [49]. However, a dataset combining the scale of a synthetic dataset and usage of real-world data is still missing.

In the case of our proposed dataset, the car carrying the LiDAR moves past 3D objects in the scene and thereby records their backsides, which are hidden in the initial scan due to self-occlusion. This is exactly the information needed for semantic scene completion as it contains the full 3D geometry of all objects while their semantics are provided by our dense annotations.

Dataset Generation. By superimposing an exhaustive number of future laser scans in a predefined region in front of the car, we can generate pairs of inputs and targets that correspond to the task of semantic scene completion. As proposed by Song *et al*. [49], our dataset for the scene completion task is a voxelized representation of the 3D scene.

We select a volume of $51.2\,\text{m}$ ahead of the car, $25.6\,\text{m}$ to every side and $6.4\,\text{m}$ in height with a voxel resolution of $0.2\,\text{m}$, which results in a volume of $256 \times 256 \times 32$ voxels to predict. We assign a single label to every voxel based on the majority vote over all labeled points inside a voxel. Voxels that do not contain any points are labeled as *empty*.

To compute which voxels belong to the occluded space, we check for every pose of the car which voxels are visible to the sensor by tracing a ray. Some of the voxels, *e.g*. those inside objects or behind walls are never visible, so we ignore them during training and evaluation.

Overall, we extracted $19\,130$ pairs of input and target voxel grids for training, 815 for validation and $3\,992$ for testing. For the test set, we only provide the unlabeled input voxel grid and withhold the target voxel grids. Figure 5 shows an example of an input and target pair.

Task and Metrics. In semantic scene completion, we are interested in predicting the complete scene inside a certain volume from a single initial scan. More specifically, we use as input a voxel grid, where each voxel is marked as empty or occupied, depending on whether or not it contains a laser measurement. For semantic scene completion, one needs to predict whether a voxel is occupied and its semantic label in the completed scene.

For evaluation, we follow the evaluation protocol of Song *et al*. [49] and compute the IoU for the task of scene completion, which only classifies a voxel as being occupied or empty, *i.e*., ignoring the semantic label, as well as mIoU (1) for the task of semantic scene completion over the same 19 classes that were used for the single scan semantic segmentation task (see Section 4).

State of the Art. Early approaches addressed the task of scene completion either without predicting semantics [16], thereby not providing a holistic understanding of the scene, or by trying to fit a fixed number of mesh models to the scene geometry [20], which limits the expressiveness of the approach.

Song *et al*. [49] were the first to address the task of semantic scene completion in an end-to-end fashion. Their work spawned a lot of interest in the field yielding models that combine the usage of color and depth informa-



Figure 5: Left: Visualization of the incomplete input for the semantic scene completion benchmark. Note that we show the labels only for better visualization, but the real input is a single raw voxel grid without any labels. Right: Corresponding target output representing the completed and fully labeled 3D scene.

tion [33, 18] or address the problem of sparse 3D feature maps by introducing submanifold convolutions [64] or increase the output resolution by deploying a multi-stage coarse to fine training scheme [12]. Other works experimented with new encoder-decoder CNN architectures as well as improving the loss term by adding adversarial loss components [58].

Baseline Approaches. We report the results of four semantic scene completion approaches. In the first approach, we apply SSCNet [49] without the flipped TSDF as input feature. This has minimal impact on the performance, but significantly speeds up the training time due to faster preprocessing [18]. Then we use the Two Stream (TS3D) approach [18], which makes use of the additional information from the RGB image corresponding to the input laser scan. Therefore the RGB image is first processed by a 2D semantic segmentation network, using the approach DeepLab v2 (ResNet-101) [9] trained on Cityscapes to generate a semantic segmentation. The depth information from the single laser scan and the labels inferred from the RGB image are combined in an early fusion. Furthermore, we modify the TS3D approach in two steps: First, by directly using labels from the best LiDAR-based semantic segmentation approach (DarkNet53Seg) and secondly, by exchanging the 3D-CNN backbone by SATNet [33].

Results and Discussion. Table 5 shows the results of each of the baselines, whereas results for individual classes are reported in the supplement. The TS3D network, incorporating 2D semantic segmentation of the RGB image, performs similar to SSCNet which only uses depth information. However, the usage of the best semantic segmentation directly working on the point cloud slightly outperforms SSCNet on semantic scene completion (TS3D + DarkNet53Seg). Note that the first three approaches are based on SSCNet's 3D-CNN architecture, which performs a 4 fold downsampling in a forward pass and thus renders them incapable of dealing with details of the scene. In our final approach, we exchange the SSCNet-backbone of TS3D + DarkNet53Seg with SATNet [33], which is capable of dealing with the desired output resolution. Due to memory limitations, we use random cropping during training. During inference, we divide each volume into six equal parts, perform scene completion on them individually and subsequently fuse them. This approach performs much better than the SSCNet based approaches.

	Completion (IoU)	Semantic Scene Completion (mIoU)
SSCNet [49]	29.83	9.53
TS3D [18]	29.81	9.54
TS3D [18] + DarkNet53Seg	24.99	10.19
TS3D [18] + DarkNet53Seg + SATNet	50.60	17.70

Table 5: Semantic scene completion baselines.

Apart from dealing with the target resolution, a challenge for current models is the sparsity of the laser input signal in the far field as can be seen from Figure 5. To obtain a higher resolution input signal in the far field, approaches would have to exploit more efficiently information from high resolution RGB images provided along with each laser scan.

6. Conclusion and Outlook

In this work, we have presented a large-scale dataset showing unprecedented scale in point-wise annotation of point cloud sequences. We provide a range of different baseline experiments for three tasks: (i) semantic segmentation using a single scan, (ii) semantic segmentation using multiple scans, and (iii) semantic scene completion.

In future work, we plan to provide also instance-level annotation over the whole sequence, *i.e.*, we want to distinguish different objects in a scan, but also identify the same object over time. This will enable to investigate temporal instance segmentation over sequences. However, we also see potential for other new tasks based on our labeling effort, such as the evaluation of semantic SLAM.

Acknowledgments We thank all students that helped with annotating the data. The work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under FOR 1505 Mapping on Demand, BE 5996/1-1, GA 1927/2-2, and under Germanys Excellence Strategy, EXC-2070 – 390732324 (PhenoRob).

References

[1] Anuraag Agrawal, Atsushi Nakazawa, and Haruo Takemura. MMM-classification of 3D Range Data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009. 5

[2] Dragomir Anguelov, Ben Taskar, Vassil Chatalbashev, Daphne Koller, Dinkar Gupta, Geremy Heitz, and Andrew Ng. Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 169–176, 2005. 5

[3] Iro Armeni, Alexander Sax, Amir R. Zamir, and Silvio Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *arXiv preprint*, 2017. 2

[4] Jens Behley, Kristian Kersting, Dirk Schulz, Volker Steinhage, and Armin B. Cremers. Learning to Hash Logistic Regression for Fast 3D Scan Point Classification. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5960–5965, 2010. 5

[5] Jens Behley and Cyrill Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018. 3

[6] Jens Behley, Volker Steinhage, and Armin B. Cremers. Performance of Histogram Descriptors for the Classification of 3D Laser Range Data in Urban Environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012. 2, 3

[7] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. *Computers & Graphics*, 2017. 5

[8] Angel X. Chang, Thomas Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University and Princeton University and Toyota Technological Institute at Chicago, 2015. 2, 5

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation withDeep Convolutional Nets, Atrous Convolution,and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(4):834–848, 2018. 8

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3, 4, 6

[11] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[12] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 8

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[14] Francis Engelmann, Theodora Kontogianni, Jonas Schult, and Bastian Leibe. Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds. *arXiv preprint*, 2018. 5

[15] Mark Everingham, S.M. Ali Eslami, Luc van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge a Retrospective. *International Journal on Computer Vision (IJCV)*, 111(1):98–136, 2015. 4

[16] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J. Brostow. Structured Prediction of Unobserved Voxels From a Single Depth Image. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5431–5440, 2016. 7

[17] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[18] Martin Garbade, Yueh-Tung Chen, J. Sawatzky, and Juergen Gall. Two Stream 3D Semantic Scene Completion. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019. 8

[19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 1, 2, 3

[20] Andres Geiger and Chaohui Wang. Joint 3d Object and Layout Inference from a single RGB-D Image. In *Proc. of the German Conf. on Pattern Recognition (GCPR)*, pages 183–195, 2015. 7

[21] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[22] Fabian Groh, Patrick Wieschollek, and Hendrik Lensch. Flex-Convolution (Million-Scale Pointcloud Learning Beyond Grid-Worlds). In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, Dezember 2018. 5

[23] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D. Wegner, Konrad Schindler, and Marc Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98, 2017. 2

[24] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. SceneNN: A Scene Meshes Dataset with aNNotations. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2016. 2

[25] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise Convolutional Neural Networks. In *Proc. of the IEEE*

Conf. on Computer Vision and Pattern Recognition (CVPR), 2018. 5

[26] Jing Huang and Suya You. Point Cloud Labeling using 3D Convolutional Neural Network. In *Proc. of the Intl. Conf. on Pattern Recognition (ICPR)*, 2016. 5

[27] Varun Jampani, Martin Kiefel, and Peter V. Gehler. Learning Sparse High Dimensional Filters: Image Filtering, Dense CRFs and Bilateral Neural Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[28] Mingyang Jiang, Yiran Wu, and Cewu Lu. PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation. *arXiv preprint*, 2018. 5

[29] Andrew E. Johnson and Martial Hebert. Using spin images for effcient object recognition in cluttered 3D scenes. *Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(5):433–449, 1999. 5

[30] Roman Klukov and Victor Lempitsky. Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017. 5

[31] Loic Landrieu and Martin Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5, 6

[32] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset. In *Proc. of the British Machine Vision Conference (BMVC)*, 2018. 2

[33] Shice Liu, Yu Hu, Yiming Zeng, Qiankun Tang, Beibei Jin, Yainhe Han, and Xiaowei Li. See and Think: Disentangling Semantic Scene Completion. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, pages 261–272, 2018. 8

[34] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015. 5

[35] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J. Davison. SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation? In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017. 2

[36] Daniel Munoz, J. Andrew Bagnell, Nicolas Vandapel, and Martial Hebert. Contextual Classification with Functional Max-Margin Markov Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2, 5

[37] Daniel Munoz, Nicholas Vandapel, and Marial Hebert. Directional Associative Markov Network for 3-D Point Cloud Classification. In *Proc. of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, pages 63–70, 2008. 5

[38] Daniel Munoz, Nicholas Vandapel, and Martial Hebert. Onboard Contextual Classification of 3-D Point Clouds with Learned High-order Markov Random Fields. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009. 5

[39] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017. 2, 3

[40] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5, 6

[41] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2017. 5, 6

[42] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint*, 2018. 5

[43] Dario Rethage, Johanna Wald, Jürgen Sturm, Nassir Navab, and Frederico Tombari. Fully-Convolutional Point Networks for Large-Scale Point Clouds. *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 5

[44] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning Deep 3D Representations at High Resolutions. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[45] Jason Rock, Tanmay Gupta, Justin Thorsen, JunYoung Gwak, Daeyun Shin, and Derek Hoiem. Completing 3D Object Shape from One Depth Image. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. 7

[46] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[47] Xavier Roynard, Jean-Emmanuel Deschaud, and Francois Goulette. Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *Intl. Journal of Robotics Research (IJRR)*, 37(6):545–557, 2018. 2, 3

[48] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2012. 2, 5, 7

[49] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic Scene Completion from a Single Depth Image. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7, 8

[50] Bastian Steder, Giorgio Grisetti, and Wolfram Burgard. Robust Place Recognition for 3D Range Data based on Point Features. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2010. 2

[51] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5, 6

[52] Maxim Tatarchenko, Jaesik Park, Vladen Koltun, and Qian-Yi Zhou. Tangent Convolutions for Dense Prediction in 3D.

In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5, 6, 7

[53] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, Jun Young Gwak, and Silvio Savarese. SEGCloud: Semantic Segmentation of 3D Point Clouds. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2017. 5

[54] Gusi Te, Wei Hu, Zongming Guo, and Amin Zheng. RGCNN: Regularized Graph CNN for Point Cloud Segmentation. *arXiv preprint*, 2018. 5

[55] Antonio Torralba and Alexei A. Efros. Unbiased Look at Dataset Bias. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011. 2, 3

[56] Rudolph Triebel, Krisitian Kersting, and Wolfram Burgard. Robust 3D Scan Point Classification using Associative Markov Networks. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2603–2608, 2006. 5

[57] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep Parametric Continuous Convolutional Neural Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[58] Yida Wang, Davod Tan Joseph, Nassir Navab, and Frederico Tombari. Adversarial Semantic Scene Completion from a Single Depth Image. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, pages 426–434, 2018. 8

[59] Zongji Wang and Feng Lu. VoxSegNet: Volumetric CNNs for Semantic Part Segmentation of 3D Shapes. *arXiv preprint*, 2018. 5

[60] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018. 5, 6

[61] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019. 5, 6

[62] Xuehan Xiong, Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. 3-D Scene Analysis via Sequenced Predictions over Points and Regions. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2609–2616, 2011. 5

[63] Wei Zeng and Theo Gevers. 3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local and Global Contextual Cues. *arXiv preprint*, 2017. 5

[64] Jiahui Zhang, Hao Zhao, Anbang Yao, Yurong Chen, Li Zhang, and Hongen Liao. Efficient Semantic Scene Completion Network with Spatial Group Convolution. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 733–749, 2018. 8

[65] Richard Zhang, Stefan A. Candra, Kai Vetter, and Avideh Zakhor. Sensor Fusion for Semantic Segmentation of Urban Scenes. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015. 2

[P] X. Chen, A. Milioto, E. Palazzolo, *et al.*, "SuMa++: Efficient LiDAR-based Semantic SLAM," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019. DOI: 10.1109/IROS40897.2019.8967704

2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

SuMa++: Efficient LiDAR-based Semantic SLAM

Xieyuanli Chen Andres Milioto Emanuele Palazzolo Philippe Giguère Jens Behley Cyrill Stachniss

***Abstract*— Reliable and accurate localization and mapping are key components of most autonomous systems. Besides geometric information about the mapped environment, the semantics plays an important role to enable intelligent navigation behaviors. In most realistic environments, this task is particularly complicated due to dynamics caused by moving objects, which can corrupt the mapping step or derail localization. In this paper, we propose an extension of a recently published surfel-based mapping approach exploiting three-dimensional laser range scans by integrating semantic information to facilitate the mapping process. The semantic information is efficiently extracted by a fully convolutional neural network and rendered on a spherical projection of the laser range data. This computed semantic segmentation results in point-wise labels for the whole scan, allowing us to build a semantically-enriched map with labeled surfels. This semantic map enables us to reliably filter moving objects, but also improve the projective scan matching via semantic constraints. Our experimental evaluation on challenging highways sequences from KITTI dataset with very few static structures and a large amount of moving cars shows the advantage of our semantic SLAM approach in comparison to a purely geometric, state-of-the-art approach.**

I. Introduction

Accurate localization and reliable mapping of unknown environments are fundamental for most autonomous vehicles. Such systems often operate in highly dynamic environments, which makes the generation of consistent maps more difficult. Furthermore, semantic information about the mapped area is needed to enable intelligent navigation behavior. For example, a self-driving car must be able to reliably find a location to legally park, or pull over in places where a safe exit of the passengers is possible – even in locations that were never seen, and thus not accurately mapped before.

In this work, we propose a novel approach to simultaneous localization and mapping (SLAM) able to generate such semantic maps by using three-dimensional laser range scans. Our approach exploits ideas from a modern LiDAR SLAM pipeline [2] and incorporates semantic information obtained from a semantic segmentation generated by a Fully Convolutional Neural Network (FCN) [21]. This allows us to generate high-quality semantic maps, while at the same time improve the geometry of the map and the quality of the odometry.

The FCN provides class labels for each point of the laser range scan. We perform highly efficient processing of the point cloud by first using a spherical projection. Classification results on this two-dimensional spherical projection are then back-projected to the three-dimensional point cloud. However, the back-projection introduces artifacts, which we reduce by a two-step process of an erosion followed by depth-based flood-fill of the semantic labels. The semantic labels are then integrated into the surfel-based map representation and exploited to better register new observations to the already built map. We furthermore use the semantics to filter moving objects by checking semantic consistency between the new observation and the world model when updating



Fig. 1: Semantic map of the KITTI dataset generated with our approach using only LiDAR scans. The map is represented by surfels that have a class label indicated by the respective color. Overall, our semantic SLAM pipeline is able to provide high-quality semantic maps with higher metric accuracy than its non-semantic counterpart.

X.Chen, A. Milioto, E. Palazzolo, J. Behley, and C. Stachniss are with the University of Bonn, Germany. Philippe Giguère is with the Laval University, Québec, Canada. This work has partly been supported by the German Research Foundation under Germany's Excellence Strategy, EXC-2070 - 390732324 (PhenoRob) and under grant number BE 5996/1-1 as well as by the Chinese Scholarship Committee.

the map. In this way, we reduce the risk of integrating dynamic objects into the map. Fig. 1 shows an example of our semantic map representation. The semantic classes are generated by the FCN of Milioto *et al*. [21], which was trained using the SemanticKITTI dataset by Behley *et al*. [1].

The main contribution of this paper is an approach to integrate semantics into a surfel-based map representation and a method to filter dynamic objects exploiting these semantic labels. In sum, we claim that we are (i) able to accurately map an environment especially in situations with a large number of moving objects and we are (ii) able to achieve a better performance than the same mapping system simply removing possibly moving objects in general environments, including urban, countryside, and highway scenes. We experimentally evaluated our approach on challenging sequences of KITTI [11] and show superior performance of our semantic surfel-mapping approach, called *SuMa++*, compared to purely geometric surfel-based mapping and compared mapping removing all potentially moving objects based on class labels. The source code of our approach is available at:
`https://github.com/PRBonn/semantic_suma/`

II. Related Work

Odometry estimation and SLAM are classical topics in robotics with a large body of scientific work covered by several overview articles [6], [7], [28]. Here, we mainly concentrate on related work for semantic SLAM based on learning approaches and dynamic scenes.

Motivated by the advances of deep learning and Convolutional Neural Networks (CNNs) for scene understanding, there have been many semantic SLAM techniques exploiting this information using cameras [5], [31], cameras + IMU data [4], stereo cameras [10], [15], [18], [33], [38], or RGB-D sensors [3], [19], [20], [26], [27], [30], [39]. Most of these approaches were only applied indoors and use either an object detector or a semantic segmentation of the camera image. In contrast, we only use laser range data and exploit information from a semantic segmentation operating on depth images generated from LiDAR scans.

There exists also a large body of literature tackling localization and mapping changing environments, for example by filtering moving objects [14], considering residuals in matching [22], or by exploiting sequence information [34]. To achieve outdoor large-scale semantic SLAM, one can also combine 3D LiDAR sensors with RGB cameras. Yan *et al*. [37] associate 2D images and 3D points to improve the segmentation for detecting moving objects. Wang and Kim [35] use images and 3D point clouds from the KITTI dataset [11] to jointly estimate road layout and segment urban scenes semantically by applying a relative location prior. Jeong *et al*. [12], [13] also propose a multi-modal sensor-based semantic 3D mapping system to improve the segmentation results in terms of the intersection-over-union (IoU) metric, in large-scale environments as well as in environments with few features. Liang *et al*. [17] propose a novel 3D object detector that can exploit both LiDAR and camera data to perform accurate object localization. All of these approaches focus on combining 3D LiDAR and cameras to improve the object detection, semantic segmentation, or 3D reconstruction.

