UNIVERSITÄT BONN

Inaugural-Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Landwirtschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn
Institut für Geodäsie und Geoinformation

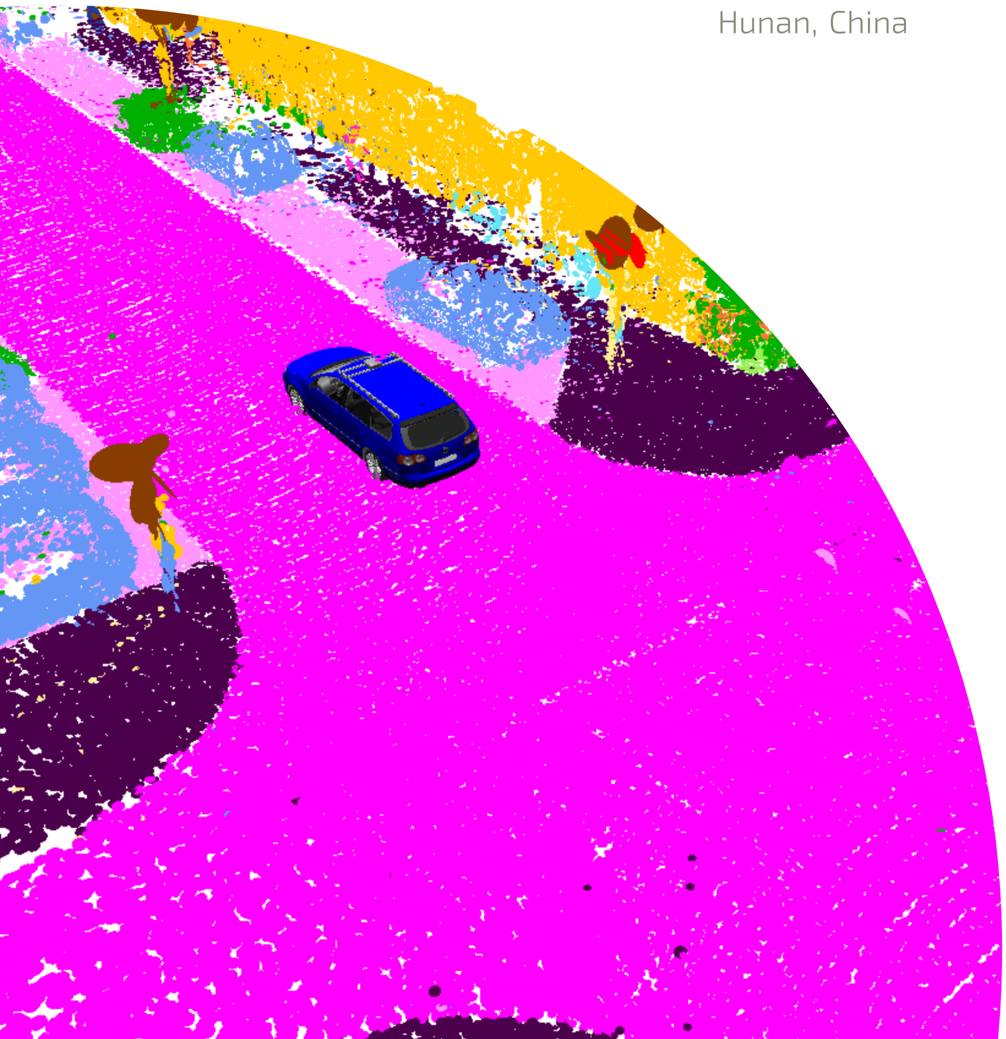
LiDAR-Based Semantic Perception for Autonomous Vehicles

von

Xieyuanli Chen

aus

Hunan, China



Bonn 2022

Referent:
Prof. Dr. Cyrill Stachniss, University of Bonn, Germany

1. Korreferent:
Prof. Dr. Christopher McCool, University of Bonn, Germany

2. Korreferent:
Prof. Dr. Philippe Giguère, Laval University, Canada

Tag der mündlichen Prüfung: 17.08.2022

Angefertigt mit Genehmigung der Landwirtschaftlichen Fakultät der Universität Bonn

Erklärung der Urheberschaft

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenenen Hilfsmittel angefertigt habe; die aus fremdem Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum	(Unterschrift)

Zusammenfassung

EINE der grundlegenden Bausteine, die es mobilen Systemen ermöglichen, autonom zu agieren, ist das Verständnis für die das System umgebenden Welt. Hierfür ist ein komplexer Verarbeitungsprozess nötig, der ausgehend von den auf dem System verbauten Sensoren deren Daten aufnimmt, analysiert, verarbeitet uns schließlich eine Interpretation der u. U. dynamischen Szene liefert. Laserscanning-Sensoren, kurz LiDAR für “light detection and ranging sensors”, sind eine der am häufigsten benutzten Sensoren, da sie robust gegenüber Beleuchtungsänderungen sind und hochgenaue Distanzmessungen erlauben. Basierend auf LiDAR-Systemen können autonome Fahrzeuge komplexe Aufgaben erfüllen, indem sie die Position und Art der Objekte in der Szene verstehen, und daraus Handlungsalternativen entwickeln und schließlich ausführen. Zwei wichtige dieser komplexen Aufgaben sind die Lokalisierung in einer gegebenen Karte und simultane Kartenerzeugung und Lokalisierung, kurz SLAM. Die Position des Systems in einer Karte ist die Vorraussetzung für viele andere Aufgaben. Für eine Innenraum-Umgebung mit der Annahme einer statischen Szene können traditionelle, auf LiDAR basierende Methoden für die globale Lokalisierung oder SLAM-Algorithmen genaue Posenschätzungen zur Verfügung stellen. Sollen sich allerdings selbstfahrende Fahrzeuge im Außenbereich in dynamisch verändernden Szenen bewegen, ist eine rein-geometrische Information für eine sichere Lokalisierung und Kartierung nicht ausreichend. Ein Verstehen der Szene auf einem höheren Abstraktionsniveau, das insbesondere die semantische Information über die Welt beinhaltet, wird zum robusten und sicheren Einsatz von selbstfahrenden Fahrzeugen in komplexen Umgebungen benötigt.

Der Hauptbeitrag dieser Doktorarbeit sind neue Methoden für die LiDAR-basierte globale Lokalisierung und SLAM, die die Vorteile der semantischen Objektinformation ausnutzen. Die Arbeit gliedert sich in drei Teile: Der erste Teil behandelt die Frage, wie SLAM und Lokalisierung durch semantische Information verbessert werden kann. Es wird eine semantik-basierte LiDAR-SLAM Methode vorgestellt. Diese benutzt die Ergebnisse eines semantischen neuronalen Netzes, um die Posengenauigkeit zu verbessern und konsistente semantische Karten der Umgebung zu erstellen. Weiterhin schlagen wir ein neues neuronales Netz vor, das die Ähnlichkeit zwischen zwei LiDAR-Scans unter Ausnutzung von geometrischen und semantischen Informationen ermitteln kann. Basierend auf diesem Ähnlichkeitsmaß kann unser Netzwerk erfolgreich Kandidaten für einen Schleifenschluss in einem SLAM System finden. Außerdem ist es damit möglich, globale Lokalisierung auch bei Daten aus unterschiedlichen Jahreszeiten zu erreichen.

Der zweite Teil der Arbeit untersucht, welche semantischen Klassen für spezielle Aufgaben nützlich sind. In diesem Kontext schlagen wir für SLAM eine neue Methode für die Segmentierung sich bewegender Objekte vor. Die Methode unterscheidet fahrende Fahrzeuge von statischen Objekten wie parkenden Fahrzeugen, Gebäuden, etc. Mit Hilfe der speziellen semantischen Klassen "statisch" und "sich bewegend" erhält man mit SLAM ein besseres Ergebnis als unter Verwendung von einer allgmeinen semantischen Segmentierung. Für die Lokalisierung schlagen wir vor, auf Stangen montierte Objekte wie Verkehrsschilder, Straßenlampen etc. zu verwenden. Diese sind zeitlich stabil und lassen sich einfach von der Umgebung in den Sensordaten unterscheiden. Mit Hilfe der Extraktion solcher Objekte aus den Sensordaten erreichen wir zuverlässige und genaue Lokalisierung des Fahrzeugs über lange Zeiträume hinweg.

Deep-learning basierte Ansätze können genaue punktweise semantische Ergebnisse liefern. Die Qualität hängt allerdings stark von der Menge und Diversität der verwendeten semantisch annotierten Trainingsdaten ab. Daher fokussieren wir uns im dritten Teil der Arbeit auf Verfahren zur automatischen Annotation für das Training von neuronalen Netzwerken. Für unsere speziellen Fragestellungen profitieren wir hier von der Vereinfachung der semantischen Klassen von einem Multi-Klassen-Problem hin zu einem Zwei-Klassen-Problem. Das macht eine vollautomatische Annotation möglich. Mit unserem Ansatz verringern wir die Äbhängigkeit von manuell durch den Menschen erzeugten Labels, sodass wir für die Netzwerke vom überwachten zum sebstüberwachten ("self-supervised") Training wechseln können. Unsere Methoden können einfach auch mit anderen Umgebungsdaten und anderen LiDAR Sensoren verwendet werden.

Alle hier in dieser Arbeit vorgeschlagenen Ansätze sind in begutachteten Konferenzbeiträgen und Zeitschriftenartikeln erschienen. Unser neuronales Netzwerk "OverlapNet" für LiDAR-basierten Schleifenschluss und Lokalisierung ist für den besten Artikel im Bereich Systeme auf der Konferenz "RSS: Robotics: Science and Systems" 2020 nominiert worden. Unsere Methode zur Segmentierung von bewegten Objekten wurde für eine Präsentation auf dem "Robotics Science and Systems Pioneers event" 2021 ausgewählt. Um zukünftige Forschung in den behandelten Themenfeldern zu erleichtern und zu fördern, stellen wir darüber hinaus für alle hier beschriebenen Methoden open-source Code zur Verfügung.

Abstract

Scene understanding is one of the fundamental building blocks that enable mobile systems to achieve autonomy. It is the process of perceiving, analyzing, and elaborating an interpretation of a 3D dynamic scene observed through the onboard sensors equipped on autonomous vehicles. The light detection and ranging sensors, in short LiDAR, are one of the popular sensors for autonomous vehicles to sense their surroundings, because they are robust to light changes and provide high-accurate range measurements. Based on LiDAR sensors, autonomous vehicles can explore environments, understand the locations and types of objects therein, and then make plans and execute actions to fulfill complex tasks. Among them, key capabilities are localization within a given map as well as simultaneous localization and mapping (SLAM), which provide the robot's location, the necessary prerequisite for other downstream tasks. Traditional LiDAR-based global localization and SLAM methods can provide accurate pose estimates in indoor environments with the static world assumption. However, as the demand for autonomous driving in dynamic outdoor environments grew, using only geometric and appearance information is not enough to provide reliable localization and mapping results for autonomous systems. A high-level understanding of the world, which includes the estimation of semantic information, is required for robust and safe deployments of autonomous vehicles in dynamic and complex real-world scenarios.

The main contributions of this thesis are novel approaches that exploit semantic information to improve the performance of LiDAR perception tasks such as SLAM and global localization for autonomous vehicles. This thesis consists of three parts. The first part focuses on how to apply semantic information for SLAM and localization. We present a semantic-based LiDAR SLAM method, which exploits semantic predictions from an off-the-shelf semantic segmentation network to improve the pose estimation accuracy and generate consistent semantic maps of the environments. We furthermore propose a novel neural network exploiting both geometric and semantic information to estimate the similarities between pairs of LiDAR scans. Based on these similarity estimates, our network can better find loop closure candidates for SLAM and achieve global localization in outdoor environments across seasons.

The second part investigates which type of semantics are useful for specific tasks. In this context, we propose a novel moving object segmentation method for SLAM. It aims at separating the actually moving objects such as driving cars from static or non-moving objects such as buildings, parked cars, etc. With more specific moving/non-moving semantics, we get a better SLAM performance compared to setups using general semantics. For localization, we propose to use pole-like objects such as traffic signs, poles, lamps, etc., due to their local distinctiveness and long-term stability. As a result, we obtain reliable and accurate localization results over comparably long periods of time.

Deep learning-based approaches can provide accurate point-wise semantic predictions. They, however, strongly rely on the diversity and amount of labeled training data that may be costly to obtain. In the third part, we therefore propose approaches that can automatically generate labels for training neural networks. Benefiting from specifying and simplifying the semantics for specific tasks, we turn the comparably challenging multiclass semantic segmentation problem into more manageable binary classification tasks, which makes automatic label generation feasible. Using our proposed automatic labeling approach, we alleviate the reliance on expensive human labeling for supervised training of neural networks and enable our method to work in a self-supervised way. Therefore, our proposed task-specific semantic-based methods can be easily transferred to different environments with different LiDAR sensors.

All our proposed approaches presented in this thesis have been published in peer-reviewed conference papers and journal articles. Our proposed OverlapNet for LiDAR-based loop closing and localization was nominated for the Best System Paper at the Robotics: Science and Systems (RSS) conference in 2020. Our proposed moving object segmentation method was selected to be presented at the Robotics Science and Systems Pioneers event in 2021. Additionally, we have made implementations of all our methods presented in this thesis open-source to facilitate further research.

Acknowledgements

IT was a wonderful and unique journey in my life to conduct my doctoral studies in our Photogrammetry and Robotics Lab at Bonn. First and foremost, I want to express my deepest appreciation to my supervisor and mentor, Cyrill Stachniss. I thank him for accepting me as one of his Ph.D. students and providing me with the best working environment and tremendous help and time to guide me carry out my research. All my work in this thesis would be impossible without his support. Besides teaching me scientific skills, Cyrill is also my life mentor. I have learned invaluable lessons from him to overcome challenges with positive attitudes, support workmates with utmost sincerity, supervise students with infinite patience, and so many other things that benefit my career and life, now and future.

I also want to thank Christopher McCool for reviewing my thesis. I enjoy the fruitful discussions with him and the interesting interactions with his students. I furthermore want to thank Philippe Giguère for reviewing my thesis. I appreciate his interest in my research and the effort he put into our cooperated paper.

Many thanks to my friends and lab mates: Jens Behley, Thomas Läbe, Birgit Klein, Igor Bogoslavskyi, Kaihong Huang, Olga Vysotska, Emanuele Palazzolo, Lorenzo Nardi, Philipp Lottes, Andres Milioto, Nived Chebrolu, Ignacio Vizzo, Jan Weyler, Louis Wiesmann, Federico Magistri, Benedikt Mersch, Tiziano Guadagnino, Rodrigo Marcuzzi, Lucas Nunes, Elias Marks, Matteo Sodano, Nicky Zimmerman, Haofei Kuang, Gianmarco Roggiolani, Yue Chong, Xingguang Zhong, and also Marija Popović. The interesting discussions, fantastic events and parties have made my working and living here so fulfilling and pleasant. These will be my fond memories and timeless treasures.

Specially thanks to Jens Behley and Thomas Läbe for their incredible and invaluable advice and support on my research and writing this thesis. I work most closely with Jens and appreciate all his guidance and the late-night discussions and inspirations on all my research. I am grateful for Thomas' effort and dedication to our nominated work. Without his support, this work would not have been possible with this quality. I also appreciate his help on the German abstract of this thesis.

Specially thanks also to Birgit Klein who has provided great support in my work and life in Bonn even before I came to Germany. My four-year journey here would not be so wonderful without her invaluable help.

I want to express my gratitude to all my friends and coauthors: Shijie Li, Louis Wiesmann, Ignacio Vizzo, Andrzej Reinke, Chenghao Shi, Wengbang Deng, Mengjie Zhou, Benedikt Mersch, Hao Dong, Mehul Arora, Lucas Nunes, Haofei Kuang, Tiziano Guadagnino, Junyi Ma, Jiadai Sun, Jingtao Sun, Si Yang, and Liren Jin. I thank them all for their outstanding work and trust in me. I am honored to join their work and look forward to future cooperation.

I want to extend my thanks to my previous supervisors and advisors: Zhiqiang Zheng, Meiping Wu, Hui Zhang, Huimin Lu, Junhao Xiao, Yaonan Wang, Jianhao Tan. I thank them for preparing me for the doctoral studies and helping me to apply for the scholarship. Thanks for their support all the time.

Last but not least, I thank my family, my parents Huajun Chen and Dizhen Xie, and my wife Jing Zhou. Thank them for their unconditional and endless support and encouragement to my everything.

The work presented in this thesis is partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob, by the EC within Horizon Europe, grant agreemenet no. 101017008 (Harmony), and by the Chinese Scholarship Committee.

Contents

Chapter 1

Introduction

THE prospect of truly autonomous systems captured many people's imaginations with tremendous advances in artificial intelligence and the emergence of deep learning in physical systems: rescue robots can help with urban searching and rescuing; assistive healthcare robots are contributing to just-in-time delivery tasks and the automation of hospital bioassay sample flow; agriculture ground-aerial collaborative robots are dedicated to sustainable crop production; self-driving cars are expected to improve road safety and efficiency. Autonomous mobile systems deployed in different complex environments need to understand the surroundings via online perception to operate safely and reliably. Most existing systems deployed today make relatively little use of explicit high-level semantic information and typically only consider geometric or appearance information of environments. Based on such information, classical methods are inadequate for providing the needed level of safety assurances, especially for safety-critical applications such as autonomous driving. A richer and high-level understanding of the world, namely semantic information, is required for robust and safe deployments of autonomous vehicles in dynamic and complex real-world scenarios.

Multiple critical components are required for most autonomous mobile systems to achieve full autonomy. Simultaneous localization and mapping (SLAM) and global localization within given maps are among the longest-running research areas, as they are one of the first perception steps for mobile systems to operate in real-world scenarios. Both, SLAM and global localization aim to understand how the environment looks like and estimate the location of the robot in the environment. The difference between them is that localization often calculates the robot's location, assuming a given map of the environment. In contrast, for SLAM, the robot needs to build the map online and localize itself within the map simultaneously.

One popular way to compute the robot's location is to find reliable landmarks in the map. Most existing methods work well by exploiting geometric and appearance information in the indoor environment with the static world assumption, which means that the world remains unchanged as the robot moves through it, and the landmarks are all static. This assumption typically holds for short-term indoor applications, but it does not hold for real-world autonomous driving, where the application scenarios are highly dynamic, and long-term reliability is essential. A high-level understanding of the environments, such as semantic information, is undoubtedly required to better deal with dynamic environments and improve the robustness of SLAM and localization.