The recent work by Parkison *et al*. [23] develops a point cloud registration algorithm by directly incorporating image-based semantic information into the estimation of the relative transformation between two point clouds. A subsequent work by Zaganidis *et al*. [40] realizes both LiDAR combined with images and LiDAR only semantic 3D point cloud registration. Both approaches use semantic information to improve pose estimation, but they cannot be used for online operation because of the long processing time.

The most similar approaches to the one proposed in this paper are Sun *et al*. [29] and Dubé *et al*. [9], which realize semantic SLAM using only a single LiDAR sensor. Sun *et al*. [29] present a semantic mapping approach, which is formulated as a sequence-to-sequence encoding-decoding problem. Dubé *et al*. [9] propose an approach called SegMap, which is based on segments extracted from the point cloud and assigns semantic labels to them. They mainly aim at extracting meaningful features for the global retrieval and multi-robot collaborative SLAM with very limited types of semantic classes. In contrast to them, we focus on generating a semantic map with an abundance of semantic classes and using these semantics to filter outliers caused by dynamic objects, like moving vehicles and humans, to improve both mapping and odometry accuracy.

III. Our Approach

The foundation of our semantic SLAM approach is our Surfel-based Mapping (SuMa) [2] pipeline, which we extend by integrating semantic information provided by a semantic segmentation using the FCN RangeNet++ [21] as illustrated in Fig. 2. The point-wise labels are provided by RangeNet++ using spherical projections of the point clouds. This information is then used to filter dynamic objects and to add semantic constraints to the scan registration, which improves the robustness and accuracy of the pose estimation by SuMa.

A. Notation

We denote the transformation of a point $\mathbf{p}_A$ in coordinate frame A to a point $\mathbf{p}_B$ in coordinate frame B by $\mathbf{T}_{BA} \in \mathbb{R}^{4\times4}$, such that $\mathbf{p}_B = \mathbf{T}_{BA}\mathbf{p}_A$. Let $\mathbf{R}_{BA} \in \mathrm{SO}(3)$ and $\mathbf{t}_{BA} \in \mathbb{R}^3$ denote the corresponding rotational and translational part of transformation $\mathbf{T}_{BA}$.

We call the coordinate frame at timestep t as C_t. Each variable in coordinate frame C_t is associated to the world frame W by a pose $\mathbf{T}_{WC_t} \in \mathbb{R}^{4\times4}$, transforming the observed point cloud into the world coordinate frame.

B. Surfel-based Mapping

Our approach relies on SuMa, but we only summarize here the main steps relevant to our approach and refer for more details to the original paper [2]. SuMa first generates a spherical projection of the point cloud $\mathcal{P}$ at timestep t, the so-called vertex map $\mathcal{V}_D$, which is then used to generate

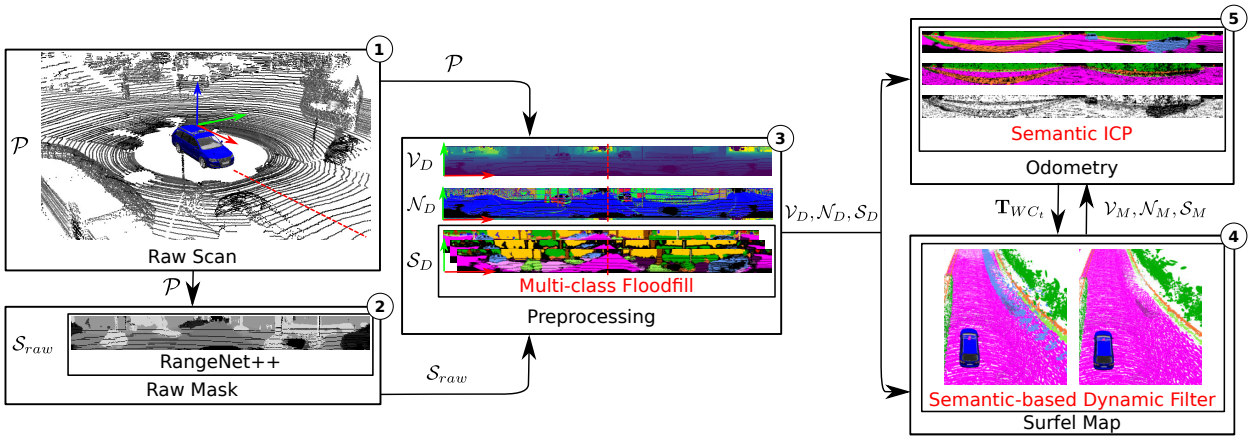


Fig. 2: Pipeline overview of our proposed approach. We integrate semantic predictions into the SuMa pipeline in a compact way: (1) The input is only the LiDAR scan $\mathcal{P}$. (2) Before processing the raw point clouds $\mathcal{P}$, we first use a semantic segmentation from RangeNet++ to predict the semantic label for each point and generate a raw semantic mask $\mathcal{S}_{\text{raw}}$. (3) Given the raw mask, we generate an refined semantic map $\mathcal{S}_D$ in the preprocessing module using multi-class flood-fill. (4) During the map updating process, we add a dynamic detection and removal module which checks the semantic consistency between the new observation $\mathcal{S}_D$ and the world model $\mathcal{S}_M$ and remove the outliers. (5) Meanwhile, we add extra semantic constraints into the ICP process to make it more robust to outliers.

a corresponding normal map $\mathcal{N}_D$. Given this information, SuMa determines via projective ICP in a rendered map view $\mathcal{V}_M$ and $\mathcal{N}_M$ at timestep $t-1$ the pose update $\mathbf{T}_{C_{t-1}C_t}$ and consequently $\mathbf{T}_{WC_t}$ by chaining all pose increments.

The map is represented by surfels, where each surfel is defined by a position $\mathbf{v}_s \in \mathbb{R}^3$, a normal $\mathbf{n}_s \in \mathbb{R}^3$, and a radius $r_s \in \mathbb{R}$. Each surfel additionally carries two timestamps: the creation timestamp t_c and the timestamp t_u of its last update by a measurement. Furthermore, a stability log odds ratio l_s is maintained using a binary Bayes Filter [32] to determine if a surfel is considered stable or unstable. SuMa also performs loop closure detection with subsequent pose-graph optimization to obtain globally consistent maps.

C. Semantic Segmentation

For each frame, we use RangeNet++ [21] to predict a semantic label for each point and generate a semantic map $\mathcal{S}_D$. RangeNet++ semantically segments a range image generated by a spherical projection of each laser scan. Briefly, the network is based on the SqueezeSeg architecture proposed by Wu *et al*. [36] and uses a DarkNet53 backbone proposed by Redmon *et al*. [25] to improve results by using more parameters, while keeping the approach real-time capable. For more details about the semantic segmentation approach, we refer to the paper of Milioto *et al*. [21]. The availability of point-wise labels in the field of view of the sensor makes it also possible to integrate the semantic information into the map. To this end, we add for each surfel the inferred semantic label y and the corresponding probability of that label from the semantic segmentation.

D. Refined Semantic Map

Due to the projective input and the blob-like outputs produced as a by-product of in-network down-sampling of RangeNet++, we have to deal with errors of the semantic labels, when the labels are re-projected to the map. To reduce these errors, we use a flood-fill algorithm, summarized in Alg. 1. It is inside the preprocessing module, which uses depth information from the vertex map $\mathcal{V}_D$ to refine the semantic mask $\mathcal{S}_D$.

Algorithm 1: flood-fill for refining $\mathcal{S}_D$.

Input: semantic mask $\mathcal{S}_{\text{raw}}$ and the corresponding vertex map $\mathcal{V}_D$
Result: refined mask $\mathcal{S}_D$
Let $\mathcal{N}_{\mathbf{s}}$ be the set of neighbors of pixel $\mathbf{s} \in \mathcal{S}$ within a filter kernel of size d.
θ is the rejection threshold.
$\mathbf{0}$ represents an empty pixel with label 0.
foreach $\mathbf{s_u} \in \mathcal{S}_{\text{raw}}$ **do**
  Let $\mathcal{S}_{\text{raw}}^{\text{eroded}}(\mathbf{u}) = \mathbf{s_u}$
  foreach $\mathbf{n} \in \mathcal{N}_{\mathbf{s_u}}$ **do**
    if $y_{\mathbf{s_u}} \neq y_{\mathbf{n}}$ **then**
      Let $\mathcal{S}_{\text{raw}}^{\text{eroded}}(\mathbf{u}) = \mathbf{0}$
      break
    end
  end
end
foreach $\mathbf{s_u} \in \mathcal{S}_{\text{raw}}^{\text{eroded}}$ **do**
  Let $\mathcal{S}_D(\mathbf{u}) = \mathbf{s_u}$
  foreach $\mathbf{n} \in \mathcal{N}_{\mathbf{s_u}}$ **do**
    if $||\mathbf{u} - \mathbf{u_n}|| < \theta \cdot ||\mathbf{u}||$ **then**
      Let $\mathcal{S}_D(\mathbf{u}) = \mathbf{n}$ **if** $y_{\mathbf{s_u}} = 0$
      break
    end
  end
end

The input to the flood-fill is the raw semantic mask $\mathcal{S}_{\text{raw}}$ generated by the RangeNet++ and the corresponding vertex map $\mathcal{V}_D$. The value of each pixel in the mask $\mathcal{S}_{\text{raw}}$ is a semantic label. The corresponding pixel in the vertex map contains the 3D coordinates of the nearest 3D point in the

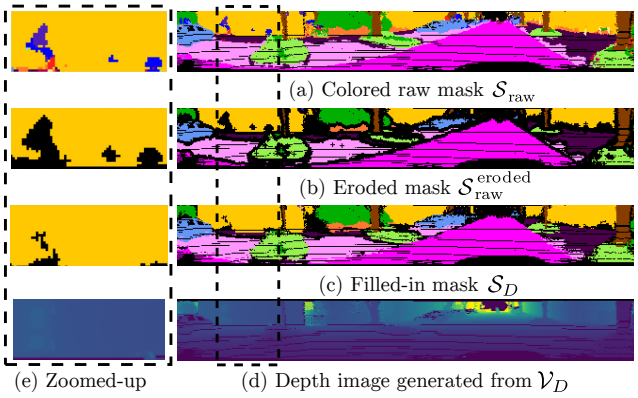


Fig. 3: Visualization of the processing steps of the proposed flood-fill algorithm. Given the (a) raw semantic map $\mathcal{S}_{\text{raw}}$, we first use an erosion to remove boundary labels and small areas of wrong labels resulting in (b) the eroded mask $\mathcal{S}_{\text{raw}}^{eroded}$. (c) We then finally fill-in eroded labels with neighboring labels to get a more consistent result $\mathcal{S}_D$. Black points represent empty pixels with label 0. (d) Shows the depth and (e) the details inside the areas with the dashed borders.

LiDAR coordinate system. The output of the approach is the refined semantic mask $\mathcal{S}_D$.

Considering that the prediction uncertainty of object boundaries is higher than that of the center of an object [16], we use the following two steps in the flood-fill. The first step is an erosion that removes pixels where the neighbors within a kernel of size d show at least one different semantic label resulting in the eroded mask $\mathcal{S}_{\text{raw}}^{\text{eroded}}$. Combining this mask with the depth information generated from the vertex map $\mathcal{V}_D$, we then fill-in the eroded mask. To this end, we set the label of an empty boundary pixel to the neighboring labeled pixels if the distances of the corresponding points are consistent, i.e., less than a threshold θ.

Fig. 3 shows the intermediate steps of this algorithm. Note that the filtered semantic map does contain less artifacts compared to the raw predictions. For instance, the wrong labels on the wall of the building are mostly corrected, which is illustrated in Fig. 3(e).

E. Filtering Dynamics using Semantics

Most existing SLAM systems rely on geometric information to represent the environment and associate observations to the map. They work well under the assumption that the environment is mostly static. However, the world is usually dynamic, especially when considering driving scenarios, and several of the traditional approaches fail to account for dynamic scene changes caused by moving objects. Therefore, moving objects can cause wrong associations between observations and the map in such situations, which must be treated carefully. Commonly, SLAM approaches use some kind of outlier rejection, either by directly filtering the observation or by building map representations that filter out changes caused by moving objects.

In our approach, we exploit the labels provided by the semantic segmentation to handle moving objects. More specifically, we filter dynamics by checking semantic consistency between the new observation $\mathcal{S}_D$ and the world model $\mathcal{S}_M$, when we update the map. If the labels are inconsistent,



(a) SuMa (b) SuMa++ (c) SuMa_nomovable

Fig. 4: Effect of the proposed filtering of dynamics. For all figures, we show the color of the corresponding label, but note that SuMa does not use the semantic information. (a) Surfels generated by SuMa; (b) our method; (c) remove all potentially moving objects.

we assume those surfels belong to an object that moved between the scans. Therefore, we add a penalty term to the computation of the stability term in the recursive Bayes filter. After several observations, we can remove the unstable surfels. In this way, we achieve a detection of dynamics and finally a removal.

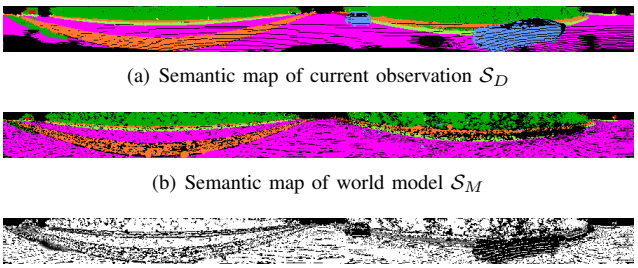
More precisely, we penalize that surfel by giving a penalty $\text{odds}(p_{\text{penalty}})$ to its stability log odds ratio l_s, which will be updated as follows:

$$\begin{aligned} l_s^{(t)} = l_s^{(t-1)} \\ + \text{odds}\left(p_{\text{stable}} \exp\left(-\frac{\alpha^2}{\sigma_\alpha^2}\right) \exp\left(-\frac{d^2}{\sigma_d^2}\right)\right) \\ - \text{odds}(p_{\text{prior}}) - \text{odds}(p_{\text{penalty}}), \end{aligned} \tag{1}$$

where $\text{odds}(p) = \log(p(1-p)^{-1})$ and p_{stable} and p_{prior} are probabilities for a stable surfel given a compatible measurement and the prior probability, respectively. The terms $\exp(-x^2\sigma^{-2})$ are used to account for noisy measurements, where α is the angle between the surfel's normal $\mathbf{n}_s$ and the normal of the measurement to be integrated, and d is the distance of the measurement in respect to the associated surfel. The measurement normal is taken from $\mathcal{N}_D$ and the correspondences from the frame-to-model ICP, see [2] for more details.

Instead of using semantic information, Pomerleau *et al.* [24] proposes a method to infer the dominant motion patterns within the map by storing the time history of velocities. In contrast to our approach, their method requires a given global map to estimate the velocities of points in the current scan. Furthermore, their robot pose estimate is assumed to be rather accurate.

In Fig. 4, we illustrate the effect of our filtering method compared to naively removing all surfels from classes corresponding to movable objects. When utilizing the naive method, surfels on parked cars are removed, even though these might be valuable features for the incremental pose estimation. With the proposed filtering, we can effectively remove the dynamic outliers and obtain a cleaner semantic world model, while keeping surfels from static objects, e.g., parked cars. These static objects are valuable information for the ICP scan registration and simply removing them can lead to failures in scan registration due to missing correspondences.



(a) Semantic map of current observation $\mathcal{S}_D$

(b) Semantic map of world model $\mathcal{S}_M$

(c) Visualized weights map

Fig. 5: Visualization of Semantic ICP: (a) semantic map $\mathcal{S}_D$ for the current laser scan, (b) corresponding semantic map $\mathcal{S}_M$ rendered from the model, (c) the weight map during the ICP. The darker the pixel, the lower is the weight of the corresponding pixel.

F. Semantic ICP

To further improve the pose estimation using the frame-to-model ICP, we also add semantic constraints to the optimization problem, which helps to reduce the influence of outliers. Our error function to minimize for ICP is given by:

$$E(\mathcal{V}_D, \mathcal{V}_M, \mathcal{N}_M) = \sum_{\mathbf{u} \in \mathcal{V}_D} w_{\mathbf{u}} \underbrace{\mathbf{n}_{\mathbf{u}}^{\top} \left(\mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} - \mathbf{v}_{\mathbf{u}} \right)^2}_{r_{\mathbf{u}}}, \quad (2)$$

where each vertex $\mathbf{u} \in \mathcal{V}_D$ is projectively associated to a reference vertex $\mathbf{v}_{\mathbf{u}} \in \mathcal{V}_M$ and its normal $\mathbf{n}_{\mathbf{u}} \in \mathcal{N}_M$ via

$$\mathbf{v}_{\mathbf{u}} = \mathcal{V}_M \left(\Pi \left(\mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} \right) \right), \quad (3)$$

$$\mathbf{n}_{\mathbf{u}} = \mathcal{N}_M \left(\Pi \left(\mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} \right) \right), \quad (4)$$

$r_{\mathbf{u}}$ and $w_{\mathbf{u}}$ are the corresponding residual and weight, respectively.

For the minimization, we use Gauss-Newton and determine increments δ by iteratively solving:

$$\delta = \left(\mathbf{J}^{\top} \mathbf{W} \mathbf{J} \right)^{-1} \mathbf{J}^{\top} \mathbf{W} \mathbf{r}, \quad (5)$$

where $\mathbf{W} \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing weights $w_{\mathbf{u}}$ for each residual $r_{\mathbf{u}}$, $\mathbf{r} \in \mathbb{R}^n$ is the stacked residual vector, and $\mathbf{J} \in \mathbb{R}^{n \times 6}$ the Jacobian of $\mathbf{r}$ with respect to the increment δ. Besides the hard association and weighting by a Huber norm, we add extra constraints from higher level semantic scene understanding to weight the residuals. In this way, we can combine semantics with geometric information to make the ICP process more robust to outliers.

Within ICP, we compute the weight $w_{\mathbf{u}}^{(k)}$ for the residual $r_{\mathbf{u}}^{(k)}$ in iteration k as follows:

$$w_{\mathbf{u}}^{(k)} = \rho_{\text{Huber}} \left(r_{\mathbf{u}}^{(k)} \right) C_{\text{semantic}}(\mathcal{S}_D(\mathbf{u}), \mathcal{S}_M(\mathbf{u})) \, \mathbb{I}\left\{ l_s^{(k)} \geq l_{stable} \right\}, \quad (6)$$

where $\rho_{\text{Huber}}(r)$ corresponds to the Huber norm, given by:

$$\rho_{\text{Huber}}(r) = \begin{cases} 1 & \text{, if } |r| < \delta \\ \delta |r|^{-1} & \text{, otherwise.} \end{cases}. \quad (7)$$

For semantic compatibility $C_{\text{semantic}}\big((y_{\mathbf{u}}, P_{\mathbf{u}}), (y_{v_{\mathbf{u}}}, P_{v_{\mathbf{u}}})\big)$, the term is defined as:

$$C_{\text{semantic}}(\cdot, \cdot) = \begin{cases} P(y_{\mathbf{u}}|\mathbf{u}) & \text{, if } y_{\mathbf{u}} = y_{v_{\mathbf{u}}} \\ 1 - P(y_{\mathbf{u}}|\mathbf{u}) & \text{, otherwise.} \end{cases}, \quad (8)$$

which is using the certainty of the predicted label to weight the residual. By $\mathbb{I}\{a\}$, we denote the indicator function that is 1 if the argument a is true, and 0 otherwise.