Semantic scene understanding has grown rapidly in the past decade with the advent of deep learning techniques and neural networks. For autonomous driving, semantic segmentation is one of the popular ways to obtain semantic information about the environment. It exploits the online perception data and assigns a class label to each data point in the input modality. One of the popular sensor modalities for autonomous vehicles in outdoor environments is the light detection and ranging (LiDAR) scanner. Due to their highly-accurate range measurements and robustness to light condition changes, there are multiple LiDAR-based semantic segmentation approaches [38, 101, 120] available for autonomous driving applications. Most of them operate online and provide accurate point-wise semantic predictions.

Despite good semantic segmentation results that have been achieved, how to use such semantics for autonomous driving downstream tasks, such as SLAM and localization, is still under research. One of the open questions is about explicitly or implicitly using semantics: should we design algorithms to explicitly exploit the represented semantic information, or do we enable an algorithm to implicitly learn task-relevant semantic concepts? Another question is which types of semantic classes are more useful for specific tasks since certain semantic categories in the scene can be more helpful than the others depending on the underlying application scenarios. The training data problem is also frequently discussed since most learning-based methods currently need a large number of manual labels, which is very costly and labor-intensive. In summary, the key questions in the context of using semantics for online perception tasks of autonomous vehicles are:

- How to use semantic segmentation results for downstream perception tasks?
- Which types of semantics are useful for specific perception tasks?
- How to generate semantic labels for training networks to learn semantics?

This thesis aims to give answers to these key questions and is divided into three parts. In Part I, we exploit the semantic information provided by existing

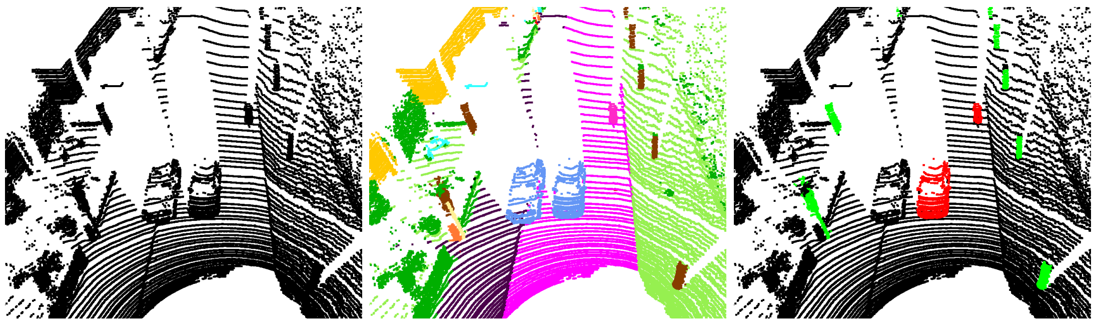


Figure 1.1: Semantic understanding of the environment. The left figure shows the raw LiDAR point cloud. The figure in the middle illustrates the point cloud after assigning semantic information to every point. The right figure illustrates different task-specific semantics, including moving objects in red for SLAM and pole-like objects for localization in green.

semantic segmentation networks and tackle the first question. We provide examples of using semantics both explicitly for 3D LiDAR SLAM and implicitly for loop closing and global localization. In Part II, we address the second question. We further provide examples of using task-specific semantics to improve the performance of SLAM and localization. Finally, in Part III, we present our methods that automatically generate labels for training segmentation networks to answer the third question.

1.1 Main Contributions

The main contributions of this thesis are novel approaches that exploit semantic information to improve the performance of LiDAR perception tasks such as SLAM and localization for autonomous vehicles. These methods use the semantic information from segmentation networks operating online on LiDAR data. We first provide brief introductions to existing basic techniques used in this thesis in Chapter 2, including LiDAR data processing, geometric-based LiDAR SLAM, classical Monte Carlo localization (MCL), and existing semantic segmentation networks. Those techniques are the foundation of our approaches, and a basic understanding of them is key to understanding our contributions.

To answer three key questions about how to use semantic information to improve the performance of LiDAR perception tasks, we divide the thesis into three parts.

Part I provides examples that exploit the multiclass semantic information provided by the existing semantic segmentation networks as shown in the middle of Figure 1.1 to improve the performance of LiDAR-based perception tasks. Chapter 3 presents a novel semantic-based LiDAR SLAM approach for building semantic maps of the environments. It explicitly checks the semantic consistencies between scans and the map to filter out dynamic objects and provide high-level constraints during the pose estimation process. The proposed semantic-enhanced SLAM generates more consistent maps with semantic information and more accurate poses than the geometric-based method.

Instead of explicitly using the consistency of semantic classes, Chapter 4 proposes a novel neural network that implicitly exploits different types of information generated from LiDAR scans to provide similarity estimates between pairs of 3D LiDAR scans. The proposed network uses both geometric and semantic information of pairs of LiDAR scans as input and estimates the similarity between them in an end-to-end fashion. We integrate the network into downstream tasks, like loop closure detection and global localization, and improve their performance.

Instead of exploiting general semantic information from existing segmentation networks, Part II aims to answer the questions of which types of semantics are more useful for different tasks. In Chapter 5, we introduce an example of using moving/non-moving semantics for the SLAM task. Instead of detecting all potentially movable objects such as vehicles or humans, the proposed method aims at separating the actually moving objects such as driving cars from static or non-moving objects such as buildings, parked cars, etc. As shown in the right part of Figure 1.1, the red points represent moving objects, and the rest are static points. It exploits sequential and temporal information to achieve an effective moving object segmentation (MOS), which is then used to improve the pose estimation and mapping results of SLAM.

Unlike SLAM estimating the vehicle's poses with respect to the on-the-fly map, localization aims to localize the robot within a given map. Thus, for localization, pole-like objects, such as traffic signs, poles, lamps, etc., are frequently used landmarks in urban environments due to their local distinctiveness and long-term stability, as highlighted in green in the right part of Figure 1.1. Chapter 6 presents a novel, accurate, and fast pole extraction approach that runs online and has little computational demands, which can be used for a localization system. It proposes to use pole-like objects as landmarks for MCL and achieve good localization results.

One of the bottlenecks in supervised learning approaches is the necessary amount of labeled data. Labeling training data for such approaches is a laborious task and thus expensive. In Part III, we present two examples to automatically generate labels for LiDAR-MOS and pole segmentation, which enable us to train segmentation networks in a self-supervised way and further improve the performance and generalization of SLAM and localization. By specifying and simplifying the categories of semantics into moving and non-moving, we turn the challenging multiclass semantic segmentation problem into an easier binary classification task, which makes automatic label generation for LiDAR-MOS feasible. In Chapter 7, we propose a method that can exploit the temporal-spatial dependence of the recorded sequential data and automatically generate labels for LiDAR-MOS. Based on such automatically generated labels, we can train segmentation networks in a self-supervised way to alleviate the lack of labeled data and improve the performance of LiDAR-MOS.

Similarly, in Chapter 8, we directly use our geometric-based pole extractor presented in Chapter 6 to automatically generate pseudo pole labels on LiDAR point clouds. Then, we train a learning-based pole segmentation network in a self-supervised fashion. This saves the costly human labeling effort and improves the generalization of the learning-based network by providing training data from different environments. In the end, we integrate our pole segmentation network into MCL to achieve LiDAR-based localization in different environments.

In Chapter 9, we discuss and compare existing approaches in the field of perception for autonomous vehicles related to this thesis. In Chapter 10, we finally present our conclusion and provide outlooks on potential future works for the perception tasks of autonomous vehicles.

Overall, this thesis presents novel approaches to exploit semantic information to improve the performance of LiDAR perception tasks such as SLAM and localization for autonomous vehicles. It provides studies of how to use semantics and which types of semantics to use for SLAM and localization. We have made implementations of all the methods presented in this thesis open-source to facilitate further research. The links to each implementation are listed in Section 10.1.

1.2 Publications

Parts of this thesis have been published in the following peer-reviewed conference and journal articles, for which I have been the main contributor:

- 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
- 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
- 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
- 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*, 46:61–81, 2021
- X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss. Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data. *IEEE Robotics and Automation Letters (RA-L)*, 6:6529–6536, 2021
- 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
- H. Dong*, X. Chen*, and C. Stachniss. Online Range Image-based Pole Extractor for Long-term LiDAR Localization in Urban Environments. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2021. (*authors contributed equally.)
- X. Chen, B. Mersch, L. Nunes, R. Marcuzzi, I. Vizzo, J. Behley, and C. Stachniss. Automatic Labeling to Generate Training Data for Online LiDAR-Based Moving Object Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):6107–6114, 2022
- H. Dong*, X. Chen*, S. Särkkä, and C. Stachniss. Online Pole Segmentation on Range Images for Long-term LiDAR Localization in Urban Environments. *Journal on Robotics and Autonomous Systems (RAS)*, 2022. (*authors contributed equally.)

Different approaches of this work were part of different collaborations, which we acknowledge in the individual chapters, and have led to the following peer-reviewed conference and journal articles:

- W. Deng, K. Huang, X. Chen, Z. Zhou, C. Shi, R. Guo, and H. Zhang. Semantic RGB-D SLAM for Rescue Robot Navigation. *IEEE Access*, 8:221320–221329, 2020
- 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
- B. Mersch, X. Chen, J. Behley, and C. Stachniss. Self-supervised Point Cloud Prediction Using 3D Spatio-temporal Convolutional Networks. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2021
- M. Arora, L. Wiesmann, X. Chen, and C. Stachniss. Mapping the Static Parts of Dynamic Scenes from 3D LiDAR Point Clouds Exploiting Ground Segmentation. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2021
- 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. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):738–745, 2022
- B. Mersch, X. Chen, I. Vizzo, L. Nunes, J. Behley, and C. Stachniss. Receding Moving Object Segmentation in 3D LiDAR Data Using Sparse 4D Convolutions. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7503–7510, 2022
- J. Ma, J. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen. OverlapTransformer: An Efficient and Yaw-Angle-Invariant Transformer Network for LiDAR-Based Place Recognition. *IEEE Robotics and Automation Letters*, 7(3):6958–6965, 2022
- M. Arora, L. Wiesmann, X. Chen, and C. Stachniss. Static Map Generation from 3D LiDAR Point Clouds Exploiting Ground Segmentation. *Journal on Robotics and Autonomous Systems (RAS)*, 2022
- J. Sun, Y. Dai, X. Zhang, J. Xu, R. Ai, W. Gu, C. Stachniss, and X. Chen. Efficient Spatial-Temporal Information Fusion for LiDAR-Based 3D Moving Object Segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022

The following are publications I was involved in during my doctorate as a collaborator but they are not part of this thesis:

- C. Shi, X. Chen, K. Huang, J. Xiao, H. Lu, and C. Stachniss. Keypoint Matching for Point Cloud Registration using Multiplex Dynamic Graph Attention Networks. *IEEE Robotics and Automation Letters (RA-L)*, 6:8221–8228, 2021

- 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)*, 6:2060–2067, 2021

- A. Reinke, X. Chen, and C. Stachniss. Simple But Effective Redundant Odometry for Autonomous Vehicles. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021

- M. Zhou, X. Chen, N. Samano, C. Stachniss, and A. Calway. Efficient localisation using images and openstreetmaps. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021

- L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2116–2123, 2022

- T. Guadagnino, X. Chen, M. Sodano, J. Behley, G. Grisetti, and C. Stachniss. Fast Sparse LiDAR Odometry Using Self-Supervised Feature Selection on Intensity Images. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7597–7604, 2022

- L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2116–2123, 2022

- J. Sun, Y. Wang, M. Feng, D. Wang, J. Zhao, C. Stachniss, and X. Chen. ICK-Track: A Category-Level 6-DoF Pose Tracker Using Inter-Frame Consistent Keypoints for Aerial Manipulation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022

- S. Yang, L. Zheng, X. Chen, L. Zabawa, M. Zhang, and M. Wang. Transfer Learning from Synthetic In-vitro Soybean Pods Dataset for In-situ Segmentation of On-branch Soybean Pod. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*, 2022

Chapter 2

Basic Techniques

We motivated in Chapter 1 why it is key for autonomous vehicles to have semantic scene understanding capabilities. We also discussed different types of semantics that can be used for different perception tasks of autonomous vehicles. To be more specific, in this thesis, we exploit semantic information that can be estimated from LiDAR data by semantic segmentation networks to improve LiDAR perception tasks such as SLAM and localization. Therefore, a basic understanding of them is key to understanding our contributions. In this chapter, we focus on introducing the basic techniques that are related to the development of this thesis, including LiDAR data processing in Section 2.1, Monte Carlo localization in Section 2.2, Surfel-based LiDAR SLAM in Section 2.3, and LiDAR-based semantic segmentation in Section 2.4.

2.1 LiDAR Data Processing

The light detection and ranging (LiDAR) sensor, has become a commonly used sensor component for most robotics applications over the past two decades. Laser beams inside the LiDAR unit emit pulsed light waves into the surrounding environment and use the time for each pulse to return to calculate the distance each pulse travels. Repeating this process millions of times per second creates precise, real-time 3D measurements of an environment.

There are mainly two types of LiDAR sensors in the market for robotics applications, mechanically rotating LiDAR and solid-state LiDAR. This thesis focuses on mechanically rotating LiDAR as it is the more common type of LiDAR for autonomous vehicles to this date. Therefore, from now on, we refer to mechanically rotating LiDAR as LiDAR. We consider the data generated by one turn of such LiDAR sensors as one LiDAR *scan* containing measurements during a full 360-degree sweep of an environment.

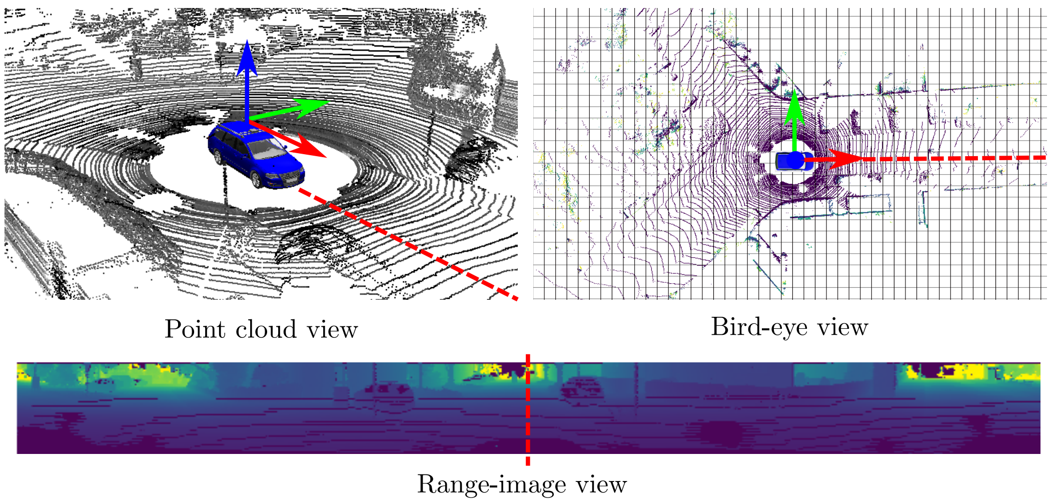


Figure 2.1: Different representations for LiDAR data. The red, green, and blue arrows represent the x-axis, y-axis, and z-axis. The red dash lines show the corresponding zero-yaw view angles between the point cloud view, bird-eye view, and range-image view. Colors in the range-image view varying from purple to yellow correspond to the range of each measurement. In the bird-eye view, the colors present different heights of the measurements.

There are multiple ways to represent one LiDAR scan for subsequent processing. The most popular three representations are point cloud view, bird-eye view, and range image view, as illustrated in Figure 2.1.

The point cloud is the most typical data product of most LiDAR instruments. The raw format of the measurements from a LiDAR is just a collection of range measurements and sensor orientation parameters. After initial processing, the range and orientation for each laser shot are converted into a position in the local 3D coordinate system of the LiDAR sensor as follows:

$$\begin{aligned} x &= r\cos(\alpha)\sin(\theta)\,, \\ y &= r\cos(\alpha)\cos(\theta)\,, \\ z &= r\sin(\alpha)\,, \end{aligned} \tag{2.1}$$

where $(x, y, z)^\top$ is the 3D coordinates of a LiDAR point, r is the corresponding range measurement, and θ, α are azimuth and inclination orientation angles as shown in Figure 2.2. This leads to a cloud of points, which is the base subsequent processing and analysis. However, a typical rotating 3D LiDAR sensor usually generates hundreds of thousand points in one frame, which is very resource-consuming to process. Moreover, due to the distance-dependent sparsity and unordered feature of the raw point cloud, it is usually difficult to be directly exploited by the downstream tasks. Especially for the methods exploiting neural networks, processing 3D point cloud is still very challenging and not available for applications in outdoor large-scale such as autonomous driving.

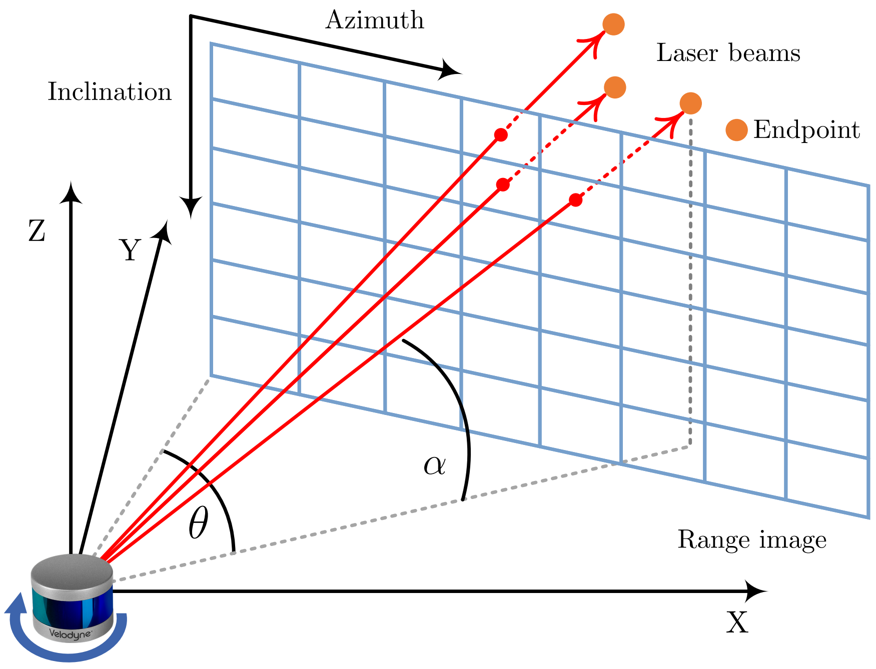


Figure 2.2: Illustrating the generation of a range image from a point cloud. This figure is adapted from the work by Fan et al. [59].