Fig. 5 shows the weighting for a highway scene with two moving cars visible in the scan, see Fig. 5(a). Note that our filtering of dynamics using the semantics, as described in Sec. III-E, removed the moving cars from the map, see Fig. 5(b). Therefore, we can also see a low weight corresponding to lower intensity in Fig. 5(c), since the classes of the observation and the map disagree.

IV. Experimental Evaluation

Our experimental evaluation is designed to support our main claims that we are (i) able to accurately map even in situations with a considerable amount of moving objects and we are (ii) able to achieve better performance than simply removing possibly moving objects in general environments, including urban, countryside, and highway scenes.

To this end, we evaluate our approach using data from the KITTI Vision Benchmark [11], where we use the provided point clouds generated by a Velodyne HDL-64E S2 recorded at a rate of 10 Hz. To evaluate the performance of odometry, the dataset proposes to compute relative errors in respect to translation and rotation averaged over different distances between poses and averaging it. The ground truth poses are generated using pose information from an inertial navigation system and in most sequences the GPS position is referenced to a base station, which makes it quite accurate, but still often only locally consistent.

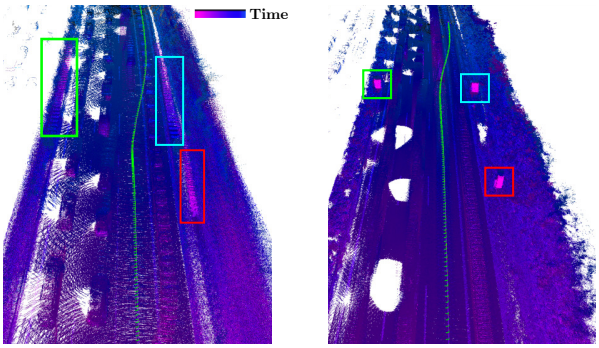
In the following, we compare our proposed approach (denoted by SuMa++) against the original surfel-based mapping (denoted by SuMa), and SuMa with the naive approach of removing all movable classes (cars, buses, trucks, bicycles, motorcycles, other vehicles, persons, bicyclists, motorcyclist) given by the semantic segmentation (denoted by SuMa_nomovable).

The RangeNet++ for the semantic segmentation was trained using point-wise annotations [1] using all training sequences from the KITTI Odometry Benchmark, which are the labels available for training purposes. This includes sequences 00 to 10, except for sequence 08 which is left out for validation.

We tested our approach on an Intel Xeon(R) W-2123 with 8 cores @3.60 GHz with 16 GB RAM, and an Nvidia Quadro P4000 with 8 GB RAM. The RangeNet++ needs on average 75 ms to generate point-wise labels for each scan and the surfel-mapping needs on average 48 ms, but we need at most 190 ms to integrate loop closures in some situations (on sequence 00 of the training set with multiple loop closures).

A. KITTI Road Sequences

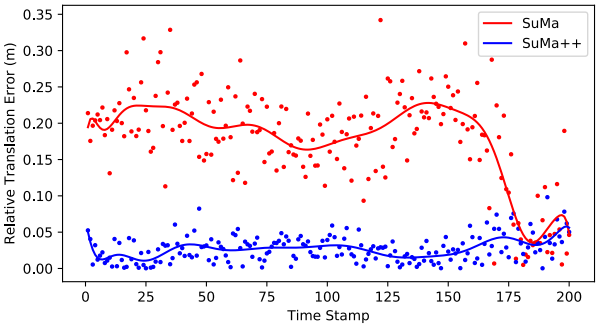
The first experiment is designed to show that our approach is able to generate consistent maps even in situations with many moving objects. We show results on sequences from the road category of the raw data of the KITTI Vision Benchmark. Note that these sequences are not part of the odometry benchmark, and therefore no labels are provided



(a) SuMa (b) SuMa++



(c) Corresponding front-view camera image



(d) Relative translation error plot for each time step

Fig. 6: Qualitative results. (a) SuMa without semantics fails to correctly estimate the motion of the sensor due to the consistent movement of cars in the vicinity of the sensor. The frame-to-model ICP locks to consistently moving cars leading to the map inconsistencies, highlighted by rectangles. (b) By incorporating semantics, we are able to correctly estimate the sensor's movement and therefore get a more consistent map of the environment and a better estimate of sensor pose via ICP. The color of the 3D points refers to the timestamp when the point has been recorded for the first time. (c) Corresponding front-view camera image, where we highlight the traffic signs. (d) Corresponding relative translation error plot for each time step. The dots are the calculated relative translational errors in each time stamp and the curves are polynomial fitting results of those dots.

for the semantic segmentation, meaning that our network learned to infer the semantic classes of road driving scenes, and it is not simply memorizing. These sequences, especially the highway sequences, are challenging for SLAM methods, since here most of the objects are moving cars. Moreover, there are only sparse distinct features on the side of the road, like traffic signs or poles. Building corners or other more distinctive features are not available to guide the registration process. In such situations, wrong correspondences on consistently moving outliers (like cars in a traffic jam) often lead to wrongly estimated pose changes and consequently

TABLE I: Results on KITTI Road dataset

Sequence	Environment	Approach		
		SuMa	SuMa_nomovable	SuMa++
30	country	0.38/0.96	0.39/0.97	**0.38/0.90**
31	country	1.54/2.02	1.66/2.13	**1.19/2.02**
32	country	1.38/1.70	1.63/1.76	**1.00/1.57**
33	highway	**1.61/1.79**	1.72/1.80	1.67/1.87
34	highway	0.79/1.17	0.70/1.14	**0.60/1.09**
35	highway	5.11/26.8	3.20/1.22	**2.90/1.11**
36	highway	0.93/1.31	0.95/**1.30**	**0.93**/1.40
37	country	0.65/1.51	0.62/**1.36**	**0.60**/1.48
38	highway	1.07/1.66	1.04/1.46	**0.89/1.42**
39	country	0.46/1.04	0.47/**0.98**	**0.44**/1.05
40	country	1.09/18.0	0.79/**1.92**	**0.75**/1.95
41	highway	1.24/15.6	**0.92/1.46**	1.14/1.67
Average		1.35/6.13	1.17/1.46	**1.04/1.46**

Relative errors averaged over trajectories of 5 to 400 m length: relative rotational error in degrees per 100 m / relative translational error in %. Bold numbers indicate top performance for laser-based approaches.
We rename the KITTI road raw dataset '2011_09_26_drive_0015_sync'-'2011_10_03_drive_0047_sync' into sequences 30-41.

inconsistencies in the generated map.

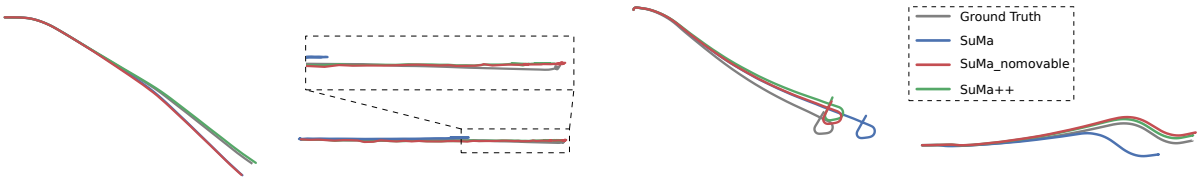
Fig. 6 shows an example generated with SuMa and the proposed SuMa++. In the case of the purely geometric approach, we clearly see that the pose cannot be correctly estimated, since the highlighted traffic signs show up at different locations leading to large inconsistencies. With our proposed approach, where we are able to correctly filter the moving cars, we instead generate a consistent map as evident by the highlighted consistently mapped traffic signs. We also plot the relative translational errors of odometry results of both SuMa and SuMa++ in this example. The dots represent the relative translational errors in each timestamp and the curves are polynomial fitting results given the dots. It shows that SuMa++ achieves more accurate pose estimates in such a challenging environments with many outliers caused by moving objects.

Tab. I shows the relative translational and relative rotational error and Fig. 7 shows the corresponding trajectories for different methods tested on this part of the dataset. Generally, we see that our proposed approach, SuMa++, generates more consistent trajectories and achieves in most cases a lower translational error than SuMa. Compared to the baseline of just removing all possibly moving objects, SuMa_nomovable, we see very similar performance compared to SuMa++. This confirms that the major reason for the worse performance of SuMa in such cases is the inconsistencies caused by actually moving objects. However, we will show in the next experiments that removing all potentially moving objects can also have negative effects on the pose estimation performance in urban environments.

B. KITTI Odometry Benchmark

The second experiment is designed to show that our methods performs superior compared to simply removing certain semantic classes from the observations. This evaluation is performed on the KITTI Odometry Benchmark.

Tab. II shows the relative translational and relative rotational errors. IMLS-SLAM [8] and LOAM [41] are state-of-the-art LiDAR-based SLAM approaches. In most sequences, we can see similar performance of SuMa++ compared to



(a) Sequence 32 (b) Sequence 35 (c) Sequence 40 (d) Sequence 41

Fig. 7: Trajectories of different methods test on KITTI road dataset.

TABLE II: Results on KITTI Odometry (training)

Approach	Sequence 00* urban	01 highway	02* urban	03 country	04 country	05* country	06* urban	07* urban	08* urban	09* urban	10 country	Average
SuMa	0.23/0.68	0.54/1.70	0.48/1.20	0.50/0.74	0.27/0.44	0.20/0.43	0.3/0.54	0.54/0.74	0.38/1.20	0.22/0.62	0.32/0.72	0.36/0.83
SuMa_nomovable	22.0/58.0	0.57/1.70	25.0/63.0	0.45/0.67	0.26/0.37	14.0/36.0	0.22/0.47	0.21/0.34	13.0/32.0	13.0/45.0	12.0/19.0	23.3/9.24
SuMa++	0.22/0.64	0.46/1.60	0.37/1.00	0.46/**0.67**	0.26/0.37	0.20/0.40	0.21/0.46	0.19/0.34	0.35/1.10	0.23/**0.47**	0.28/0.66	0.29/0.70
IMLS-SLAM [8]	-/**0.50**	-/**0.82**	-/**0.53**	-/0.68	-/**0.33**	-/**0.32**	-/**0.33**	-/**0.33**	-/**0.80**	-/0.55	-/**0.53**	-/**0.55**
LOAM [41]	-/0.78	-/1.43	-/0.92	-/0.86	-/0.71	-/0.57	-/0.65	-/0.63	-/1.12	-/0.77	-/0.79	-/0.84

Relative errors averaged over trajectories of 100 to 800 m length: relative rotational error in degrees per 100 m / relative translational error in %.
Sequences marked with an asterisk contain loop closures. Bold numbers indicate best performance in terms of translational error.

the state-of-the-art. More interestingly, the baseline method SuMa_nomovable diverges, particularly in urban scenes.

This might be counter-intuitive since these environments contain a considerable amount of man-made structures and other more distinctive features. But there are two reasons contributing to this worse performance that become clear when one looks at the results and the configuration of the scenes where mapping errors occur. First, even though we try to improve the results of the semantic segmentation, there are wrong predictions that lead to a removal of surfels in the map that are actually static. Second, the removal of parked cars is problems as these are good and distinctive features for aligning scans. Both effects contribute to making the surfel map sparser. This is even more critical as parked cars are the only distinctive or reliable features. In conclusion, the simple removal of certain classes is at least in our situation sub-optimal and can lead to worse performance.

To evaluate the performance of our approach in unseen trajectories, we uploaded our results for server-side evaluation on unknown KITTI test sequences so that no parameter tuning on the test set is possible. Thus, this serves as a good proxy for the real-world performance of our approach. In the test set, we achieved an average rotational error of 0.0032 deg/m and an average translational error of 1.06%, which is an improvement in terms of translational error, when compared to 0.0032 deg/m and 1.39% of the original SuMa.

C. Discussion

During the map updating process, we only penalize surfels of dynamics of movable objects, which means we do not penalize semantically static objects, e.g. vegetation, even though sometimes leaves of vegetation change and the appearance of vegetation changes with the viewpoint due to laser beams that only get reflected from certain viewpoints. Our motivation for this is that they can also serve as good landmarks, e.g., the trunk of a tree is static and a good feature for pose estimation. Furthermore, the original geometric-based outlier rejection mechanism employing the Huber norm often down-weights such parts.

There is an obvious limitation of our method: we cannot filter out dynamic objects in the first observation. Once there is a large number of moving objects in the first scan, our method will fail because we cannot estimate a proper initial velocity or pose. We solve this problem by removing all potentially movable object classes in the initialization period. However, a more robust method would be to backtrack changes due to the change in the observed moving state and thus update the map retrospectively.

Lastly, the results of our second experiment shows convincingly that blindly removing a certain set of classes can deteriorate the localization accuracy, but potentially moving objects still might be removed from a long-term representation of a map to also allow for a representation of otherwise occluded parts of the environment that might be visible at a different point of time.

V. Conclusion

In this paper, we presented a novel approach to build semantic maps enabled by a laser-based semantic segmentation of the point cloud not requiring any camera data. We exploit this information to improve pose estimation accuracy in otherwise ambiguous and challenging situations. In particular, our method exploits semantic consistencies between scans and the map to filter out dynamic objects and provide higher-level constraints during the ICP process. This allows us to successfully combine semantic and geometric information based solely on three-dimensional laser range scans to achieve considerably better pose estimation accuracy than the pure geometric approach. We evaluated our approach on the KITTI Vision Benchmark dataset showing the advantages of our approach in comparison to purely geometric approaches. Despite these encouraging results, there are several avenues

for future research on semantic mapping. In future work, we plan to investigate the usage of semantics for loop closure detection and the estimation of more fine-grained semantic information, such as lane structure or road type.

REFERENCES

[1] 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.

[2] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[3] B. Bescos, J.M. Fcil, J. Civera, and J. Neira. DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):4076–4083, 2018.

[4] S. Bowman, N. Atanasov, K. Daniilidis, and G.J. Pappas. Probabilistic Data Association for Semantic SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[5] N. Brasch, A. Bozic, J. Lallemand, and F. Tombari. Semantic Monocular SLAM for Highly Dynamic Environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 393–400, 2018.

[6] 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*, 2(3):194–220, 2017.

[7] 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: Towards the Robust-Perception Age. *IEEE Trans. on Robotics (TRO)*, 32:1309–1332, 2016.

[8] J. Deschaud. Imls-slam: scan-to-model matching based on 3d data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[9] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena. SegMap: 3D Segment Mapping using Data-Driven Descriptors. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[10] P. Ganti and S.L. Waslander. Visual slam with network uncertainty informed feature selection. *arXiv preprint*, 2018.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[12] J. Jeong, T. S. Yoon, and J. B. Park. Multimodal Sensor-Based Semantic 3D Mapping for a Large-Scale Environment. *Expert Systems with Applications*, 105:1–10, 2018.

[13] J. Jeong, T.S. Yoon, and J.B. Park. Towards a Meaningful 3D Map Using a 3D Lidar and a Camera. *Sensors*, 18(8), 2018.

[14] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. A Navigation System for Robots Operating in Crowded Urban Environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, Karlsruhe, Germany, 2013.

[15] P. Li, T. Qin, and S. Shen. Stereo Vision-based Semantic 3D Object and Ego-motion Tracking for Autonomous Driving. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2018.

[16] X. Li, Z. Liu, P. Luo, C.C. Loy, and X. Tang. Not All Pixels Are Equal: Difficulty-Aware Semantic Segmentation via Deep Layer Cascade. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 641–656, 2018.

[18] K.N. Lianos, J.L. Schonberger, M. Pollefeys, and T. Sattler. VSO: Visual Semantic Odometry. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 234–250, 2018.

[19] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric Object-Level SLAM. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, pages 32–41, 2018.

[20] J. McCormac, A. Handa, A.J. Davison, and S. Leutenegger. SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[21] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[22] E. Palazzolo, J. Behley, P. Lottes, P. Giguere, and C. Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[23] S.A. Parkison, L. Gan, M.G. Jadidi, and R.M. Eustice. Semantic Iterative Closest Point through Expectation-Maximization. In *Proc. of British Machine Vision Conference (BMVC)*, page 280, 2018.

[24] F. Pomerleau, P. Krüsiand, F. Colas, P. Furgale, and R. Siegwart. Long-term 3d map maintenance in dynamic environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2014.

[25] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint*, 2018.

[26] M. Rünz, M. Buffier, and L. Agapito. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects. In *Proc. of the Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20, 2018.

[27] R.F. Salas-Moreno, R.A. Newcombe, H. Strasdat, P.H.J. Kelly, and A.J. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1352–1359, 2013.

[28] C. Stachniss, J. Leonard, and S. Thrun. *Springer Handbook of Robotics, 2nd edition*, chapter Chapt. 46: Simultaneous Localization and Mapping. Springer Verlag, 2016.

[29] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett. Recurrent-OctoMap: Learning State-Based Map Refinement for Long-Term Semantic Mapping With 3-D-Lidar Data. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):3749–3756, 2018.

[30] N. Sünderhauf, T.T. Pham, Y. Latif, M. Milford, and I. Reid. Meaningful maps with object-oriented semantic mapping. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5079–5085, 2017.

[31] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 6565–6574, 2017.

[32] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[33] V. Vineet, O. Miksik, M. Lidegaard, M. Niessner, S. Golodetz, V. Prisacariu, O. Kahler, D. Murray, S. Izadi, P. Perez, and P. Torr. Incremental Dense Semantic Stereo Fusion for Large-Scale Semantic Scene Reconstruction. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[34] O. Vysotska and C. Stachniss. Lazy Data Association for Image Sequences Matching under Substantial Appearance Changes. *IEEE Robotics and Automation Letters (RA-L)*, 2016.

[35] J. Wang and J. Kim. Semantic segmentation of urban scenes with a location prior map using lidar measurements. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[36] 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 *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1887–1893, 2018.

[37] J. Yan, D. Chen, H. Myeong, T. Shiratori, and Y. Ma. Automatic Extraction of Moving Objects from Image and LIDAR Sequences. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2014.

[38] S. Yang, Y. Huang, and S. Scherer. Semantic 3D occupancy mapping through efficient high order CRFs. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[39] C. Yu, Z. Liu, X. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei. DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1168–1174, 2018.

[40] A. Zaganidis, L. Sun, T. Duckett, and G. Cielniak. Integrating Deep Semantic Segmentation Into 3-D Point Cloud Registration. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):2942–2949, 2018.

[41] J. Zhang and S. Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41:401–416, 2017.

[Q] A. Milioto, I. Vizzo, J. Behley, *et al.*, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019. DOI: 10.1109/IROS40897.2019.8967762

2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

RangeNet++: Fast and Accurate LiDAR Semantic Segmentation

Andres Milioto Ignacio Vizzo Jens Behley Cyrill Stachniss

Abstract— **Perception in autonomous vehicles is often carried out through a suite of different sensing modalities. Given the massive amount of openly available labeled RGB data and the advent of high-quality deep learning algorithms for image-based recognition, high-level semantic perception tasks are pre-dominantly solved using high-resolution cameras. As a result of that, other sensor modalities potentially useful for this task are often ignored. In this paper, we push the state of the art in LiDAR-only semantic segmentation forward in order to provide another independent source of semantic information to the vehicle. Our approach can accurately perform full semantic segmentation of LiDAR point clouds at sensor frame rate. We exploit range images as an intermediate representation in combination with a Convolutional Neural Network (CNN) exploiting the rotating LiDAR sensor model. To obtain accurate results, we propose a novel post-processing algorithm that deals with problems arising from this intermediate representation such as discretization errors and blurry CNN outputs. We implemented and thoroughly evaluated our approach including several comparisons to the state of the art. Our experiments show that our approach outperforms state-of-the-art approaches, while still running online on a single embedded GPU. The code can be accessed at `https://github.com/PRBonn/lidar-bonnetal`.**

I. Introduction

Semantic scene understanding is one of the key building blocks of autonomous robots working in dynamic, real-world environments. To achieve the required scene understanding, robots are often equipped with multiple sensors that allow them to leverage the strengths of each modality. Combining multiple complementary sensing modalities allows for covering the shortcomings of individual sensors such as cameras, laser scanners, or radars. This is particularly critical in the context of autonomous driving, where a failure of one modality can have lethal or significant monetary consequences in case it is not properly covered by another redundant sensor.