The bird-eye view is a 2D representation of a LiDAR scan, which projects the point cloud into a grid map in the x-y plane, where each grid stores the height information of the corresponding point. Since the bird-eye view is very lightweight and provides a top-down overview of the environment, it is frequently used for object detection [96], motion planning [121], and tracking [205]. However, it introduces quantization error when dividing the space into voxels or pillars, which may remove the distant objects that only have a few points. Furthermore, the bird-eye view also results in the loss of points of vertical objects such as buildings, trees, poles, etc., when projected from 3D space to the 2D x-y plane.

The range image view is a natural representation of the scan from a rotating 3D LiDAR such as a Velodyne or Ouster sensor. It can be seen as an intermediate representation in the process of converting LiDAR raw measurements into a point cloud. During the rotating scanning, the LiDAR sensor can generate a 2D matrix given the elevation and azimuth resolutions, where each element in this 2D matrix is a range measure, thus obtaining the so-called range image $\mathcal{R}$. For a LiDAR with h beams and w times shooting in one round of LiDAR scanning, it generates range images with the height of h and width of w as illustrated in Figure 2.2.

For each pixel in range image with a certain pair of elevation and azimuth angles, we can then calculate the 3D coordinates of the corresponding beam-end point using Equation (2.1). The other way around, given a point cloud, there is also a projection transformation to re-generate a range image. Specifically, we convert each LiDAR point $\boldsymbol{p} = (x, y, z)^\top$ via a mapping $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates, and finally to image coordinates, as defined by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\,\pi^{-1}\right] & w \\ \left[1 - \left(\arcsin(z\,r^{-1}) + \mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right] & h \end{pmatrix}, \tag{2.2}$$

where $(u, v)^\top$ are image coordinates, h and 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 $(u, v)^\top$ tuples containing a pair of image coordinates for each point $\boldsymbol{p}_i$.

In this thesis, we use a range projection-based representation to process the LiDAR point cloud data so that we can exploit well-studied 2D convolutional neural networks, which have been widely used for image-based tasks. Furthermore, processing such lightweight and compact range images is more efficient compared to processing point clouds directly, while not losing any raw data information compared to using 2D bird-eye views. Another benefit of using range images together with neural networks is that we can exploit different types of information obtained by LiDAR sensors altogether in an easy way. Using the image indices, we extract for each $\boldsymbol{p}_i$, its range r, its x, y, and z coordinates, and its remission e, and store them channel-wise in the image. Thus, each pixel in this representation stores 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 boosts the performance of our methods and, at the same time, makes our method easily transferable to other new architectures.

2.2 Monte Carlo Localization

Monte-Carlo localization or MCL is a localization algorithm based on the particle filter proposed by Dellaert et al. [44]. As used multiple times in this thesis, we briefly illustrate the most important parts of MCL in this section.

A particle filter is a nonparametric implementation of the Bayes filter [178] and is frequently used to estimate the state of a dynamic system. The key idea is to represent a posterior by a set of hypotheses $\mathcal{H}$ with different importance weights, which can be represented as:

$$\mathcal{H} = \left\{ \left(\boldsymbol{x}^i, w^i\right) \mid i = 1, \ldots, N \right\}, \tag{2.3}$$

where $\boldsymbol{x}$ is the state vector of the dynamic system, w is the corresponding importance weight, and N is the number of particles.

In localization scenario, we use each particle to represent a hypothesis for the robot's or autonomous vehicle's 2D pose $\boldsymbol{x}_t = (x, y, \theta)_t^\top$ at time t. Therefore, the

Algorithm 1: The Monte Carlo localization.

```
   Input: Particle set $\mathcal{H}_{t-1}$, control input $\boldsymbol{u}_t$ and observation $\boldsymbol{z}_t$
   Output: Updated particle set $\mathcal{H}_t$
 1 $\mathcal{H}'_t = \emptyset$ // intermediate set of particle proposals
 2 foreach $i \in \{1, \ldots, N\}$ do
 3     sample $\hat{\boldsymbol{x}}^i_t \sim p(\boldsymbol{x}_t \mid \boldsymbol{x}^i_{t-1}, \boldsymbol{u}_t)$
 4     $\hat{w}^i_t = \eta \cdot p(\boldsymbol{z}_t \mid \hat{\boldsymbol{x}}^i_t)$ // where $\eta$ is a normalizer
 5     $\mathcal{H}'_t = \mathcal{H}'_t \cup \{(\hat{\boldsymbol{x}}^i_t, \hat{w}^i_t)\}$
 6 end
 7 $\mathcal{H}_t = \emptyset$
 8 foreach $j \in \{1, \ldots, N\}$ do
 9     resample the $j$-th particle of $\mathcal{H}'_t$ with probability $w^j_t$
10     $\mathcal{H}_t = \mathcal{H}_t \cup \{(\boldsymbol{x}^j_t, w^j_t)\}$
11 end
```

particle filter algorithm allows us to track multiple hypotheses of the robot's pose by recursively estimating the particle set $\mathcal{H}_t$ based on the estimate $\mathcal{H}_{t-1}$ of the previous time step. As shown in Algorithm 1, it also takes the control command $\boldsymbol{u}_t$ and the current observation $\boldsymbol{z}_t$ as input, and the MCL can be summarized with the following three steps: sampling by motion model, importance weighting by observation model, and resampling the particles. In the first step, the pose of each particle is updated with a prediction based on a motion model with the control input $\boldsymbol{u}_t$, which can be formulated as sampling from the motion model density $p(\boldsymbol{x}_t \mid \boldsymbol{x}^i_{t-1}, \boldsymbol{u}_t)$ shown in line 3. Secondly, the expected observation from the predicted pose of each particle is then compared to the actual observation $\boldsymbol{z}_t$ acquired by the robot to update the particle's weight based on the observation model shown in line 4. In doing so, it creates the next generation $\mathcal{H}'_t$ of particles based on the previous set $\mathcal{H}_{t-1}$ of samples shown in line 5. In the end, the resampling selects N samples from the set $\mathcal{H}'_t$ with higher probability samples. The new set $\mathcal{H}_t$ is given by the drawn particles according to their weight distribution in line 9. In the real application, the resampling is triggered whenever the effective number N_{eff} of particles drops below 50% of the sample size for the sake of efficiency and avoiding particle depletion. N_{eff} can be calculated adaptively:

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (\bar{w}^i)^2}\,, \tag{2.4}$$

where $\bar{w}^i$ refers to the normalized weight of particle i. Using N_{eff} to trigger the resampling, MCL can approximate the target distribution with low variance. See [70] for details.

In brief, MCL realizes a recursive Bayesian filtering scheme. The key idea of

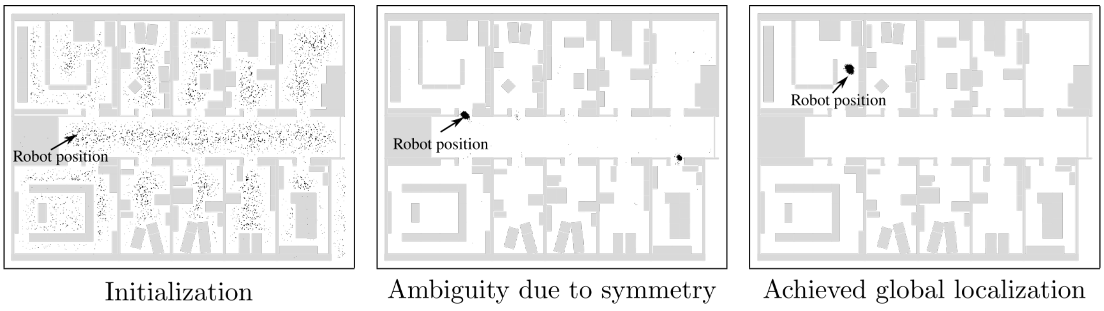


Figure 2.3: An example of the Monte Carlo localization results from initialization to achieving global localization. This figure is adapted from the original MCL paper by Dellaert et al. [44]. The original data was recorded over 20 years ago at the University of Bonn.

this approach is to maintain a probability density $p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t})$ of the pose $\boldsymbol{x}_t$ at time t given all observations $\boldsymbol{z}_{1:t}$ up to time t and motion control inputs $\boldsymbol{u}_{1:t}$ up to time t. This posterior is updated as follows:

$$p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) = \eta \, p(\boldsymbol{z}_t \mid \boldsymbol{x}_t) \int p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}) \, p(\boldsymbol{x}_{t-1} \mid \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t-1}) \, d\boldsymbol{x}_{t-1}, \tag{2.5}$$

where η is the normalization constant resulting from Bayes rule, $p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ is the motion model, and $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$ is the observation model.

As shown in Figure 2.3, MCL uses a set of weighted particles and can represent arbitrary distributions. The particles are drawn from a proposal distribution. After determining the importance weights, which account for the fact that the target distribution is different from the proposal distribution, the resampling step replaces low-weight particles with high-weight ones. These three steps are repeated recursively, and the particles are likely to converge around the true pose after several iterations of this recursive procedure.

2.3 Surfel-Based LiDAR SLAM

Simultaneous localization and mapping (SLAM) is also regarded as one of the most important problems in building real autonomous mobile robots and is the other main application that this thesis focuses on. Here, we briefly introduce the surfel-based LiDAR SLAM (SuMa) originally proposed by Behley and Stachniss [15], which is used as the basic SLAM pipeline of this thesis.