An important tasks in semantic scene understanding is the task of semantic segmentation. Semantic segmentation assigns a class label to each data point in the input modality, i.e., to a pixel in case of a camera or to a 3D point obtained by a LiDAR. In this paper, we explicitly address semantic segmentation for rotating 3D LiDARs such as the commonly used Velodyne scanners. Unfortunately, the majority of state-of-the-art methods currently available for semantic segmentation on LiDAR data either don't have enough representational capacity to tackle the task, or are computationally too expensive to operate at frame-rate on a mobile GPU. This makes them not suitable to aid the task of supporting autonomous vehicles, and addressing these issues is the aim of this work.

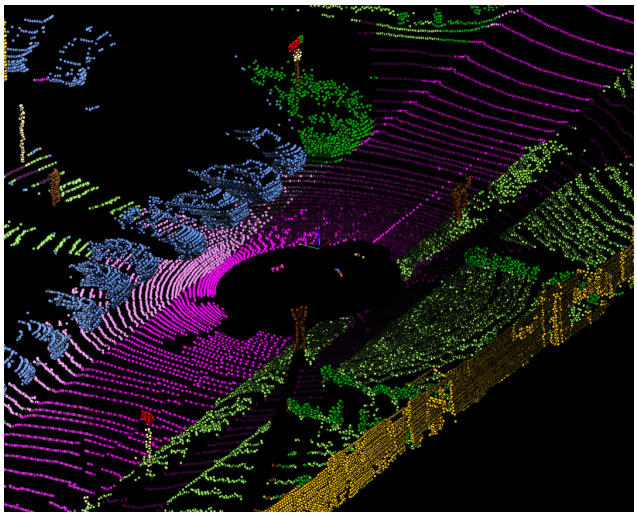


Fig. 1: Velodyne HDL-64E laser scan from KITTI dataset [7] with semantic information from RangeNet++. Best viewed in color, each color represents a different semantic class.

The main contribution of this paper is a new method for accurate, fast, LiDAR-only semantic segmentation. We achieve this by operating on a spherical projection of the input point cloud, i.e., a 2D image representation, similar to a range image, and therefore exploit the way the points are detected by a rotating LiDAR sensor. Our method infers the full semantic segmentation for each pixel of the image using any CNN as a backbone. This yields an efficient approach but can lead to issues caused by discretization or blurry CNN outputs. We effectively resolve these issues via a reconstruction of the original point with semantics without discarding any points from the original point cloud, regardless of the used resolution of the image-based CNN. This post-processing step, which also runs online, operates on the image representation and is tailored towards efficiency. We can calculate nearest neighbors in constant time for each point and exploit GPU-based calculations. This allows us to infer full semantic segmentation of LiDAR point clouds accurately and faster than the frame rate of the sensor. Since the approach runs with any range image-based CNN backbone, we call it RangeNet++. See Fig. 1 for an example.

In sum, we make three key claims: Our approach is able to (i) accurately perform semantic segmentation of LiDAR-only point clouds, surpassing the state of the art significantly, (ii) infer semantic labels for the complete original point

All authors are with the University of Bonn, Germany. This work has partly been supported by the German Research Foundation under Germany's Excellence Strategy, EXC-2070 - 390732324 (PhenoRob) as well as grant number BE 5996/1-1, and by NVIDIA Corporation.

cloud, avoiding to discard points regardless of the level of discretization used in the CNN, and (iii) work at the frame rate of a Velodyne scanner on an embedded computer that can easily fit in robots or in a vehicle.

II. Related Work

Semantic segmentation for autonomous driving using images made an immense progress in recent years due to the advent of deep learning and the availability of increasingly large-scale datasets for the task, such as CamVid [2], Cityscapes [4], or Mapillary [12]. Together, this enables the generation of complex deep neural network architectures with millions of parameters achieving high-quality results. Prominent examples are Deeplab V3 [3] and PSPNet [23].

Despite their impressive results, these architectures are too computationally expensive to run in real-time on an autonomous system, which is a must for autonomous navigation exploiting semantic cues. This spawned the creation of more efficient approaches such as Bonnet [11], ENet [13], ERFNet [17], and Mobilenets V2 [18], which leverage the law of diminishing returns to find the best trade-off between runtime, the number of parameters, and accuracy. These, however, are designed for images and not for LiDAR scans.

Transferring these results to LiDAR data has, so far, been hindered by two factors: (i) the lack of publicly available large-scale datasets for the task of semantic segmentation in autonomous driving and (ii) how prohibitively expensive to run most LiDAR semantic segmentation models are.

To tackle the problem of the lack of data, Wu *et al.* [21], [22] used the provided bounding box of the KITTI dataset [7]. They also leveraged simulation to generate realistic looking scans from a game engine. We have released the first large-scale dataset with full semantic segmentation of LiDAR scans [1], in which all scans of the KITTI odometry dataset [7] were densely annotated, i.e., over $43\,000$ scans, with over 3.5 billion annotated points. Without the data-starvation barrier, this paper investigates which of the current state-of-the-art algorithms can be exploited and adapted for point cloud in the autonomous driving context.

Leveraging large datasets for other contexts [5], [8], several deep learning-based methods for 3D semantic segmentation were recently developed, such as PointNet [14], PointNet++ [15], TangentConvolutions [20], SPLATNet [19], SuperPointGraph [10], and SqueezeSeg [21], [22].

One of the problems of dealing with point cloud data directly is the lack of a proper ordering, which makes learning order-invariant feature extractors extremely challenging. Qi *et al.* [14], [15] use as inputs the raw, un-orderered point clouds and apply symmetrical operators that are able to deal with this ordering problem. For this purpose, max pooling is used by PointNet [14] to combine the features and generate permutation-invariant feature extractors. This, however, is a limiting factor for PointNet, causing it to lose the ability to capture spatial relationships between features. This limits its applicability to complex scenes. PointNet++ [15] tackles this problem by using a hierarchical approach for feature extraction. By exploiting individual PointNets in a local vicinity, it captures short-range dependencies and then applies this concept hierarchically to capture global dependencies.

Tatarchencko *et al.* [20] take a different approach to handle unstructured point clouds. They propose TangentConvolutions that apply CNNs directly on surfaces, which can only be achieved if neighboring points are sampled from the same surface. In this case, the authors can define a tangent convolution as a planar convolution that is applied to the projection of the surface at each point. This assumption is, however, violated in case of a rotating LiDAR and the generated distance-dependent sparsity of the point cloud.

Su *et al.* [19] approach the representational problem differently in SPLATNet, by projecting the points in a high-dimensional sparse lattice. However, this approach does not scale well neither in terms of computation nor in memory consumption. To alleviate this, bilateral convolutions [9] allow them to apply theses operators exclusively on occupied sectors of the lattice.

Landrieu *et al.* [10] manage to summarize the local relationships in a similar fashion to PointNets by defining a SuperPoint Graph. This is achieved by creating so-called SuperPoints, which are locally coherent, geometrically homogeneous groups of points that get embedded by a PointNet. They create a graph of SuperPoints that is an augmented version of the original point cloud, and train a graph convolutional network to encode the global relationships.

In the case of rotating LiDAR segmentation segmentation, the number of points per scan is in the order of 10^5. This scale prevents all of these aforementioned methods from running in real-time, limiting their applicability in autonomous driving. In contrast, we propose a system that provides accurate semantic segmentation results, while still running at frame-rate of the sensor.

Leading the charge in online processing, SqueezeSeg and SqueezeSegV2 [21], [22], by Wu *et al.*, also use a spherical projection of the point cloud enabling the usage of 2D convolutions. Furthermore, a light-weight fully convolutional semantic segmentation is applied along with a conditional random field (CRF) to smooth the results. The last step is an un-discretization of the points from the range image back into the 3D world. Both are capable of running faster than the sensor rate, i.e., $10\,\text{Hz}$, and we use them as the basis of our approach.

Several limitations need to be addressed in order to provide full semantic segmentation with this framework. First, the projection needs to be extended to include the full LiDAR scan, since the SqueezeSeg framework only uses the frontal 90 degrees of the scan, where the objects of the original KITTI dataset labels are annotated by bounding boxes. Second, the SqueezeNet backbone is not descriptive enough to infer all the 19 semantic classes provided by our dataset [1]. Third, we replace the CRF, which operates in the image domain by an efficient, GPU-based nearest neighbor search acting directly on the full, un-ordered point cloud. This last step enables the retrieval of labels for all points in the cloud, even if they are not directly represented in the range image, regardless of the used resolution.

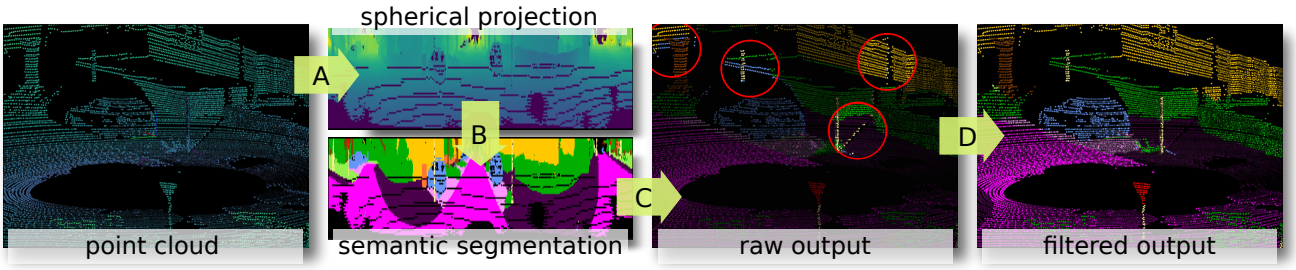


Fig. 2: Block diagram of the approach. Each of the arrows corresponds to one of our modules.

We propose a novel approach inspired by projection-based methods which allow the usage of planar convolutions and overcomes its drawbacks. Our method accurately segments entire LiDAR scans at or faster than the frame rate of the sensor (around 10 Hz), uses range images and 2D CNNs as a proxy, and deals properly with the discretization errors that need to be addressed after re-projecting the results to the 3D point cloud.

III. Our Approach

The goal of our approach is to achieve accurate and fast semantic segmentation of point clouds, in order to enable autonomous machines to make decisions in a timely manner. To achieve this segmentation, we propose a projection-based 2D CNN processing of the input point clouds and utilize a range image representation of each laser scan to perform the semantic inference. We use in the following the term range image for the spherical projection of the point cloud, but each pixel, which corresponds to a horizontal and vertical direction, can store more than only a range value. The projection is followed by a fast, GPU-based, k-Nearest-Neighbor (kNN) search over the entire point cloud, which allows us to recover semantic labels for the entire input cloud. This is particularly critical when using small resolution range images, since the projection would otherwise lead to a loss of information.

Our approach is therefore divided into four steps, depicted in Fig. 2. These four steps are discussed in detail in the following subsections: (A) a transformation of the input point cloud into a range image representation, (B) a 2D fully convolutional semantic segmentation, (C) a semantic transfer from 2D to 3D that recovers all points from the original point cloud, regardless of the used range image discretization, and (D) an efficient range image based 3D post-processing to clean the point cloud from undesired discretization and inference artifacts, using a fast, GPU-based kNN-search operating on all points.

A. Range Image Point Cloud Proxy Representation

Several LiDAR sensors, such as the Velodyne sensor represent the raw input data in a range-image-like fashion. Each column represents the range measured by an array of laser range-finders at one point in time, and each row represents different turning positions for each of those range-finders, which are fired at a constant rate. However, in a vehicle moving at high speeds this rotation does not happen fast enough to ignore the skewing generated by this sort of "rolling shutter" behavior. To obtain a more geometrically consistent representation of the environment for each scan, we must consider the vehicle motion, resulting in a point-cloud which no longer contains range measurements for each pixel, but contains multiple measurements for some others. In order to obtain an accurate semantic segmentation of the full LiDAR point cloud, our first step is to convert each de-skewed point cloud into a range representation. For this, we convert each point $\mathbf{p}_i = (x, y, z)$ via a mapping $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates and finally to image coordinates, as defined by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y,x)\,\pi^{-1}\right] & w \\ \left[1 - \left(\arcsin(z\,r^{-1}) + \mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right] & h \end{pmatrix}, \quad (1)$$

where (u, v) are said image coordinates, (h, w) are the height and width of the desired range image representation, $\mathrm{f} = \mathrm{f}_{\mathrm{up}} + \mathrm{f}_{\mathrm{down}}$ is the vertical field-of-view of the sensor, and $r = ||\mathbf{p}_i||_2$ is the range of each point. This procedure results in a list of (u, v) tuples containing a pair of image coordinates for each $\mathbf{p}_i$, which we use to generate our proxy representation. Using these indexes, we extract for each $\mathbf{p}_i$, its range r, its x, y, and z coordinates, and its remission, and we store them in the image, creating a $[5 \times h \times w]$ tensor. Because of the de-skewing of the scan, the assignment of each points to its corresponding (u, v) is done in a descending range order, to ensure that all points rendered in the image are in the current field of view of the sensor. We furthermore save this list of (u, v) pairs to gather and clean the semantics of the resulting point cloud, as we describe in Sec. III-C and Sec. III-D.

B. Fully Convolutional Semantic Segmentation

To obtain the semantic segmentation of this range image representation of the point cloud we use a 2D semantic segmentation CNN, which is modified to fit this particular input type and form factor. Similarly to Wu *et al.* [21], we use an encoder-decoder hour-glass-shaped architecture, which is depicted in Fig. 3. These types of deep hour-glass-shaped segmentation networks are characterized by having an encoder with significant downsampling, which allows the higher abstraction deep kernels to encode context information, while running faster than non-downsampling counterparts. In our case, this downsampling is 32 (see Fig. 3). This is later followed by a decoder module which upsamples the "feature code" extracted by the convolutional backbone

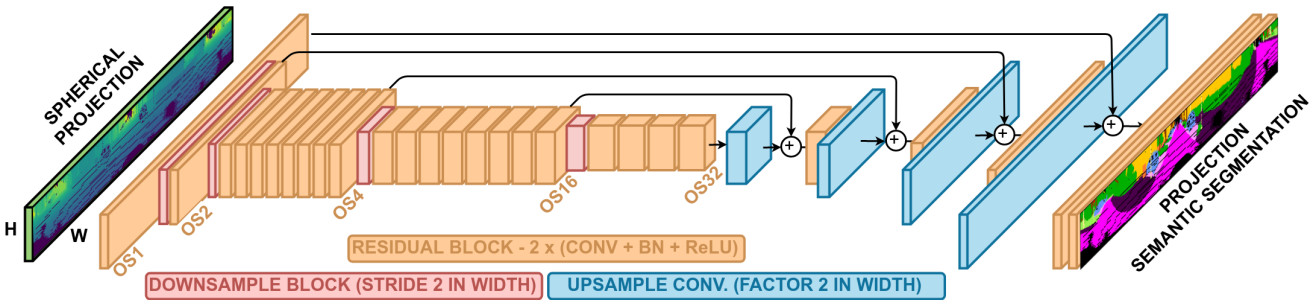


Fig. 3: Our fully convolutional semantic segmentation architecture. RangeNet53 is inspired in a Darknet53 Backbone [16].

encoder to the original image resolution, adding also convolutional layers to refine these results. At the same time, after each upsampling we also add skip connections between different levels of output stride (OS) of the encoder and sum them to the corresponding output stride feature volume in the decoder, illustrated by the black arrows, to recover some of the high-frequency edge information that gets lost during the downsampling process. After this encoding-decoding behavior, the last layer of the architecture performs a set of $[1 \times 1]$ convolutions. This generates an output volume of $[n \times h \times w]$ logits, where n is the number of classes in our data. The last layer during inference is a softmax function over the unbounded logits of the form $\hat{y}_c = \frac{e^{\text{logit}_c}}{\sum_c e^{\text{logit}_c}}$. This gives a probability distribution per pixel in the range image, where logit_c is the unbounded output in the slice corresponding to class c. During training, this network is optimized end to end using stochastic gradient descent and a weighted cross-entropy loss $\mathcal{L}$:

$$\mathcal{L} = -\sum_{c=1}^{C} w_c \, y_c \, \log\left(\hat{y}_c\right), \text{ where } w_c = \frac{1}{\log\left(f_c + \epsilon\right)} \quad (2)$$

penalizes the class c according to the inverse of its frequency f_c. This handles imbalanced data, as is the case for most datasets in semantic segmentation, e.g. the class "road" represents a significantly larger number of points in the dataset than the class "pedestrian".

To extract rich features for our encoder backbone, we define our RangeNet architectures by modifying the Darknet [16] backbone architecture in a way that makes it usable for our purposes. This backbone was designed with general image classification and object detection tasks in mind and is very descriptive, achieving state-of-the-art performance in several benchmarks for these tasks. However, it was designed to work with square aspect ratio RGB images. The first necessary modification to the backbone is to allow the first channel to take images with 5 channels. As we are dealing with a sensor that has an array of 64 vertically-placed laser range-finders producing in the order of 130 000 points per scan, this leaves us with a range image of around $w = 2048$ pixels. To retain information in vertical direction, we therefore only perform downsampling in horizontal direction. This means that in the encoder, an OS of 32 means a reduction in w of a factor of 32, but 64 pixels still remain intact in vertical direction h. To evaluate how well our post-processing recovers the original point cloud information, we analyze input sizes of $[64 \times 2048]$, $[64 \times 1024]$, and $[64 \times 512]$ in our experimental evaluation, which produce feature volumes at the end of the encoder of size $[64 \times 64]$, $[64 \times 32]$, and $[64 \times 16]$ respectively.

C. Point Cloud Reconstruction from Range Image

The common practice to map from a range image representation to a point cloud is to use the range information, along with the pixel coordinates and the sensor intrinsic calibration to realize a mapping $\Pi^* : \mathbb{R}^2 \mapsto \mathbb{R}^3$. However, since we are generating the range image from a point cloud originally, as stated in Sec. III-A, this could mean dropping a significant number of points from the original representation. This is especially critical when using smaller images in order to make the inference of the CNN faster. E.g. a scan with 130 000 points projected to a $[64 \times 512]$ range image will represent only 32 768 points, sampling the closest point in each pixel's frustum. Therefore, to infer all the original points in the semantic cloud representation, we use all the (u, v) pairs for all the $\mathbf{p}_i$ obtained during the initial rendering process and index the range image with the image coordinates that correspond to each point. This can be performed extremely fast in the GPU before the next post processing step takes place, and it results in a semantic label for each point that was present in the entire input scan, in a loss-less way.