Different from localization, where a map of an environment is usually given, SLAM tackles the problem of acquiring a spatial map of an environment while simultaneously localizing the robot within this map. There have been significant advances in SLAM based on camera images [57, 125] and RGB-D data [41, 195]

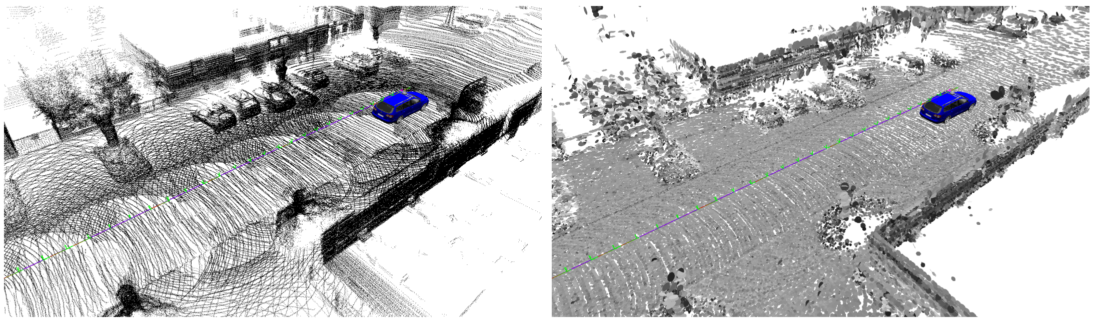


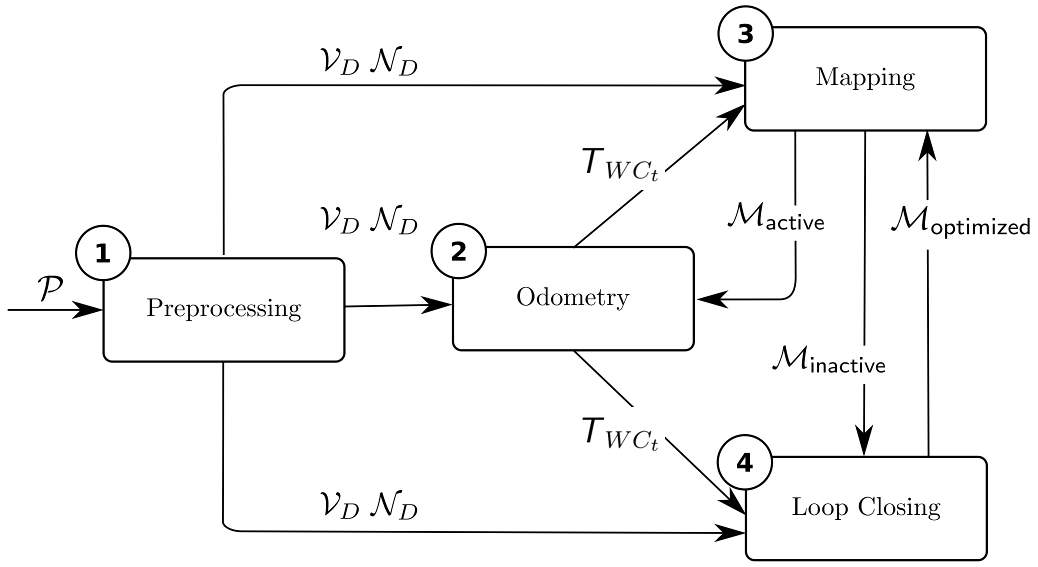
Figure 2.4: Visualization of maps from SuMa. The left figure shows the raw point clouds, and the right one shows the surfel-based map generated by SuMa. The blue car represents the current position of the vehicle and the purple line is the trajectory.

over the past few years. However, outdoor LiDAR-based SLAM is still challenging due to the sparsity of LiDAR data and dynamic environments. Instead of using feature-based solutions [69, 212], in this thesis, we choose SuMa as the baseline SLAM method to build a dense map of the environment and, at the same time, estimate the pose of the autonomous vehicle, as shown in Figure 2.4. The left figure shows the aggregated raw point clouds, and the right figure visualizes the surfel-based map generated by SuMa. As can be seen, surfels are small oriented disks or surface elements, which maintain dense, detailed geometric information of the point clouds. A surfel map $\mathcal{M}$ is an unordered set of surfels s, where each surfel is defined by a position $\boldsymbol{v}_s \in \mathbb{R}^3$, a normal $\boldsymbol{n}_s \in \mathbb{R}^3$, a radius $r_s \in \mathbb{R}$, and also a stability log odds ratio l_s maintained with a binary Bayes filter [178] to determine if a surfel is stable or not.

In the following, we briefly introduce key components of SuMa. To make the notations clear, we denote the transformation of a point $\boldsymbol{p}_a$ in LiDAR scan C_a to a point $\boldsymbol{p}_b$ in LiDAR scan C_b by $\mathsf{T}_{ba} \in \mathbb{R}^{4\times4}$, such that $\boldsymbol{p}_b = \mathsf{T}_{ba}\,\boldsymbol{p}_a$. We denote $\mathsf{R}_{ba} \in \mathrm{SO}(3)$ and $\boldsymbol{t}_{ba} \in \mathbb{R}^3$ as the corresponding rotational and translational part of transformation T_{ba}. This transformation can be equivalently expressed by first applying a rotation $\mathsf{R}_{ba} \in \mathrm{SO}(3)$ and then a translation $\boldsymbol{t}_{ba} \in \mathbb{R}^3$, i.e., $\boldsymbol{p}_b = \mathsf{T}_{ba}\,\boldsymbol{p}_a = \mathsf{R}_{ba}\,\boldsymbol{p}_a + \boldsymbol{t}_{ba}$. We call the currently observed LiDAR scan C_t at timestep t and a rendered world frame C_m from our map representation at a given coordinate frame, where $m \in \{0, \dots, t\}$. The goal is to find a global transformation $\mathsf{T}_{WC_t} \in \mathbb{R}^{4\times4}$ to associate the currently observed point cloud in frame C_t to the world coordinate frame W. To this end, we calculate the global transformation by applying recursively the pose changes $\mathsf{T}_{C_mC_t}$ given the rendered model at $\mathsf{T}_{WC_{t-1}}$, i.e.,

$$\begin{aligned} \mathsf{T}_{WC_t} &= \mathsf{T}_{WC_0}\,\mathsf{T}_{C_0C_1}\cdots\mathsf{T}_{C_{t-2}C_{t-1}}\,\mathsf{T}_{C_mC_t}\,, \\ &= \mathsf{T}_{WC_0}\,\mathsf{T}_{C_0C_1}\cdots\mathsf{T}_{C_{t-1}C_t}\,, \end{aligned} \tag{2.6}$$

where we assume T_{WC_0} to be the identity $\mathsf{Id} \in \mathbb{R}^{4\times4}$.



$\mathcal{V}_D\ \mathcal{N}_D$
3
Mapping
T_{WC_t}
$\mathcal{V}_D\ \mathcal{N}_D$
$\mathcal{M}_{\mathsf{active}}$
$\mathcal{M}_{\mathsf{optimized}}$
1
2
$\mathcal{P}$
Preprocessing
Odometry
$\mathcal{M}_{\mathsf{inactive}}$
T_{WC_t}
4
$\mathcal{V}_D\ \mathcal{N}_D$
Loop Closing

Figure 2.5: Overview of SuMa. There are four main modules in SuMa, including (1) preprocessing, (2) odometry, (3) mapping, and (4) loop closing. This figure is adapted from the original SuMa paper by Behley and Stachniss [15].

The overview of SuMa is shown in Figure 2.5. SuMa has four main modules, including preprocessing, odometry, mapping, and loop closing. For each input point cloud $\mathcal{P} = \{\boldsymbol{p} \in \mathbb{R}^3\}$, it estimates the pose T_{WC_t} at timestep t for the LiDAR scan C_t and at the same time update the map using the new observation obtained by frame C_t.

In the preprocessing module, SuMa uses the projection function $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ as introduced in the previous section Equation (2.2) to generate a range image-like vertex image $\mathcal{V}_D : \mathbb{R}^2 \mapsto \mathbb{R}^3$ mapping a 2D image coordinate $(u,v)^\top \in \mathbb{R}^2$ to a point $(x,y,z)^\top \in \mathbb{R}^3$. Each pixel in $\mathcal{V}_D$ contains the 3D coordinates of the corresponding LiDAR point. Given the vertex image $\mathcal{V}_D$, the preprocessing module also computes for each coordinate $(u,v)^\top$ a corresponding normal in the normal image $\mathcal{N}_D$ using cross products over forward differences:

$$\begin{aligned}\mathcal{N}_D((u,v)) = {} & (\mathcal{V}_D((u+1,v)) - \mathcal{V}_D((u,v))) \\ & \times (\mathcal{V}_D((u,v+1)) - \mathcal{V}_D((u,v)))\,. \end{aligned} \tag{2.7}$$

Note that, we convert points $\boldsymbol{p} \in \mathbb{R}^3$ and normals $\boldsymbol{n} \in \mathbb{R}^3$ to corresponding homogeneous coordinates before application of the affine transformation. For the sake of simplicity, we do not show this operation explicitly in the following derivations.