D. Efficient Point Cloud Post-processing

Unfortunately, the benefits of the expedite semantic segmentation of LiDAR scans through 2D semantic segmentation of range images does not come without draw-backs. The encoder-decoder hour-glass-like CNNs provide blurry outputs during inference, which is also a problem for RGB and RGBD semantic segmentation. Some methods, such as [21] use a conditional random field over the predictions in the image domain after the 2D segmentation to eliminate this "bleeding" of the output labels. Using the softmax probabilities of each pixel as unary potentials for the CRF, and penalizing jumps in signal and Euclidean distance between neighboring points. Even though this helps in 2D, it does not fix the problem after the re-projection to three-dimensional space, since once the labels are projected into the original

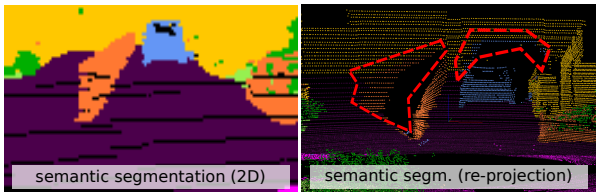


Fig. 4: Illustration of the label re-projection problem. Both the fence and the car in the range image (left) were given the proper semantic label, but during the process of sending the semantics back to the original points (right), the labels were also projected as "shadows".

point cloud, two or more points which were stored into the same range image pixel will get the same semantic label. This effect is illustrated in Fig. 4, where the labels of the inferred point cloud present shadows in objects in the background due to the blurry CNN mask, and the mentioned discretization. Moreover, if we wish to use smaller range image representations to infer the semantics, this problem becomes even stronger, resulting in shadow-like artifacts of the semantic information in objects of different classes.

To solve this problem, we propose a fast, GPU-enabled, k-nearest neighbor (kNN) search operating directly in the input point cloud. This allows us to find, for each point in the semantic point cloud, a consensus vote of the k points in the scan that are the closest to it in 3D. As it is common in kNN search, we also set a threshold for the search, which we call *cut-off*, setting the maximum allowed distance of a point considered a near neighbor. The distance metric to rank the k closest points can be the absolute differences in the range, or the Euclidean distance. Although, we also tried to use the remission as a penalization term, which did not help in our experience. From now on, we explain the algorithm considering the usage of the absolute range difference as the distance, but the Euclidean distance works analogously, albeit being slower to compute.

We explain the steps of our algorithm, described in Alg. 1, referring to the corresponding line numbers. Our approximate nearest neighbor search uses the range image representation to obtain, for each point in the $[h, w]$ range image, an $[S, S]$ window around it in the image representation, with S being a value found empirically through a grid-search in the validation set. This operation is performed through the *"im2col"* primitive, which is internally used by most parallel computing libraries to calculate a convolution, and therefore directly accessible through all deep learning frameworks. This results in a matrix of dimension $[S^2, hw]$, which contains an unwrapped version of the $[S, S]$ neighborhood in each column, and each column center contains the actual pixel's range (lines 2–4). As not all points are represented in the range image, we use the (u, v) tuples for each $\mathbf{p}_i$ obtained during the range image rendering process, and extend this representation to a matrix of dimension $[S^2, N]$, containing the range neighborhoods of all the scan points (lines 5–7). As this is done by indexing the unfolded image matrix, the centers of columns don't represent the actual range values anymore. Thus, we replace the center row of the matrix by the actual range readings for each point. The result of this is a $[S^2, N]$ matrix which contains all the range readings for the points in the center row, and in each column, its unwrapped $[S, S]$ neighborhood (lines 8–9). This is a key checkpoint in the algorithm, because it allows us to find in a quick manner a set of S^2 candidates to consider during the neighbor search for each point, in parallel. This allows our algorithm to run orders of magnitude faster than other nearest neighbor search approaches such the ones in FLANN, which work in unordered point clouds, by exploiting the arrangement of the scan points in the sensor. This key structural difference allows us to run in real-time even for large point clouds.

The following two steps are analogous to this unwrapping (lines 10–15), but instead of obtaining the ranges of the neighbor candidates, it contains their labels. This $[S^2, N]$ label matrix is later used to gather the labels for the consensus voting, once the indexes for the k neighbors are found. At this point of the algorithm, we are able to calculate the distance to the actual point for each of the S^2 candidates. If we subtract the $[1, N]$ range representation of the LiDAR scan from each row of the $[S^2, N]$ neighbor matrix, and point-wise apply the absolute value, we obtain a $[S^2, N]$ matrix where each point contains the range difference between the center of the neighborhood (which is the query point) and the surrounding points (lines 16–18). The fact that we are using a small $[S, S]$ neighborhood search around each point allows us to make the assumption that the absolute difference in the range is a good proxy for the euclidean distance, as points that are close in (u, v) coordinates will only have similar range if their actual distance in 3D space are similar. This is tested empirically in our experimental section, allowing us to make the distance calculation more efficiently, and obtaining the same result for the post-processing.

The next step is to weight the distances by an inverse Gaussian kernel, which penalizes the bigger differences in the range between points that are distant in (u, v) more. This is done by the point-wise multiplication of each column with the unwrapped kernel (lines 19–27).

After this, we need to find the k closest points for each column containing the S^2 candidates, which is a done through an *argmin* operation (lines 28–29). This allows us to get the indexes for the k points in the S neighborhood with the least weighted distance.

The last step in our search is to check which ones of those k points fit within the allowed threshold, which we call *cut-off*, and accumulate the votes from all the labels of the points within that radius. This is performed through a gather add operation, which generates a $[C, N]$ matrix, where C is the number of classes, and each row contains the number of votes in its index class (lines 30–41). A simple *argmax* operation over the columns of this matrix returns a $[1, N]$ vector which contains the clean labels for each point in the input LiDAR point cloud, and serves as the output of our approach (lines 42–43).

It is important to notice that, given the independence of all the points inside the loops in Alg. 1, each of the main components can be represented either with a parallel computing primitive or in a highly vectorized way, both

Algorithm 1: Efficient Projective Nearest Neighbor Search for Point Labels

Data: Range Image $\mathbf{I}_{\text{range}}$ of size $W \times H$,
Label Image $\mathbf{I}_{\text{label}}$ of predictions of size $W \times H$,
Ranges $\mathcal{R}$ for each point $\mathbf{p} \in \mathcal{P}$ of size N,
Image coordinates (u, v) of each point in R.
Result: Labels $\mathbf{L}_{\text{consensus}}$ for each point of size N.

Let be $[l{:}u] = \{i \mid l \leq i \leq u\}$ the range from l to u.
/* Get S^2 neighbors $\mathbf{N}'$ for each pixel */
foreach $(u, v) \in [1{:}W] \times [1{:}H]$ **do**
  foreach $(i, j) \in [1{:}S] \times [1{:}S]$ **do**
    $\mathbf{N}'[v \cdot W + u, j \cdot S + i] = \mathbf{I}_{\text{range}}[u + i, v + j]$
/* Get neighbors $\mathbf{N}$ for each point */
foreach $(u, v) \in \mathcal{C}$ **do**
  foreach $(i, j) \in [1{:}S] \times [1{:}S]$ **do**
    $\mathbf{N}[v \cdot W + u, i \cdot S + j] = \mathbf{N}'[v \cdot W + u, i \cdot S + j]$
/* Fill in real point ranges */
foreach $i \in [1{:}N]$ **do**
  $\mathbf{N}[i, \lfloor (S \cdot S - 1)/2 \rfloor] = \mathcal{R}(i)$
/* Label neighbors $\mathbf{L}'$ for each pixel */
foreach $u \in [1{:}W]$, $v \in [1{:}H]$ **do**
  foreach $(i, j) \in [1{:}S] \times [1{:}S]$ **do**
    $\mathbf{L}'[v \cdot W + u, i \cdot S + j] = \mathbf{I}_{\text{label}}[u + i, v + j]$
/* Get label neighbors $\mathbf{L}$ for each point */
foreach $(u, v) \in \mathcal{C}$ **do**
  foreach $(i, j) \in [1{:}S] \times [1{:}S]$ **do**
    $\mathbf{L}[v \cdot W + u, i \cdot S + j] = \mathbf{L}'[v \cdot W + u, i \cdot S + j]$
/* Distances to neighbors $\mathbf{D}$ for each point */
foreach $i \in n\,[1{:}N]$ **do**
  foreach $j \in n\,[1{:}S{\cdot}S]$ **do**
    $\mathbf{D}[i, j] = |\mathbf{N}[i, j] - \mathcal{R}(i)|$
/* Compute inverse Gaussian Kernel */
Let $\mathcal{N}(u \mid \mu, \sigma)$ be a Gaussian with mean μ and std. deviation σ.
foreach $(i, j) \in [1{:}S] \times [1{:}S]$ **do**
  $\mathbf{G}'[j \cdot S + i] = \mathcal{N}(i \mid 0, \sigma) \cdot \mathcal{N}(j \mid 0, \sigma)$
Let be $G_{\max} = \max\{\mathbf{G}'[i] \mid i \in [1{:}S{\cdot}S]\}$ the maximium of $\mathbf{G}'$
foreach $i \in [1{:}S{\cdot}S]$ **do**
  $\mathbf{G}[i] = 1 - G_{\max} \cdot \mathbf{G}'[i]$
/* Weight neighbors with inverse Gaussian kernel */
foreach $i \in [1{:}N]$ **do**
  foreach $j \in [1{:}S{\cdot}S]$ **do**
    $\mathbf{D}[i, j] = \mathbf{D}[i, j] \cdot \mathbf{G}[j]$
/* Find k-nearest neighbors $\mathcal{S}$ for each point */
foreach $i \in [1{:}N]$ **do**
  $\mathcal{S}[i] = \{j \mid |\{n \in [1{:}S{\cdot}S] \mid \mathbf{D}[i, n] < \mathbf{D}[i, j]\}| \leq k\}$
/* Gather votes. */
foreach $i \in [1{:}N]$ **do**
  $n = 1$
  foreach $j \in \mathcal{S}[i]$ **do**
    if $D[i, j] > \delta_{cutoff}$ **then**
      $\mathbf{L}_{\text{knn}}[i, n] = C + 1$
    else
      $\mathbf{L}_{\text{knn}}[i, n] = \mathbf{L}[i, j]$
    $n = n + 1$
/* Accumulate votes. */
foreach $i \in [1{:}N]$ **do**
  foreach $j \in [[:1] : k]$ **do**
    if $\mathbf{L}_{knn}[i, j] \leq C$ **then**
      $\mathbf{V}[i, \mathbf{L}_{\text{knn}}[i, j]] = \mathbf{V}[i, \mathbf{L}_{\text{knn}}[i, j]] + 1$
/* Find maximum consensus. */
foreach $i \in [1{:}N]$ **do**
  $\mathbf{L}_{\text{consensus}}[i] = \arg\max_c \mathbf{L}_{\text{knn}}[i, c]$

of which are directly implementable in a GPU, using off-the-shelf, high abstraction deep learning or data science frameworks.

This algorithm requires setting four different hyperparameters: (i) S which the size of the search window, (ii) k which is number of nearest neighbors, (iii) *cut-off* which is the maximum allowed range difference for the k, and (iv) σ for the inverse gaussian. The values for the hyperparameters are calculated empirically through a data-driven search in the validation set of our training data, and a brief analysis is provided in the experimental section.

IV. Experimental Evaluation

The experimental evaluation is designed to support our claims that our approach: (i) outperforms the state of the art in the task of semantic segmentation of LiDAR scans, (ii) infers the entire point cloud while recovering the high frequency information in the un-projection step, and (iii) runs online in an embedded computer at sensor frame-rate.

Dataset. We train and evaluate our approach on a large-scale dataset that provides dense point-wise annotations for the entire KITTI Odometry Benchmark [7], [1]. The dataset is comprised of over $43\,000$ scans from which over $21\,000$ from sequences 00 to 10 are available for training and the remaining scans from sequences 11 to 21 are used as test set. We used sequence 08 as validation set for hyperparameter selection and trained our approach on the remaining training sequences. Overall, the dataset provides 22 classes from which 19 classes are evaluated on the test set via our benchmark website.

Hyperparameter selection. All hyperparameters for RangeNet models are selected and evaluated on the validation set (sequence 8). For all backbone trainings, we use a learning rate of $1 \cdot 10^{-3}$, with a decay of 0.99 every epoch, and train for 150 epochs. For all CNN backbones, convergence was achieved in less than 150 epochs. For all the state-of-the-art methods, the hyperparameters were also selected on the validation set.

Metrics. To assess the labeling performance, we use the commonly applied mean Jaccard Index or mean intersection-over-union (IoU) metric, mIoU, over all classes [6] given by

$$mIoU = \frac{1}{C} \sum_{c=1}^{C} \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c + \text{FN}_c} \tag{3}$$

where TP_c, FP_c, and FN_c correspond to the number of true positive, false positive, and false negative predictions for class c and C is the number of classes.

To better asses the performance with respect to the precision of the prediction, we propose an additional evaluation metric which we call *border-IoU*. This metric is defined in the same way as the standard IoU, but only applies within the subset of points defined by an extra parameter which considers how far a point is to the self occlusion of the sensor, which is manifested in a change in the label in the range image. This metric is designed to show how much our algorithm can help the "shadow-like" wrong label projections in the semantic point clouds.

TABLE I: IoU [%] on test set (sequences 11 to 21). RangeNet21 and RangeNet53 represent the new baselines with augmented Darknet backbones (21 and 53 respectively), and the versions with (++) are treated with our fast point cloud post-processing based on range.

Approach	Size	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	mean IoU	Scans/sec
Pointnet [14]	50000pts	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0.0	61.6	15.8	35.7	1.4	41.4	12.9	31.0	4.6	17.6	2.4	3.7	14.6	2
Pointnet++ [15]		53.7	1.9	0.2	0.9	0.2	0.9	1.0	0.0	72.0	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30.0	6.0	8.9	20.1	0.1
SPGraph [10]		68.3	0.9	4.5	0.9	0.8	1.0	6.0	0.0	49.5	1.7	24.2	0.3	68.2	22.5	59.2	27.2	17.0	18.3	10.5	20.0	0.2
SPLATNet [19]		66.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	70.4	0.8	41.5	0.0	68.7	27.8	72.3	35.9	35.8	13.8	0.0	22.8	1
TangentConv [20]		86.8	1.3	12.7	11.6	10.2	17.1	20.2	0.5	82.9	15.2	61.7	9.0	82.8	44.2	75.5	42.5	55.5	30.2	22.2	35.9	0.3
SqueezeSeg [21]	64 × 2048 px	68.8	16.0	4.1	3.3	3.6	12.9	13.1	0.9	85.4	26.9	54.3	4.5	57.4	29.0	60.0	24.3	53.7	17.5	24.5	29.5	**66**
SqueezeSeg-CRF [21]		68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	04.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3	30.8	55
SqueezeSegV2 [22]		81.8	18.5	17.9	13.4	14.0	20.1	25.1	3.9	88.6	45.8	67.6	17.7	73.7	41.1	71.8	35.8	60.2	20.2	36.3	39.7	50
SqueezeSegV2-CRF [22]		82.7	21.0	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41.0	68.5	36.9	58.9	12.9	41.0	39.6	40
RangeNet21 [Ours]		85.4	26.2	26.5	18.6	15.6	31.8	33.6	4.0	91.4	57.0	74.0	26.4	81.9	52.3	77.6	48.4	63.6	36.0	50.0	47.4	20
RangeNet53 [Ours]	64 × 2048 px	86.4	24.5	32.7	25.5	22.6	36.2	33.6	4.7	**91.8**	64.8	74.6	**27.9**	84.1	55.0	78.3	50.1	64.0	38.9	52.2	49.9	13
	64 × 1024 px	84.6	20.0	25.3	24.8	17.3	27.5	27.7	7.1	90.4	51.8	72.1	22.8	80.4	50.0	75.1	46.0	62.7	33.4	43.4	45.4	25
	64 × 512 px	81.0	9.9	11.7	19.3	7.9	16.8	25.8	2.5	90.1	49.9	69.4	2.0	76.0	45.5	74.2	38.8	62.7	25.5	38.1	39.3	52
RangeNet53++ [Ours+kNN]	64 × 2048 px	**91.4**	**25.7**	**34.4**	**25.7**	**23.0**	**38.3**	**38.8**	**4.8**	**91.8**	**65.0**	**75.2**	27.8	**87.4**	**58.6**	**80.5**	**55.1**	**64.6**	**47.9**	**55.9**	**52.2**	12
	64 × 1024 px	90.3	20.6	27.1	25.2	17.6	29.6	34.2	7.1	90.4	52.3	72.7	22.8	83.9	53.3	77.7	52.5	63.7	43.8	47.2	48.0	21
	64 × 512 px	87.4	9.9	12.4	19.6	7.9	18.1	29.5	2.5	90.0	50.7	70.0	2.0	80.2	48.9	77.1	45.7	64.1	37.1	42.0	41.9	38

A. *Performance of RangeNet++ w.r.t. State-of-the-art*

The first experiment is designed to support our claim that our approach over-performs the state of the art in the task of scene semantic segmentation of LiDAR point clouds. Tab. I shows the difference between our RangeNet backbones, using 21 and 53 layers, and 7 other baseline methods. The superior performance of our RangeNet baselines, even without our cleaning, for all the input resolutions of shows that it is a solid baseline to benchmark our efficient kNN cleaning. Tab. I also shows that our method, RangeNet++, which includes our kNN post processing consistently beats its unprocessed RangeNet counterpart, showing the efficacy of our kNN search. The kNN cleaning is consistently better for all but one classes, unlike the CRF, which is a conclusion reached by the original SqueezeSeg [21] paper as well, even when the overall IoU is higher.

B. *Ablation Studies*

The second experiment shows the influence of the k and S parameters in the validation set. For each of the 4 parameters k, S, σ, and *cut-off* we chose a wide range of values and evaluated the result of post-processing the inference results of the RangeNet53 backbones for all input resolutions. Fig. 5 shows a normalized result of the IoU in the validation set for each parameter set, for various k and S and the argmax of σ and *cut-off*. The results also show that we can obtain similar results using small kernels and the absolute range difference, as a proxy for the Euclidean distance. This supports our statement that this range difference is a good proxy for the actual distance the closer the points are in the image.

C. *Post-Processing Influence*

The third experiment is designed to support our claim that our algorithm improves the reconstruction of the semantics of the entire point cloud even for smaller range image resolutions. For this, we use our *border-IoU* metric, which only considers points that are a certain number of points away from a change in label. In Fig. 6 we show

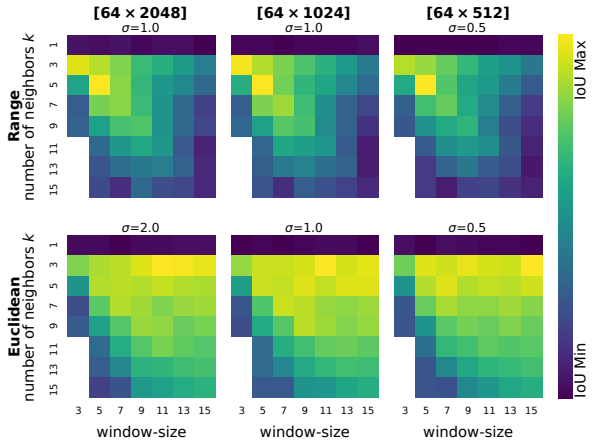


Fig. 5: Hyperparameter search post-processing for both Range (top row) and Euclidean distance (bottom row) using RangeNet53++, and different input resolutions. All experiments used *cutoff* = 1.0 m.

the value of the IoU and the value of the border IoU for different distances to border. Note that our post-processing approach doesn't only improve the IoU score by a couple of % points, but it significantly improves the border IoU score for low values of the distance to border parameter. This means that our approach is especially useful to help in cases of label "bleeding" or "shadowing" described in Sec. III-D. Another important conclusion is that there is only marginal differences between using the faster to compute range difference and the actual Euclidean distance, throughout the entire spectrum of border distances, and in the IoU, which support our statement that it is a good approximation.