To estimate the odometry, i.e., the transformation $T_{C_{t-1}C_t}$, SuMa also renders a pair of vertex image $\mathcal{V}_M$ and normal image $\mathcal{N}_M$ at the last pose estimate $T_{WC_{t-1}}$ from the current active surfel map $\mathcal{M}_{\mathsf{active}}$. $\mathcal{M}_{\mathsf{active}}$ contains the last Δ_{active} point clouds up to timestep $t-1$ i.e., $t_u \geq t - \Delta_{\mathsf{active}}$, which consists of only recently updated surfels to speed up the rendering. It then aligns the current observation $\mathcal{V}_D$ with the rendered map vertex image $\mathcal{V}_M$ by incrementally minimize the point-to-plane error:

$$E(\mathcal{V}_D, \mathcal{V}_M, \mathcal{N}_M) = \sum_{\boldsymbol{u} \in \mathcal{V}_D} \left(\boldsymbol{n}_u^\top \left(T^{(k)}_{C_{t-1}C_t} \boldsymbol{u} - \boldsymbol{v}_u \right) \right)^2 , \tag{2.8}$$

where each vertex $\boldsymbol{u} \in \mathcal{V}_D$ is projectively associated to a reference vertex $\boldsymbol{v}_u \in \mathcal{V}_M$ and its normal $\boldsymbol{n}_u \in \mathcal{N}_M$ via

$$\boldsymbol{v}_u = \mathcal{V}_M \left(\Pi \left(T^{(k)}_{C_{t-1}C_t} \boldsymbol{u} \right) \right) , \tag{2.9}$$

$$\boldsymbol{n}_u = \mathcal{N}_M \left(\Pi \left(T^{(k)}_{C_{t-1}C_t} \boldsymbol{u} \right) \right) . \tag{2.10}$$

Here, $T^{(k)}_{C_{t-1}C_t}$ corresponds to the current pose estimate of the frame-to-model iterative closest point (ICP) algorithm [19, 80, 148, 195] at iteration k. The classical ICP algorithm [19, 148] has been widely used for point clouds registration. Instead of registering the currently observed LiDAR scan to the previous one, SuMa borrows the ideas from other modern SLAM methods [80, 195] to register the current scan to the map representation, i.e., frame-to-model ICP, to obtain better odometry estimates. For outlier rejection, SuMa filters out correspondences exceeding a distance of δ_{ICP} or having an angle difference larger than θ_{ICP} between $\boldsymbol{n}_u$ and the corresponding normal of $\boldsymbol{u}$ in $\mathcal{N}_D$. It initializes the ICP pose $T^{(0)}_{C_{t-1}C_t} = T_{C_{t-2}C_{t-1}}$ with the last pose increment to warm start the optimization. The yielded $T_{C_{t-1}C_t}$ is then used to update the global pose T_{WC_t} via Equation (2.6).

With the current pose T_{WC_t}, SuMa updates its surfel map $\mathcal{M}_{\mathsf{active}}$ by initializing surfels for previously unseen areas, but also refining surfels of already covered areas. It starts by computing the radius r_s of potential new surfel s for each $\boldsymbol{v}_s \in \mathcal{V}_D$ and corresponding normal $\boldsymbol{n}_s \in \mathcal{N}_D$ with the aim to cover roughly the corresponding pixel of the vertex map:

$$r_s = \frac{\sqrt{2}\, \|\boldsymbol{v}_s\|_2\, p}{\operatorname{clamp}(-\boldsymbol{v}_s^\top\, \boldsymbol{n}_s\, \|\boldsymbol{v}_s\|_2^{-1}, 0.5, 1.0)} , \tag{2.11}$$

where $p = \max(w\, f_{\text{horiz}}^{-1}, h\, f_{\text{vert}}^{-1})$ corresponds to the pixel size and $\operatorname{clamp}(x, l, u) = \min(u, \max(x, l))$ to the clamping operation. Then, it renders $\mathcal{V}_M$, $\mathcal{N}_M$, 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 T_{WC_t} to determine visible model surfels and their indices for updating. Each measurement point $\boldsymbol{v}_s$ is projected to $\mathcal{I}_M$ using Equation (2.2) to find the corresponding model surfel s'. For updating the surfel map, SuMa first determines if the data surfel s is compatible with the associated surfel s', i.e., it holds $|\boldsymbol{n}_{s'}^\top(\boldsymbol{v}_s - \boldsymbol{v}_{s'})| < \delta_M$ and $\|\boldsymbol{n}_s \times \boldsymbol{n}_{s'}\|_2 < \sin(\theta_M)$. If the data surfel is compatible, SuMa increases the stability of the associated model surfel. If the measurement is more precise, i.e., $r_s < r_{s'}$, SuMa updates the model surfel using an exponential moving average:

$$\boldsymbol{v}_{s'}^{(t)} = (1-\gamma)\,\boldsymbol{v}_s + \gamma\,\boldsymbol{v}_{s'}^{(t-1)}\,, \tag{2.12}$$

$$\boldsymbol{n}_{s'}^{(t)} = (1-\gamma)\,\boldsymbol{n}_s + \gamma\,\boldsymbol{n}_{s'}^{(t-1)}\,, \tag{2.13}$$

$$r_{s'}^{(t)} = r_s\,. \tag{2.14}$$

Otherwise, it decreases the stability of the associated model surfel s' and initializes a new surfel. If a measurement cannot be assigned to any existing surfel, it initializes a new surfel.

For loop closing, SuMa searches for a potential loop closure in the inactive map $\mathcal{M}_{\mathsf{inactive}}$ and tries to align the current measurements with the map. From all frames in $\mathcal{M}_{\mathsf{inactive}}$, $T_{WC_0}, \ldots, T_{WC_{t-\Delta_{\mathsf{Active}}}}$, SuMa considers only one candidate in a radius δ_{loop} of the current pose, i.e., $j^* = \arg\min_{j\in 0,\ldots,t-\Delta_{\mathsf{Active}}} \|\boldsymbol{t}_{WC_t} - \boldsymbol{t}_{WC_j}\|_2$. For a candidate j^*, it tries to align the current point cloud to the rendered view at the corresponding pose $T_{WC_{j^*}}$ using the frame-to-model ICP. Since ICP heavily depends on the initialization, it checks multiple initializations $T^{(0)}_{C_{j^*}C_t}$ with respect to the old pose to compensate for rotational and translational drift. Given the transformation between the current point cloud and the potential loop closure, it transforms both into a common coordinate frame and generates 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 $\mathcal{V}_D$, it adds entries to the composed maps if the point in $\mathcal{V}_D$ is closer than the existing point in $\mathcal{V}_C$. It then computes $E(\mathcal{V}_D, \mathcal{V}_C, \mathcal{N}_C)$, which is called map residual E_{map}, in respect to the composed vertex set $\mathcal{V}_C$ and $\mathcal{N}_C$. Only if this composed view is consistent with the current measurements, SuMa considers 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., $E_{\mathsf{map}} < \epsilon_{\mathsf{residual}}\, E_{\mathsf{odom}}$, and if there are enough inliers and valid points in the rendered composed view. Once a loop closure candidate at timestep t is found, SuMa then tries to verify it in the subsequent timesteps $t+1, \ldots, t+\Delta_{\mathsf{verification}}$, which ensures that it only adds consistent loop closures to the pose graph. 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 denoted as $\mathcal{M}_{\mathsf{optimized}}$.

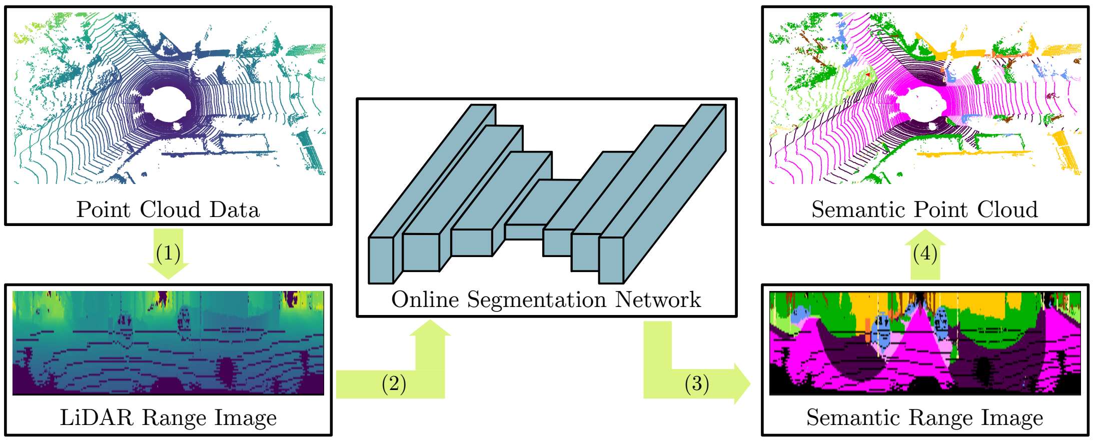


Figure 2.6: Illustrating the LiDAR range image-based semantic segmentation framework. It first projects a point cloud into a LiDAR range image. Then, it uses a neural network to estimate the semantic label for each pixel in the range image. In the end, it assigns the semantic label to the corresponding point via back-projection or pixel-point association. This figure is adapted from the original RangeNet++ paper by Milioto et al. [120].

2.4 LiDAR-Based Semantic Segmentation

Scene understanding is a key building block of autonomous vehicles operating in dynamic environments. One of the important tasks in scene understanding is semantic segmentation, which assigns a class label to each data point in the input modality, i.e., a 3D point obtained by a LiDAR sensor. This section introduces the basic LiDAR-based semantic segmentation framework frequently used in developing different perception tasks in this thesis.