D. *Runtime*

The fourth experiment is designed to support our claim that the approach can run in its totality online in a moving platform, using a single GPU. Tab. II shows the runtime for the backbone, different post-processing distance functions (for the best parameters), and the total time required. As expected, the range-based post-processing is faster to calcu-

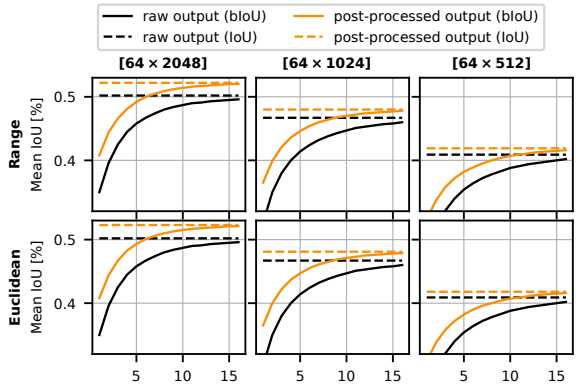


Fig. 6: Border IoU (bIoU) and IoU as a function of the distance to label change. This plot shows that our post-processing improves the IoU, and significantly improves the borderIoU, which means that it recovers blurry mask and discretization errors better.

TABLE II: Runtime of RangeNet53++.

Hardware	Resolution (px)	Processing time (ms) CNN	Range	Euclid	Total	FPS
Quadro P6000	64 × 512	19	7	11	26	38
	64 × 1024	40			47	21
	64 × 2048	75			82	12
Jetson AGX	64 × 512	45	35	52	80	13
	64 × 1024	87			122	8
	64 × 2048	153			188	5

late, since each distance calculation requires a subtraction and an absolute value, compared to 3 squares, 2 sums, and a square root. Therefore, since the difference in performance is negligible, we use the sum of our CNN backbone plus this range processing time for our total runtime, which we evaluate in two different types of hardware.

V. Conclusion

In this work, we presented a fast and accurate framework for semantic segmentation of point clouds recorded by a rotating LiDAR sensor. Our main contribution is a novel deep-learning-supported approach that exploits range images and 2D convolutions, followed by a novel, GPU-accelerated post-processing to recover consistent semantic information during inference for entire LiDAR scans. Our experimental evaluation suggests that our modified 2D deep CNN operating on range images outperforms the current state of the art in semantic segmentation of LiDAR point clouds. Moreover, our efficient, GPU-enabled post-processing can further improve on these results by recovering important boundary information lost during the de-skewing of the laser scans, the lossy discretization into a proxy representation, and the inference through an hour-glass-shaped CNN. Overall, our approach outperforms the state of the art both in accuracy and runtime, taking a step forward towards sensor redundancy for semantic segmentation for autonomous vehicles and robots.

References

[1] 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.

[2] G.J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008.

[3] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint*, 2017.

[4] M. Cordts, S. Mohamed Omran, Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] A. Dai, A.X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] M. Everingham, S. Eslami, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge a Retrospective. *International Journal on Computer Vision (IJCV)*, 111(1):98–136, 2015.

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[8] T. Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98, 2017.

[9] V. Jampani, M. Kiefel, and P.V. Gehler. Learning Sparse High Dimensional Filters: Image Filtering, Dense CRFs and Bilateral Neural Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] L. Landrieu and M. Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] A. Milioto and C. Stachniss. Bonnet: An Open-Source Training and Deployment Framework for Semantic Segmentation in Robotics using CNNs. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[12] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[13] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *arXiv preprint*, 2016.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] C.R. Qi, K. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2017.

[16] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint*, 2018.

[17] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Trans. on Intelligent Transportation Systems (ITS)*, 19(1):263–272, 2018.

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arxiv*, 2018.

[19] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M-H. Yang, and J. Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] M. Tatarchenko, J. Park, V. Koltun, and Q-Y. Zhou. Tangent Convolutions for Dense Prediction in 3D. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] 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 *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[22] 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. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[23] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *arXiv preprint*, abs/1612.01105, 2016.

[R] J. Behley and C. Stachniss, "Efficient Surfel-based SLAM using 3D Laser Range Data in Urban Environments," in *Proc. of Robotics: Science and Systems (RSS)*, 2018. DOI: 10.15607/RSS.2018.XIV.016

Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments

Jens Behley and Cyrill Stachniss
Institute of Geodesy and Geoinformation, University of Bonn, Germany
Email: {firstname.lastname}@igg.uni-bonn.de

***Abstract*—Accurate and reliable localization and mapping is a fundamental building block for most autonomous robots. For this purpose, we propose a novel, dense approach to laser-based mapping that operates on three-dimensional point clouds obtained from rotating laser sensors. We construct a surfel-based map and estimate the changes in the robot's pose by exploiting the projective data association between the current scan and a rendered model view from that surfel map. For detection and verification of a loop closure, we leverage the map representation to compose a virtual view of the map before a potential loop closure, which enables a more robust detection even with low overlap between the scan and the already mapped areas. Our approach is efficient and enables real-time capable registration. At the same time, it is able to detect loop closures and to perform map updates in an online fashion. Our experiments show that we are able to estimate globally consistent maps in large scale environments solely based on point cloud data.**

I. INTRODUCTION

Most autonomous robots, including self-driving cars, must be able to reliably localize themselves, ideally by using only their own sensors without relying on external information such as GPS or other additional infrastructure placed in the environment. There has been significant advances in vision-based [6, 7] and RGB-D-based [18, 33, 3] SLAM systems over the past few years. Most of these approaches use (semi-)dense reconstructions of the environment and exploit them for frame-to-model tracking, either by jointly optimizing the map and pose estimates or by alternating pose estimation and map building [21]. Dense approaches have a prospective advantage over feature-based and sparse approaches as they use all available information and thus do not depend on reliable feature extraction or availability of such features.

In contrast to these developments, current 3D laser-based mapping systems mainly accomplish the estimation relying on feature-based solutions [34, 35], reduced map representations [14, 13], voxel grid-based methods [16], or point sub-sampling [30], which all effectively reduce the data used for alignment. Compared to most indoor applications using RGB-D sensors, we have to tackle additional challenges in outdoor applications using 3D laser sensors, *i.e.*, (1) fast sensor movement resulting in large displacements between scans, (2) comparably sparse point clouds, and (3) large-scale environments.

In this paper, we present a dense mapping approach called Surfel-based Mapping (SuMa), which builds globally consistent maps by tracking the pose, so-called odometry, and closing loops using a pose graph as illustrated in Figure 1. To this end, we employ a surfel map to efficiently generate synthetic views for projective data association and additional loop closure detection with a verification step afterwards. Furthermore, the surfel representation enables us to efficiently represent large environments. We are able to update the map on loop closures by exploiting a pose graph optimization integrating odometry and loop closure constraints.

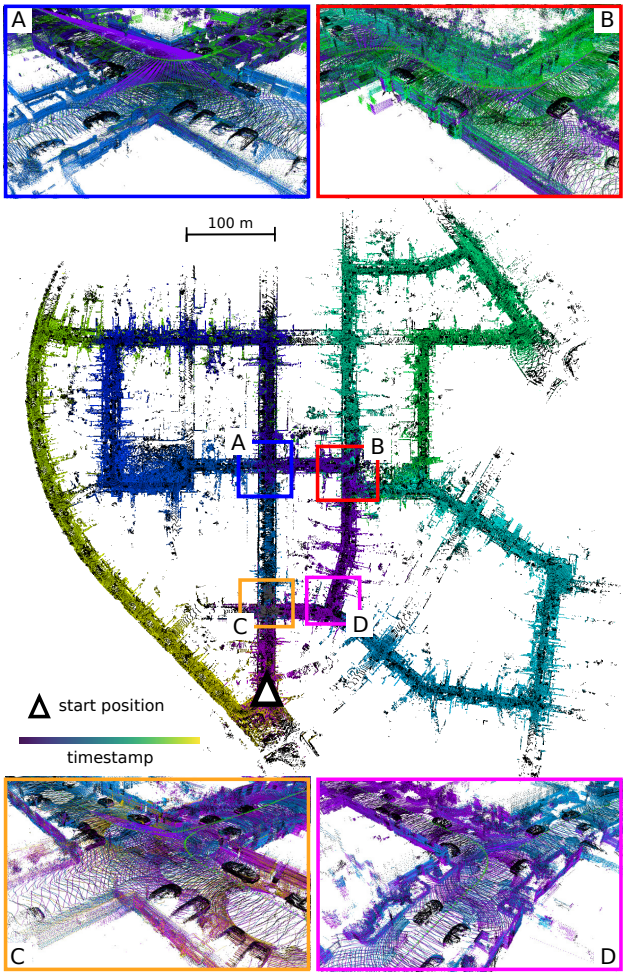


Fig. 1. Mapping result of our approach on sequence 00 of the KITTI Vision Benchmark. Shown is the complete trajectory color encoded with the timestamp of the scan ranging from purple to yellow. Also shown are some loop closures (A)-(D) found by our approach after pose graph optimization.

The key contributions of our work are as follows:

- We present a SLAM system for efficient mapping of three-dimensional laser range data resulting in globally consistent maps.
- We propose a novel map-based criterion for loop closure

detection that is able to detect loop closures even in situations with small overlap between scans.

Our approach is inspired by recent work in RGB-D SLAM [12, 33], but is designed to operate in setups with fast moving vehicles in comparably large outdoor environments. Our experimental evaluation on the KITTI Odometry Benchmark shows that our approach is on par with other laser-based approaches [34, 35], while performing 3D point cloud registration, map update, and loop closure detection including loop verification at 20 Hz on average. To the best of our knowledge, this is the first approach for 3D laser-based mapping using surfels for mapping with loop closures that produces globally consistent maps and operates at that speed. We furthermore release our implementation of the approach.[1]

II. RELATED WORK

Odometry estimation and Simultaneous Localization and Mapping (SLAM), especially with (stereo) cameras and 2D laser range scanners, is a classical topic in robotics and in the computer vision community. We acknowledge the large body of work in this field, but concentrate here on approaches based on 3D laser range data and closely related work using RGB-D sensors. For a more detailed recent overview, we refer to the article by Cadena *et al*. [1] and the references therein.

Laser-based Mapping. Laser-based odometry and mapping systems often reduce the 3D point cloud data by relying on features [34], subsampled clouds [16, 30], or voxel-based [34] as well as NDT-based map representations [27, 23, 4]. In contrast to that, we operate on all laser points and perform a registration to a surfel map at every step of the algorithm. While the approach by Moosmann and Stiller [16] also uses depth images to accelerate the computation of normals, it uses nearest neighbors in a grid-based map representation to estimate correspondences. Thus, it is inevitable to use subsampling to accelerate the processing and only $1,500$ points are retained. The Lidar Odometry and Mapping (LOAM) [34, 35] by Zhang and Singh extracts distinct features corresponding to surfaces and corners, which are then used to determine point-to-plane and point-to-line distances to a voxel grid-based map representation. To enable real-time operation, LOAM switches between frame-to-frame at 10 Hz and frame-to-model operation at 1 Hz update frequency. Like the approach by Stückler *et al*. [28], we also use surfels, but we exploit the representation to render synthetic views allowing fast data association and furthermore perform loop closure detection.

In contrast to the mentioned approaches, our method uses projective data association to estimate dense correspondences for each projected point and therefore is more robust to missing features or missing data. Unlike other approaches, it can perform frame-to-model tracking in every iteration and updates the map representation at the same time. Lastly, none of the mentioned approaches integrates loop closures in an online fashion to obtain globally consistent maps. Very recently, Park *et al*. [20] proposed a surfel-based mapping approach based on ElasticFusion by Whelan *et al*. [33] that allows optimization of continuous-time trajectories for a rotating 2D laser scanner. In contrast to this work, we use projective data association to find correspondences between the current scan and rendered model views, which is computationally more efficient than relying on nearest neighbor search.

RGB-D-based Mapping. In recent years, there has been considerable progress in the field of RGB-D-based SLAM. KinectFusion, the seminal system by Newcombe *et al*. [18], largely impacted the development in the subsequent years. In line with Newcombe *et al*.'s approach, the approach of Keller *et al*. [12] for RGB-D SLAM uses projective data association in a dense model, but relies on a surfel-based map [31] for tracking. The surfel-based representation allows to model comparably large environments without a compromise in terms of reduced reconstruction fidelity due to reduced grid resolution or a more complex implementation [32, 19]. Other recent RGB-D approaches use the surfels to extract planar surfaces [24] or perform online loop closure integration by deforming the surfel representation [33, 31].

While our method borrows some ideas for frame-to-model registration from Keller *et al*. and Whelan *et al*., we specifically designed our approach to work with rotating laser range scanners in highly dynamic environments, like busy inner-city traffic, and furthermore perform loop closure detection and verification using a map-based criterion, which enables a more reliable detection of loop closures using 3D laser range data.

Laser-based Loop Closure Detection. For loop closure detection using three-dimensional laser-range data, mainly feature-based approaches [25, 26, 22] were investigated. These approaches either use specific interest points [25, 26] to aggregate features or generally generate a global feature representation [15, 22] of a point cloud, which is then used to compute a distance between two point clouds. Often, a simple thresholding is used to identify potential loop closure candidates, which then must be verified. Only recently, the SegMatch approach by Dubé *et al*. [5] investigated an approach that matches segments extracted from a scan to find loop closures via segment-based features. A geometric test via RANSAC is used to verify a potential loop closure identified by the matching procedure.

In contrast to the aforementioned approaches, we directly determine the compatibility of the current laser scan and nearby poses using ICP. For this purpose, we propose to use a criterion based on a rendered view of the map to determine if a loop closure leads to a consistent map given the current scan. We verify this multiple times after the first detection before we actually integrate the scan into the map. Hess *et al*. [10] propose a similar strategy for 2D laser range data, as they also search in nearby submaps for loop closure candidates based on the current pose estimate. However, they use an exhaustive branch-and-bound search on the occupancy grid maps, which is not possible for 3D laser range data due to runtime constraints.

[1]See our project website for additional information, results, and videos: `https://jbehley.github.io/projects/surfel_mapping/`

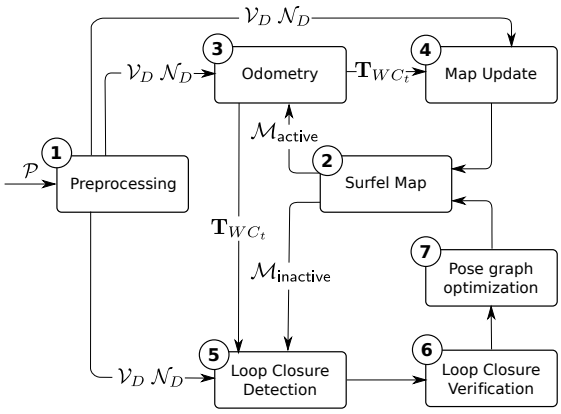


Fig. 2. Our approach. Starting with a point cloud $\mathcal{P}$, (1) a vertex map $\mathcal{V}_D$ and normal map $\mathcal{N}_D$ are generated, which are used for frame-to-model alignment (3) to our model vertex map $\mathcal{V}_M$ and normal map $\mathcal{N}_M$ rendered from $\mathcal{M}_{\text{active}}$ (2). The transformation $\mathbf{T}_{WC_t}$ is then used to update the surfel map $\mathcal{M}_{\text{active}}$ (4). Using the inactive map $\mathcal{M}_{\text{inactive}}$, we try to find a loop closure candidate (5), which then needs to be verified (6) in subsequent scans. Verified loop closures are then integrated into a pose graph and used for optimization (7) in a separate thread. The optimized poses are then used to modify the surfel map yielding a globally consistent map.

III. APPROACH

Notation. In the following, we denote the transformation of a point $\mathbf{p}_A$ in coordinate frame A to a point $\mathbf{p}_B$ in coordinate frame B by $\mathbf{T}_{BA} \in \mathbb{R}^{4\times4}$, such that $\mathbf{p}_B = \mathbf{T}_{BA}\mathbf{p}_A$. Let $\mathbf{R}_{BA} \in \mathrm{SO}(3)$ and $\mathbf{t}_{BA} \in \mathbb{R}^3$ denote the corresponding rotational and translational part of transformation $\mathbf{T}_{BA}$.

We distinguish between the observed *data* in the coordinate frame C_t at timestep t and a rendered representation, called *model*, which corresponds to our map representation at a given coordinate frame C_k, $k \in \{0, \dots, t\}$. Each data coordinate frame C_t is associated to the world frame W by a pose $\mathbf{T}_{WC_t} \in \mathbb{R}^{4\times4}$, transforming the observed point cloud into the world coordinate frame. Our aim is to determine this transformation for each point cloud via pose changes $\mathbf{T}_{C_{k-1}C_k}$ given the rendered model at $\mathbf{T}_{WC_{k-1}}$, *i.e.*,

$$\mathbf{T}_{WC_t} = \mathbf{T}_{WC_0}\mathbf{T}_{C_0C_1}\cdots\mathbf{T}_{C_{t-1}C_t}, \tag{1}$$

assuming $\mathbf{T}_{WC_0}$ to be the identity $\mathbf{Id} \in \mathbb{R}^{4\times4}$ or the extrinsic calibration of the laser in respect to a fixed reference frame.

Overview. Before we explain our approach in detail, we provide a brief overview of the processing steps, shown in Figure 2. For each point cloud $\mathcal{P} = \{\mathbf{p} \in \mathbb{R}^3\}$, we estimate the pose $\mathbf{T}_{WC_t}$ at timestep t by performing the following steps, also indicated by corresponding numbers in Figure 2:

1) First, we use a projection function $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to generate a vertex map $\mathcal{V}_D : \mathbb{R}^2 \mapsto \mathbb{R}^3$ mapping a two-dimensional image coordinate $(u, v)^T \in \mathbb{R}^2$ to a point $(x, y, z)^T \in \mathbb{R}^3$. We generate a corresponding normal map $\mathcal{N}_D : \mathbb{R}^2 \mapsto \mathbb{R}^3$ exploiting the vertex map $\mathcal{V}_D$.[2]

[2]Note that we convert points $\mathbf{p} \in \mathbb{R}^3$ and normals $\mathbf{n} \in \mathbb{R}^3$ to corresponding homogeneous coordinates before application of the affine transformation, *i.e.*, $\dot{\mathbf{p}} = (x, y, z, 1)$ and $\dot{\mathbf{n}} = (x, y, z, 0)$, but will not include this operation explicitly in the following derivations.

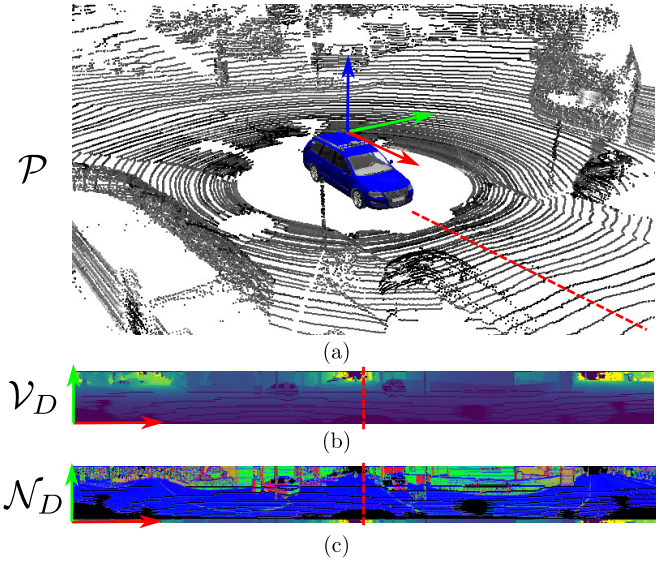


Fig. 3. Preprocessing of a point cloud generated by a rotating laser range scanner. (a) A point cloud from the KITTI Vision Benchmark, (b) the corresponding vertex map $\mathcal{V}_D$, and (c) the corresponding normal map $\mathcal{N}_D$. In the vertex map $\mathcal{V}_D$, colors ranging from purple to yellow, correspond to the distance of the point in the given pixel. Colors in the visualization of the normal map $\mathcal{N}_D$ correspond to the direction of the normal, *i.e.*, blue are normals pointing in the direction of the z-axis (upward). Also shown are the coordinate systems and the correspondence between the point cloud and the vertex/normal map indicated by a dashed line.

2) Next, we render a vertex map $\mathcal{V}_M$ and a normal map $\mathcal{N}_M$ at the last pose estimate $\mathbf{T}_{WC_{t-1}}$ from the current active surfel map $\mathcal{M}_{\text{active}}$, which contains the last Δ_{active} point clouds up to timestep $t-1$.
3) Using both the vertex map and normal map, we estimate the odometry, *i.e.*, the transformation $\mathbf{T}_{C_{t-1}C_t}$. We use ICP to align the points in $\mathcal{V}_D$ with the points of $\mathcal{V}_M$, where we exploit projective data association. This so-called frame-to-model ICP yields $\mathbf{T}_{C_{t-1}C_t}$, which is used to update the global pose $\mathbf{T}_{WC_t}$ via Equation 1.
4) With the current pose $\mathbf{T}_{WC_t}$, we update the surfel map $\mathcal{M}_{\text{active}}$ by initializing surfels for previously unseen areas, but also refining surfels of already covered areas.
5) Next, we search for a potential loop closure in the so-called inactive map $\mathcal{M}_{\text{inactive}}$ and try to align the current measurements with the map.
6) If we have found a loop closure candidate at timestep t, we try to verify it in the subsequent timesteps $t+1, \dots, t+\Delta_{\text{verification}}$, which ensures that we only add consistent loop closures to the pose graph.
7) In a separate thread, a pose graph is optimized consisting of the relative poses of the odometry and the loop closures. The optimized poses are then used for updating the surfel map.

We will now describe these steps in more detail.

A. Preprocessing

For projective data association, we first project the point cloud $\mathcal{P}$ to the vertex map $\mathcal{V}_D : \mathbb{R}^2 \mapsto \mathbb{R}^3$, where each

pixel contains the nearest 3D point. Each point $\mathbf{p}_i = (x, y, z)$ is converted via the function $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates and finally to image coordinates (u, v), *i.e.*,

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x) \cdot \pi^{-1}\right] \cdot w \\ \left[1 - \left(\arcsin(z \cdot r^{-1}) + \mathrm{f}_{\mathrm{up}}\right) \mathrm{f}^{-1}\right] \cdot h \end{pmatrix}, \quad (2)$$

where $r = ||\mathbf{p}||_2$ is the range, $\mathrm{f} = \mathrm{f}_{\mathrm{up}} + \mathrm{f}_{\mathrm{down}}$ is the vertical field-of-view of the sensor, and w, h are the width and height of the resulting vertex map $\mathcal{V}_D$. This projection function is then also used to find the correspondence between the current scan and the rendered model view.

This projective data association is the key to enable highly efficient, dense mapping with laser range data, since we avoid to explicitly search for nearest neighbors.

Figure 3 shows an example point cloud with corresponding vertex and normal map, where we also included the corresponding coordinate systems.

Given the vertex map $\mathcal{V}_D$, we compute for each coordinate (u, v) a corresponding normal in $\mathcal{N}_D$ using cross products over forward differences, *i.e.*,

$$\begin{aligned} \mathcal{N}_D((u, v)) = &\left(\mathcal{V}_D((u+1, v)) - \mathcal{V}_D((u, v))\right) \\ &\times \left(\mathcal{V}_D((u, v+1)) - \mathcal{V}_D((u, v))\right). \end{aligned} \quad (3)$$

We only store normals if the vertex map contains valid points for all coordinates and wrap image coordinates in the x-direction, *i.e.*, $\mathcal{V}_D((u, v)) = \mathcal{V}_D((0, v))$ if $u \geq w$ and $\mathcal{V}_D((u, v)) = \mathcal{V}_D((w-1, v))$ if $u < 0$. While this normal estimation can be affected by noisy measurements, we found that the accuracy of the normal estimates did not influence the performance of the frame-to-model ICP significantly. We furthermore did not see any benefit from a bilateral filtering of the vertex map before the computation of normal estimates that is usually applied with RGB-D data [18].

B. Map Representation

We employ a surfel-based map [31, 12, 24, 33], since it allows us to represent even large-scale environments and maintain dense, detailed geometric information of the point clouds at the same time (see Figure 4 for an example). Surfels are furthermore relatively fast to render and therefore well suited for our application. A surfel map $\mathcal{M}$ is an unordered set of surfels s, where each surfel is defined by a position $\mathbf{v}_s \in \mathbb{R}^3$, a normal $\mathbf{n}_s \in \mathbb{R}^3$, a radius $r_s \in \mathbb{R}$. Each surfel additionally carries two timestamps: the creation timestamp t_c and the timestamp t_u of its last update by a measurement.

Furthermore, a stability log odds ratio l_s is maintained using a binary Bayes Filter [29] to determine if a surfel is considered stable or unstable. For rendering of the model view using the map, only stable surfels are considered and rendered. We update l_s as follows:

$$\begin{aligned} l_s^{(t)} = l_s^{(t-1)} &+ \mathrm{odds}\left(p_{\mathrm{stable}} \cdot \exp\left(-\frac{\alpha^2}{\sigma_\alpha^2}\right) \exp\left(-\frac{d^2}{\sigma_d^2}\right)\right) \\ &- \mathrm{odds}(p_{\mathrm{prior}}), \end{aligned} \quad (4)$$

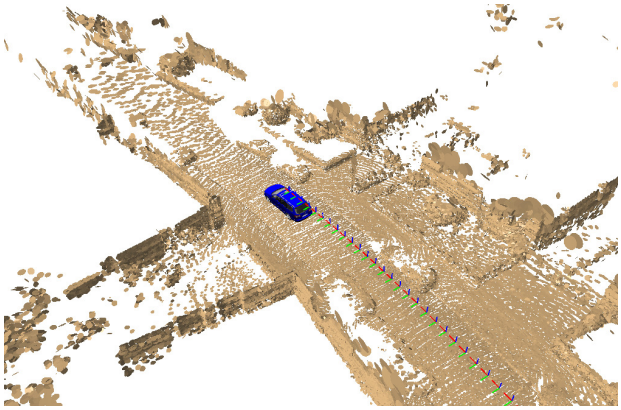


Fig. 4. Surfel map aggregated over multiple scans.

where $\mathrm{odds}(p) = \log(p \cdot (1-p)^{-1})$ and p_{stable} and p_{prior} are probabilities for a stable surfel given a compatible measurement and the prior probability, respectively. The terms $\exp(-x^2\sigma^{-2})$ are used to account for noisy measurements, where α is the angle between the surfel's normal $\mathbf{n}_s$ and the normal of the measurement to be integrated and d is the distance of the measurement in respect to the associated surfel. The measurement normal is taken from $\mathcal{N}_D$ and the correspondences from the frame-to-model ICP, see Section III-C.

In contrast to approaches using a deformation graph [31, 33], we let the poses directly manipulate the map by linking surfels to a pose via the creation timestamp t_c. Thus, the surfel's position $\mathbf{v}_s$ and normal $\mathbf{n}_s$ are local coordinates in the coordinate frame C_{t_c}, which allows us to modify the surfel map by a simple modification of the corresponding pose $\mathbf{T}_{WC_{t_c}}$ at time t_c. Thus, we are able to modify the map after a loop closure without the need to rebuild the map by reintegrating the past laser scans.

In line with Whelan *et al.* [33], we distinguish the active parts $\mathcal{M}_{\mathsf{active}}$ and inactive map parts $\mathcal{M}_{\mathsf{inactive}}$ at timestep t, comprised of all recently updated surfels, *i.e.*, $t_u \geq t - \Delta_{\mathsf{active}}$, and not recently created surfels, *i.e.*, $t_c < t - \Delta_{\mathsf{active}}$, respectively. The odometry is only estimated using $\mathcal{M}_{\mathsf{active}}$ and the loop closures are only searched in $\mathcal{M}_{\mathsf{inactive}}$.

C. Odometry Estimation

The observerd point cloud is aligned to a rendered representation $\mathcal{V}_M$ and $\mathcal{N}_M$ of the model in coordinate frame C_{t-1}, *i.e.*, we apply $\mathbf{T}_{WC_{t-1}}^{-1}$ before rendering all surfels, resulting in a local view of the already mapped world at timestmap $t-1$. We incrementally minimize the point-to-plane error given by

$$E(\mathcal{V}_D, \mathcal{V}_M, \mathcal{N}_M) = \sum_{\mathbf{u} \in \mathcal{V}_D} \underbrace{\mathbf{n}_u^\top \cdot \left(\mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} - \mathbf{v}_u\right)^2}_{r_{\mathbf{u}}}, \quad (5)$$

where each vertex $\mathbf{u} \in \mathcal{V}_D$ is projectively associated to a reference vertex $\mathbf{v}_u \in \mathcal{V}_M$ and its normal $\mathbf{n}_u \in \mathcal{N}_M$ via

$$\mathbf{v}_u = \mathcal{V}_M\left(\Pi\left(\mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u}\right)\right), \quad (6)$$

$$\mathbf{n}_u = \mathcal{N}_M\left(\Pi\left(\mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u}\right)\right). \quad (7)$$

Here, $\mathbf{T}_{C_{t-1}C_t}^{(k)}$ corresponds to the current pose estimate of the so-called *frame-to-model ICP* at iteration k. If $\Pi(\mathbf{u})$ maps to outside of the vertex map or the corresponding vertex or normal are undefined, we ignore the error term. For outlier rejection, we filter out correspondences exceeding a distance of δ_{ICP} or having an angle difference larger than θ_{ICP} between $\mathbf{n}_u$ and the corresponding normal of $\mathbf{u}$ in $\mathcal{N}_D$. We initialize the ICP pose $\mathbf{T}_{C_{t-1}C_t}^{(0)} = \mathbf{T}_{C_{t-2}C_{t-1}}$ with the last pose increment to warm start the optimization.

For solving the non-linear least squares problem, we represent the linearized pose increments by Lie-algebra elements $\delta \in \mathfrak{se}(3)$, which we simply write as vectors $\delta \in \mathbb{R}^6$ and use the Rodrigues' formula to compute the exponential map $\exp(\cdot) : \mathfrak{se}(3) \mapsto SE(3)$ to update the current pose estimate $\mathbf{T}_{C_{t-1}C_t}^{(k)}$, *i.e.*, $\mathbf{T}_{C_{t-1}C_t}^{(k)} = \exp(\delta) \cdot \mathbf{T}_{C_{t-1}C_t}^{(k-1)}$.

We minimize the objective of Equation 5 using Gauss-Newton and determine increments δ by iteratively solving

$$\delta = \left(\mathbf{J}^\top \mathbf{W} \mathbf{J}\right)^{-1} \mathbf{J}^\top \mathbf{W} \mathbf{r}, \tag{8}$$

where $\mathbf{W} \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing weights w_u for each residual r_u, $\mathbf{r} \in \mathbb{R}^n$ is the stacked residual vector, and $\mathbf{J} \in \mathbb{R}^{n \times 6}$ the Jacobian of $\mathbf{r}$ with respect to the increment δ. For a single residual $r_{\mathbf{u}}$ of vertex $\mathbf{u}$, it follows (see *e.g.* [18, 9]) that a row of the Jacobian $\mathbf{J}$ is given by

$$\mathbf{J}_u = \begin{bmatrix} \mathbf{n}_u^\top & (\mathbf{n}_{\mathbf{u}} \times \mathbf{v}_{\mathbf{u}})^\top \end{bmatrix}. \tag{9}$$

We use a Huber weighting [11] to weaken the influence of outliers that could not be removed by our outlier rejection.

D. Map Update

Given the current pose $\mathbf{T}_{WC_t}$ of the frame-to-model ICP, we integrate the points inside $\mathcal{V}_D$ into the surfel map. For this purpose, we have to decide which already existing surfels must be updated and which measurements must be used to initialize a new surfel in the surfel map.

For this decision, we start by computing the radius r_s of potential new surfel s for each $\mathbf{v}_s \in \mathcal{V}_D$ and corresponding normal $\mathbf{n}_s \in \mathcal{N}_D$ with the aim to cover roughly the corresponding pixel of the vertex map:

$$r_s = \frac{\sqrt{2}\,\|\mathbf{v}_s\|_2 \cdot p}{\text{clamp}(-\mathbf{v}_s^\top \mathbf{n}_s \cdot ||\mathbf{v}_s||_2^{-1}, 0.5, 1.0)}, \tag{10}$$

where $p = \max(w \cdot f_{\text{horiz}}^{-1}, h \cdot f_{\text{vert}}^{-1})$ corresponds to the pixel size and $\text{clamp}(x, u, l) = \min(u, \max(x, l))$ to the clamping operation. In contrast to prior approaches [31, 12, 33], we found this restriction needed in outdoor environments to avoid overly large surfels at long ranges. The radius is proportional to the accuracy of the measurement, since a small radius corresponds to a close distance measurement on surfaces with a normal pointing in the direction of the laser beam.

Then, we render $\mathcal{V}_M$, $\mathcal{N}_M$ and an index map $\mathcal{I}_M$ containing the indices of the nearest surfels in respect to the sensor origin with the final estimated pose $\mathbf{T}_{WC_t}$ to determine visible model surfels and their indices for updating. Each measurement point $\mathbf{v}_s$ is projected to $\mathcal{I}_M$ using Equation 2 to find the corresponding model surfel s'.

For updating the surfel map, we first determine if the data surfel s is compatible with the associated surfel s', *i.e.*, it holds $|\mathbf{n}_{s'}^\top(\mathbf{v}_s - \mathbf{v}_{s'})| < \delta_M$ and $||\mathbf{n}_s \times \mathbf{n}_{s'}|| < \sin(\theta_M)$. Depending on the compatibility, we now distinguish the following cases for a data surfel s and its associated model surfel s':

1) If the data surfel is compatible, we increase the stability of the associated model surfel. If the measurement is more precise, *i.e.*, $r_s < r_{s'}$, we also update the model surfel using an exponential moving average:
$$\mathbf{v}_{s'}^{(t)} = (1-\gamma) \cdot \mathbf{v}_s + \gamma \cdot \mathbf{v}_{s'}^{(t-1)}, \tag{11}$$
$$\mathbf{n}_{s'}^{(t)} = (1-\gamma) \cdot \mathbf{n}_s + \gamma \cdot \mathbf{n}_{s'}^{(t-1)} \tag{12}$$
$$r_{s'}^{(t)} = r_s \tag{13}$$
Thus, we only refine surfels to avoid losing information we gathered from closer ranges [12].
2) Otherwise, we decrease the stability of the associated model surfel s' and initialize a new surfel. If a measurement cannot be assigned to any existing surfel, we initialize a new surfel.

After updating the map, we remove all unstable surfels which have a too low stability value, $l_s < \tau_D$, and are at the same time already too old, $t - t_c > \tau_O$. By this means, we are able to remove unstable surfels mainly caused by dynamic objects, but also irrelevant surfels that are caused by clutter.

E. Loop Closures

For loop closure detection, we exploit the surfel map to render views from $\mathcal{M}_{\text{inactive}}$. Analogous to the odometry, we now use this rendered view to register the current point cloud. A loop closure is only considered if a composed virtual map view built from this alignment fits to the current measurements. We try to verify a potential loop closure by tracking the pose in the active and the inactive map at the same time. Only after successful verification, we include a loop closure constraint in the pose graph.

Detection. For loop closure detection, we first try to find a possible loop closure candidate in the inactive map $\mathcal{M}_{\text{inactive}}$. From all poses, $\mathbf{T}_{WC_0}, \ldots, \mathbf{T}_{WC_{t-\Delta_{\text{active}}}}$, we consider only one candidate in a radius δ_{loop} of the current pose estimate, *i.e.*, $j^* = \arg\min_{j \in 0,\ldots,t-\Delta_{\text{active}}} ||\mathbf{t}_{WC_t} - \mathbf{t}_{WC_j}||$. Note that the orientation of the candidate does not matter and therefore even candidates with small overlap between scans and different viewpoints are considered.

For candidate j^*, we try to align the current point cloud to the rendered view at the corresponding pose $\mathbf{T}_{WC_{j^*}}$ using the frame-to-model ICP. Since ICP heavily depends on the initialization, we check multiple initializations $\mathbf{T}_{C_{j^*}C_t}^{(0)}$ with respect to the old pose to compensate for rotational and translational drift. We compose the initial pose from the rotational difference $\mathbf{R}_{C_{j^*}C_t}$ between the current pose estimate and the candidate's rotation and the translational difference $\mathbf{t}_{C_{j^*}C_t}$, *i.e.*,

$$\mathbf{R}_{C_{j^*}C_t} = \mathbf{R}_{WC_{j^*}}^{-1} \mathbf{R}_{WC_t} \tag{14}$$
$$\mathbf{t}_{C_{j^*}C_t} = \mathbf{R}_{WC_{j^*}}^{-1} (\mathbf{t}_{WC_t} - \mathbf{t}_{WC_{j^*}}) \tag{15}$$

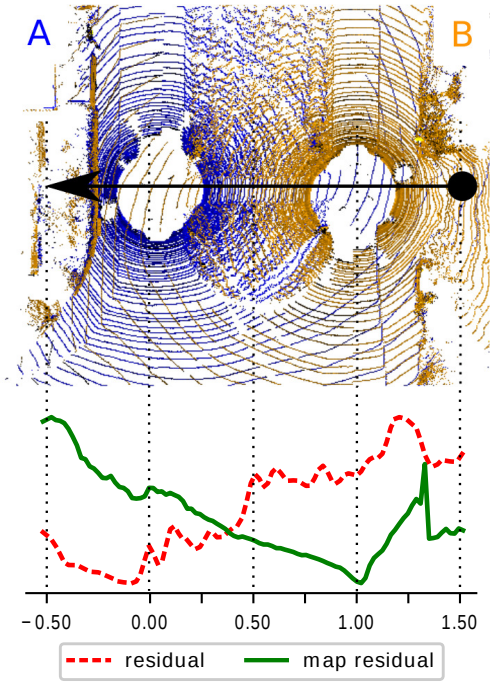


Fig. 5. Shown are two scans A (blue) and B (orange) from different sides of a street from KITTI sequence 06 from a birds eye view. From both sides, the other side's street is not visible and therefore hard to distinguish the correct from the incorrect loop closure just by means of residual of an alignment. Shown below is the residual along the arrow, *i.e.*, at every position on the arrow we compute the residual from A to B if B would exactly lie at that position. Shown is the residual (red) of B to A and the map residual (green) of scan B to the composed virtual view .

We then compose the initialization $\mathbf{T}^{(0)}_{C_{j^*}C_t}$ using $\mathbf{R}_{C_{j^*}C_t}$ as rotation and $\lambda \mathbf{t}_{C_{j^*}C_t}$, with $\lambda = \{0.0, 0.5, 1.0\}$, as translation.

Due to these multiple initializations, it is crucial to decide which of the ICP's results is the best and most consistent with the current measurements. As we also have to consider situations with low overlap between individual scans due to occlusions, we cannot simply evaluate the residual (see Equation 5), outlier count, or inlier count, of the current measurement with respect to the rendered inactive map.

Figure 5 shows such a situation for an example of the KITTI Vision Benchmark. Here, two point clouds (blue and orange) are shown which partially overlap and where we would like to find a loop closure. Below the correctly aligned scans, we show the resulting residuals for different points on a line between the ground truth poses, here $x = \{0.0, 0.5, 1.0\}$ correspond to the three initializations used for loop closure detection. The lowest residual (depicted as red dashed line) is reached between -0.2 and 0.0, while the residual at the correct location, $x = 1.0$, is by far larger than the wrong alignments. Thus, we would reject the correct alignment since its residual is simply larger than an incorrect alignment.

For solving such cases with small overlap between scans, we render a virtual view of the map after a potential loop closure using the transformations from the odometry estimation and aforementioned candidates. The virtual view is a composition

TABLE I
PARAMETERS OF OUR APPROACH

Parameter	value	description
$w \times h$	900×64	dimensions of vertex/normal map
θ_{ICP}	$30.0°$	outlier threshold (angle)
δ_{ICP}	$2\,m$	outlier threshold (distance)
δ_{M}	$0.2\,m$	map outlier threshold (distance)
θ_{M}	$30°$	map outlier threshold (angle)
k	0.5	Huber factor
$\text{f}_{\text{up}}, \text{f}_{\text{down}}$	$3.0°$, $25°$	vertical field-of-view
p_{stable}	0.6	stability
p_{prior}	0.5	prior stability
t_{old}	100	surfel age threshold
σ_α, σ_d	1.0	stability scaling
τ_D	$\text{odds}(1 - p_{\text{stable}})$	stability threshold
τ_O	3	max. unstable age
γ	0.9	exponential weight
G	9	number of submap grid cells
e	10 m	size of a submap
Δ_{active}	100	time horizon for active map
$\Delta_{\text{verification}}$	5	number of verifications needed
δ_{loop}	50	max. distance of loop closure
$\epsilon_{\text{residual}}$	1.15	acceptance threshold

of the rendered model view of $\mathcal{M}_{\text{inactive}}$ and the current point cloud and this composition should only fit to the current scan if the composition agrees with the current point cloud data.

Given the transformation between the current point cloud and the potential loop closure, we can now transform both into a common coordinate frame and generate a composed vertex map $\mathcal{V}_C$ and normal map $\mathcal{N}_C$. First, the vertex and normal maps are filled by rendering the inactive map at the candidate position. From the current vertex $\mathcal{V}_D$ and normal map $\mathcal{V}_D$, we add entries to the composed maps if the point in $\mathcal{V}_D$ is closer than the existing point in $\mathcal{V}_C$.

We compute now $E(\mathcal{V}_D, \mathcal{V}_C, \mathcal{N}_C)$, which we call *map residual* E_{map}, in respect to the current scan, but now with the composed vertex map $\mathcal{V}_C$ and $\mathcal{N}_C$. Only if this composed view is consistent with our current measurements, we consider the candidate as a valid loop closure candidate. A possible alignment is consistent if the relative error between the residuals of that composed map E_{map} and the residual in respect to the active map E_{odom} is small, *i.e.*, $E_{\text{map}} < \epsilon_{\text{residual}} \cdot E_{\text{odom}}$, and if there are enough inliers and valid points in the rendered composed view. As can be seen from Figure 5, the map residual is minimal for the correct pose at $x = 1.0$ and therefore makes it possible to distinguish valid and invalid loop closure candidates.

Verification. Even though we try to rule out wrong loop closures by the rendering of a virtual map, this criterion can still lead to invalid loop closures. Thus, we aim to verify the loop closure by tracking the position in the active and inactive map. At each timestep $t+1, \ldots, t+\Delta_{\text{verification}}$ after a detected loop closure at timestep t, we estimate the odometry increment and apply this to the pose in the active and inactive part of the map. We render a virtual view of the map with these poses and check again for consistency with the current measurements, as described for the detection. Only if the consistency check succeeds for at least $\Delta_{\text{verification}}$ timesteps, we consider the loop closure valid and verified.

TABLE II
RESULTS ON KITTI ODOMETRY (TRAINING)

Approach	Sequence 00*	01	02*	03	04	05*	06*	07*	08*	09*	10	Average
Frame-to-Frame	0.9/2.1	1.2/4.0	0.8/2.3	0.7/1.4	1.1/11.9	0.8/1.5	0.6/1.0	1.2/1.8	1.0/2.5	0.8/1.9	1.0/1.8	0.9/2.9
Frame-to-Model	0.3/**0.7**	0.5/1.7	0.4/1.1	0.5/**0.7**	0.3/**0.4**	0.2/0.5	0.2/**0.4**	0.3/**0.4**	0.4/**1.0**	0.3/**0.5**	0.3/**0.7**	0.3/**0.7**
Frame-to-Model with loop closure	0.2/**0.7**	0.5/1.7	0.4/1.2	0.5/**0.7**	0.3/**0.4**	0.2/**0.4**	0.3/0.5	0.6/0.7	0.4/1.2	0.2/0.6	0.3/**0.7**	0.3/0.8
LOAM [35]	-/0.8	-/**1.4**	-/**0.9**	-/0.9	-/0.7	-/0.6	-/0.7	-/0.6	-/1.1	-/0.8	-/0.8	-/0.8
S-LSD [6]	0.3/0.6	0.3/2.4	0.2/0.8	0.3/1.0	0.3/0.4	0.2/0.7	0.2/0.7	0.3/0.6	0.3/1.1	0.3/1.1	0.3/0.7	0.3/0.9
SOFT-SLAM [2]	0.2/0.7	0.2/1.0	0.2/1.4	0.2/0.7	0.2/0.5	0.2/0.4	0.1/0.4	0.2/0.4	0.2/0.8	0.2/0.6	0.3/0.7	0.2/0.7

Relative errors averaged over trajectories of 100 to 800 m length: relative rotational error in degrees per 100 m / relative translational error in %,
Sequences marked with an asterisk contain loop closures. Bold numbers indicate top performance for laser-based approaches.

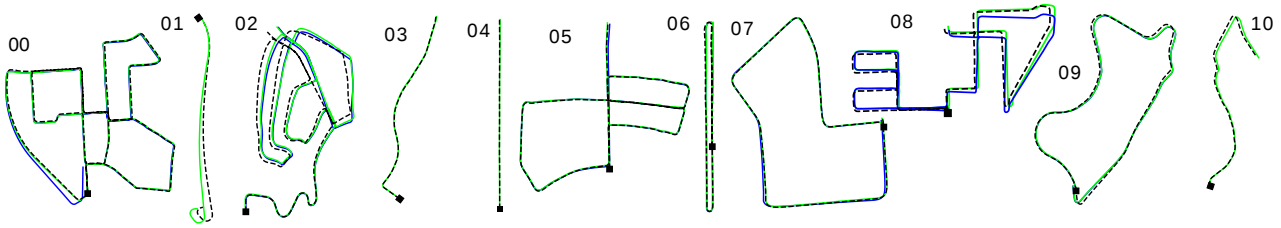


Fig. 6. Trajectories of the training set. The dashed black trajectory correspond to the GPS-based ground truth, blue to our approach without loop closure and green to our approach with loop closure. Sequence 01, 03, 10 have been rotated to save space. (Best viewed in color)

Pose graph optimization. As soon as a verified loop closure is found, we add loop closure constraints to a pose graph, which is then used for optimization. The poses are always initialized with the currently estimated poses.

In line with Hess *et al.* [10], we also perform the actual pose graph optimization in a separate thread. When the pose graph optimization is finished, we integrate the optimized poses in the map and update the current pose estimates accordingly.

F. Implementation Details

We implemented our approach using OpenGL 4.0 and we represented most of the data structures via textures or transform feedbacks and manipulate the values via vertex, geometry and fragment shaders. We furthermore leverage the rendering pipeline to generate vertex and normal maps by rendering small disks for each front-facing surfel.

To allow real-time capable processing even in ever growing maps, we employ a submapping approach to offload parts of the map from the GPU memory to the main memory. To this end, we implemented a two-dimensional rolling grid data structure of size G with extent e for each submap, which is used to determine which parts are obsolete and can be downloaded from the GPU. However, if parts of the environment are revisited, the corresponding surfels can also be uploaded again. The position of the rolling grid is updated according to the currently estimate sensor pose.

To avoid large peaks in the runtime, we implemented a delayed downloading procedure which only downloads a single grid cell in each update iteration. This effectively amortizes the time for downloading larger parts and spreads it over multiple map updates.

For pose graph optimization, we employ gtsam[3] and optimize the pose graph using Levenberg Marquardt with a maximum of 100 iterations. As the optimized poses are directly integrated into the map, we initialize the pose estimates with the last optimized poses and the current odometry estimates. Usually, the pose graph optimization terminates in less than 10 iterations.

IV. EXPERIMENTAL EVALUATION

We evaluate our approach on the odometry datasets of the KITTI Vision Benchmark [8], where we use the provided point clouds from the Velodyne HDL-64E S2 recorded at a rate of 10 Hz. The dataset contains data from different street environments ranging from inner city to highway traffic. Especially, the highway datasets are a major challenge, since the sensor moves at high speed, other fast moving dynamic objects are present, and the environment has only very few distinct structures which can be used. Here, most of the laser returns correspond to the flat street and only few correspond to traffic signs or some sparse trees or bushed along the highway.

The parameters of our approach, which we experimentally selected for the training data, are summarized in Table I. We use an Intel i7-6700@3.4 GHz with 16 GB RAM, and an Nvidia GeForce GTX 960 with 4 GB RAM.

Ablation study. Using the training data of the KITTI Vision Benchmark, we show the influence of the different design decisions of our approach.

Table II shows our approach without using the map (frame-to-frame), using the map (frame-to-model), and the map-based

[3]Version 4.0, available at https://bitbucket.org/gtborg/gtsam.

approach with loop closures for all sequences in detail.

Starting from a 'frame-to-frame ICP', which simply projectively associates points based on the last point cloud, we now discuss the influence of different design decisions.

Adding then the surfel map to aggregate past information helps to reduce drift, since it also adds information in areas which are currently sparsely covered. The stability estimation of a surfel ensures that only static objects remain in the surfel map. Therefore dynamic objects cannot cause false correspondences and consequently lead to inferior pose estimation performance. However, since we are not integrating loop closures, we cannot correct accumulated drift if we revisit already mapped areas.

Finally, adding loop closures and the pose graph optimization does lead to globally consistent maps. Adding the ability to close loops does only marginally improve the performance with respect to the KITTI metrics as the performance is average over short parts (up to 800 m) of the trajectory. While the numbers do not properly convey the improvement, the plots of the trajectories (see Figure 6) shows a more globally consistent trajectory. Please see also Figure 1 for a qualitative assessment of the global consistency of the trajectory for sequence 00.

Comparison with state-of-the-art. We included here for comparison the reported results of LOAM [35], a laser-based odometry approach, and for indirect comparison with state-of-the-art vision approaches, Stereo LSD-SLAM [6], a stereo vision-based complete SLAM system with pose graph optimization, and the currently best performing approach SOFT SLAM [2], also a stereo vision-based approach.

We perform generally on par with the state-of-the-art in laser-based odometry and often achieve better results in terms of translational error. However, the actual improvement of the ability to close loops is hard to evaluate using the KITTI Vision Benchmark, since it only provides ground truth trajectories generated with an GPS-based inertial navigation system. In the training data, we observed some inconsistencies in the height of the recorded GPS-based trajectory. For indoor datasets, it is possible to record highly precise motion capture data, which also enables to evaluate the absolute trajectory error in respect to the 'perfect' drift free motion capture trajectory.

Finally, we estimated the poses of the testset using the aforementioned parameters. Overall, we achieved an average rotational error of 0.0032 deg/m and an average translational error of 1.4% compared to 0.0017 deg/m and 0.7% translational error of LOAM on KITTI.

From visual inspection, we found that our method cannot correctly estimate the motion of the vehicle in sequences with very few structured objects, like highways. Due to the lack of structure to distinguish static and moving objects, the method sometimes fails to distinguish which objects move, especially if multiple objects are moving consistently, like cars in a traffic jam. In particular, dynamic objects can corrupt the map by spurious wrongly integrated surfels. Often two different cars can lead to consistent surfels in consecutive laser scans and therefore get wrongly integrated. In these ambiguous cases, integration of inertial measurements could help to get more consistent and accurate estimation results.

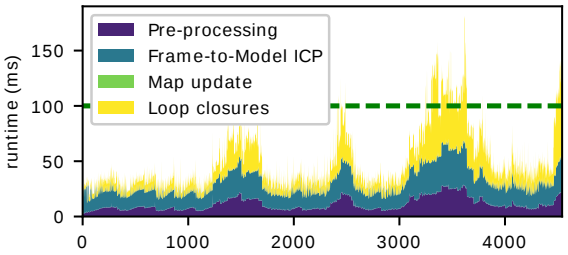


Fig. 7. Processing time needed to map sequence 00 from the KITTI Vision Benchmark. See Figure 1 for the finally registered complete point cloud.

Runtime. Figure 7 shows the processing time needed to process sequence 00 from the KITTI Vision Benchmark (see also Figure 1 for a visual impression of the complexity). We selected this sequence where parts of the environment are revisited and therefore often parts have to be down-, but also uploaded to the GPU, but also loop closure must be detected and subsequently verified. The odometry with the update of the map can be done with 31 ms on average and needs at most 71 ms. Together with the loop closure detection and verification, we need at most 189 ms. After few detection steps, where we have to check multiple candidates (peaks in the runtime plot), we then only have to verify a detected loop closure. Overall, our approach runs at 48 ms on average and therefore is able to process a scan at 20 Hz on average.

V. CONCLUSION

We presented a dense approach for laser-based odometry and mapping that is capable of performing all computations online and produces globally consistent maps. Our approach leverages a surfel-based map representation and performs projective data association of the current scan to the surfel maps at each iteration. As our experimental evaluation illustrates, this yields accurate registration results and globally consistent maps that are on par with the state-of-the-art.

A natural next step would be the integration of color information to improve the selection of correspondences while the projective data association [33, 20]. We furthermore plan to integrate a global loop closure search, which would also enable to find loops closures if the odometry estimate drifts to much. Also, a detection and tracking of dynamic objects such as the one by Moosmann and Stiller [17] has the potential to increase the robustness and quality of the registration results, since it would enable to reliably remove dynamic objects.

VI. ACKNOWLEDGEMENTS

This work has been supported by the DFG (BE 5996/1-1). We thank Johannes Schneider, Lorenzo Nardi, Mathias Hans, Igor Bogoslavskyi, and Philipp Lottes for fruitful discussions. We thank Thomas Whelan for making his ElasticFusion code available, which inspired our implementation.

REFERENCES

[1] 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: Towards the Robust-Perception Age. *IEEE Trans. on Robotics (TRO)*, 32:1309–1332, 2016.

[2] I. Cvisic, J. Cesic, I. Markovic, and I. Petrovic. SOFT-SLAM: Computationally Efficient Stereo Visual SLAM for Autonomous UAVs. *Journal of Field Robotics (JFR)*, 2017.

[3] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. BundleFusion: Real-time Globally Consistent 3D Reconstruction using Online Surface Reintegration. *ACM Trans. on Graphics (TOG)*, 2016.

[4] A. Das, J. Servos, and S.L. Waslander. 3D Scan Registration Using the Normal Distributions Transform with Ground Segmentation and Point Cloud Clustering. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[5] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C.C. Lerma. SegMatch: Segment Based Place Recognition in 3D Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[6] J. Engel, J. Stückler, and D. Cremers. Large-scale direct slam with stereo cameras. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[7] J. Engel, V. Koltun, and D. Cremers. Direct Sparse Odometry. *arXiv:1607.02565*, 2016.

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[9] A. Handa. Simplified Jacobians in 6-DoF Camera Tracking. Technical report, University of Cambridge, 2014.

[10] W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-Time Loop Closure in 2D LIDAR SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[11] P. J. Huber. *Robust Statistics*. Wiley, 1981.

[12] M. Keller, D. Lefloch, M. Lambers, and S. Izadi. Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, pages 1–8, 2013.

[13] J. Levinson and S. Thrun. Robust Vehicle Localization in Urban Environments Using Probabilistic Maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 4372–4378, 2010.

[14] J. Levinson, M. Montemerlo, and S. Thrun. Map-Based Precision Vehicle Localization in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[15] M. Magnusson, H. Andreasson, A. Nuechter, Achim, and J. Lilienthal. Appearance-Based Loop Detection from 3D Laser Data Using the Normal Distributions Transform. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[16] F. Moosmann and C. Stiller. Velodyne SLAM. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, pages 393–398, 2011.

[17] F. Moosmann and C. Stiller. Joint Self-Localization and Tracking of Generic Objects in 3D Range Data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1138–1144, 2013.

[18] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proc. of the Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011.

[19] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. *Proc. of the SIGGRAPH Asia*, 32(6), 2013.

[20] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan. Elastic LiDAR Fusion: Dense Map-Centric Continuous-Time SLAM. *arXiv preprint*, abs/1711.01691, 2017.

[21] L. Platinsky, A.J. Davison, and S. Leutenegger. Monocular Visual Odometry: Sparse Joint Optimisation or Dense Alternation? In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[22] T. Röhling, J. Mack, and D. Schulz. A Fast Histogram-Based Similarity Measure for Detecting Loop Closures in 3-D LIDAR Data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 736–741, 2015.

[23] J.P. Saarinen, T. Stoyanov, H. Andreasson, and A.J. Lilienthal. Fast 3D Mapping in Highly Dynamic Environments Using Normal Distributions Transform Occupancy Maps. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[24] R. F. Salas-Moreno, B. Glocker, P. H. J. Kelly, and A. J. Davison. Dense Planar SLAM. In *Proc. of the Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, pages 157–164, 2014.

[25] B. Steder, G. Grisetti, and W. Burgard. Robust Place Recognition for 3D Range Data Based on Point Features. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2010.

[26] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place Recognition in 3D Scans Using a Combination of Bag of Words and Point Feature Based Relative Pose Estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[27] T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal. Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations. *Intl. Journal of Robotics Research (IJRR)*, 31(12):1377–1393, 2012.

[28] J. Stückler and S. Behnke. Multi-Resolution Surfel Maps for Efficient Dense 3D Modeling and Tracking. *Journal of Visual Communication and Image Representation (JVCIR)*, 25(1):137–147, 2014.

[29] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[30] M. Velas, M. Spanel, and A. Herout. Collar Line Segments for Fast Odometry Estimation from Velodyne Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[31] T. Weise, T. Wismer, B. Leibe, and L. Van Gool. Online loop closure for real-time interactive 3D scanning. *Computer Vision and Image Understanding*, 115:635–648, 2011.

[32] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. Real-time large scale dense RGB-D SLAM with volumetric fusion. *Intl. Journal of Robotics Research (IJRR)*, 34(4-5):598–626, 2014.

[33] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *Proc. of Robotics: Science and Systems (RSS)*, 2015.

[34] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*, 2014.

[35] J. Zhang and S. Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41:401–416, 2017.