Most state-of-the-art methods currently available for semantic segmentation on point cloud data either do not 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 for supporting autonomous vehicles, since offline processing is not possible for most tasks pertaining to autonomous vehicles, such as affordance analysis of the scene, localization, obstacle avoidance, etc. In this thesis, we, therefore, choose range-image-based methods [38, 101, 120] to conduct semantic segmentation on point cloud data for online autonomous vehicle applications.

The state-of-the-art methods for semantic segmentation are based on deep convolutional neural networks (CNNs). There are usually four main modules in range-image-based semantic segmentation networks, as shown in Figure 2.6, (1) projecting the input point cloud into a range image representation, (2) feeding range image into a deep-learning-based semantic segmentation neural network, (3) semantically segmenting the range image, and (4) back-projecting the semantic labels from the semantic range image to the point cloud yielding the point-wise semantic labels.

In the following, we briefly introduce each step of a typical range-image-based semantic segmentation method. As introduced in Section 2.1, the range image representation is a natural representation of the scan from a rotating 3D LiDAR such as a Velodyne or Ouster sensor. It can be seen as an intermediate representation in the process of converting LiDAR raw measurements into a point cloud. Therefore, a range image pixel is one-to-one corresponds to point cloud point via the projection function as introduced in Equation (2.2). Benefiting from the 2D image-like representation, we can exploit well-studied image-base CNNs, which can be computed fast on a GPU.

There are several existing CNNs that aim at range image-based semantic segmentation [38, 101, 120]. Most of them use an encoder-decoder hour-glass-shaped architecture, which is depicted in the middle of Figure 2.6. These deep hour-glass-shaped segmentation networks have an encoder with significant downsampling, which allows the higher abstraction deep kernels to encode context information while running faster than their non-downsampling counterparts. In this thesis, we focus more on how to use the semantic information obtained by such range-image-based CNNs instead of changing their architecture. Thus, the methods introduced in this thesis are not bound to a specific semantic segmentation network, and we test three off-the-shelf range-image-based semantic segmentation networks, as proposed by Milioto et al. [120], Cortinhal et al. [38], and Li et al. [101]. Milioto et al. [120] propose RangeNet++, which uses the Darknet [140] backbone architecture designed for image classification and object detection tasks and is very descriptive, achieving the state-of-the-art performance on these tasks. RangeNet makes it usable for range images and trains it with semantic labels provided by the SemanticKITTI dataset [12] to achieve semantic segmentation for LiDAR range images. Cortinhal et al. [38] propose SalsaNext using another encoder-decoder backbone, SalsaNet [1]. SalsaNext gains a better performance by exploiting a Bayesian treatment and the prediction of uncertainty for semantic segmentation. In contrast to them, MINet proposed by Li et al. [101] focuses more on the runtime performance and operates much faster than its counterparts on the embedded platforms, which can be deployed for real autonomous driving applications. To achieve this, it uses a multi-scale backbone [145] together with additional dense interactions between different scales avoiding redundant computations. For more details about the network architectures, we refer to the original papers [38, 101, 120].

The output of such image-based CNNs is pixel-wise semantic predictions. To obtain the final point-wise semantic results, such range-image-based methods usually use the back-projection. Since the encoder-decoder CNNs usually provide blurry outputs during inference, it leads to many wrong predictions during the back-projection to the point cloud, especially on the borders of objects. To eliminate such artifacts, most range-image-based methods use a GPU-enabled, k-nearest neighbor (kNN) search operating directly in the input point cloud and use the neighborhood voting to filter out the wrong predictions.

The whole procedure results in semantic labels for each point present in the entire input scan in a lossless way. We will then use such semantic information to improve the performance of perception tasks, such as LiDAR SLAM and localization. We will give detailed information on our proposed approaches and contributions in the following chapters.

Part I

Exploiting Learning-Based Semantic Information for LiDAR Perception

Chapter 3

Semantic LiDAR Odometry and Mapping

ACCURATE localization and reliable mapping of unknown environments are fundamental for most autonomous mobile systems. Such systems often operate in highly dynamic environments, which makes the generation of consistent maps more difficult in real-world scenarios. Semantics provide a richer source of information and a high-level understanding of the mapped area, which 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 at locations that were never seen, and thus not accurately mapped before.

In this chapter, we focus on the problem of how to build an accurate map of the dynamic environment, and at the same time, localize an autonomous vehicle within the map reliably and accurately. We introduce a semantic-based approach to simultaneous localization and mapping using only LiDAR data. We build our approach based on SuMa, a 3D LiDAR SLAM method (see Section 2.3), and incorporate semantic information obtained from a semantic segmentation generated by a fully convolutional neural network, RangeNet++ (see Section 2.4). 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 in outdoor dynamic environments.

For the purpose of building an online semantic SLAM method, we perform highly efficient processing on the LiDAR range image, which is a spherical projection of the point cloud as introduced in Section 2.4. We first use the semantic segmentation neural network to estimate the semantic class labels for each point of the LiDAR scan. Such semantic segmentation results on the 2D spherical projection are then back-projected to the three-dimensional point cloud. However, the back-projection introduces artifacts, especially on the board of objects, which

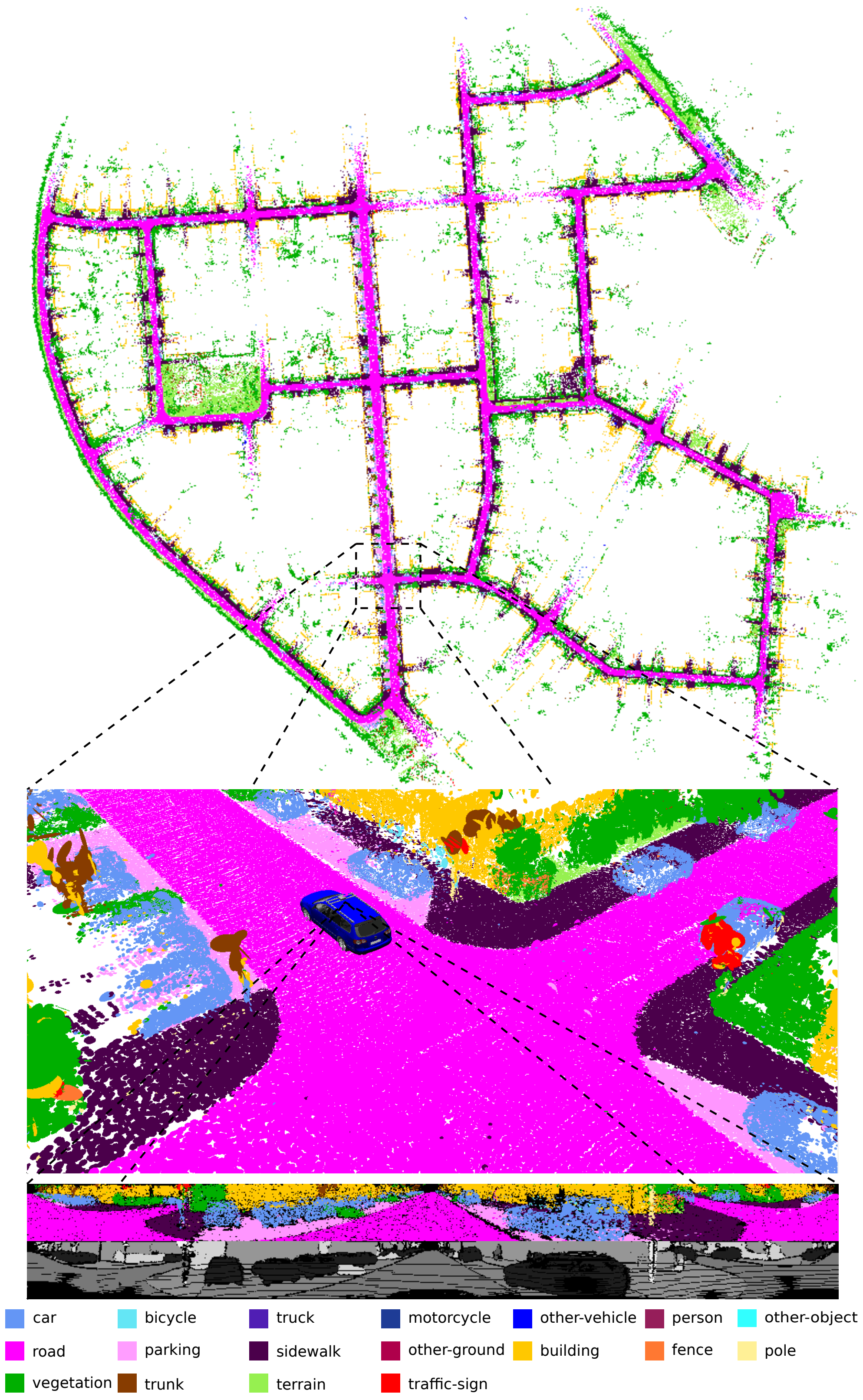


Figure 3.1: Semantic map of the KITTI dataset generated with our approach using only LiDAR scans. The map is represented by surfels that have a class label indicated by the respective color. Overall, our semantic SLAM pipeline is able to provide high-quality semantic maps with a high metric accuracy.

leads to inconsistent semantic mapping results. We propose to reduce the artifacts by a two-step process which uses first an erosion and then recovers with 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 the map. In this way, we reduce the risk of integrating dynamic objects into the map. Figure 3.1 shows an example of our semantic map representation.

The main contribution of this chapter 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 by simply removing possibly moving objects in general environments, including urban, countryside, and highway scenes. We experimentally evaluated our approach on challenging sequences of the KITTI [67] dataset. The evaluation results show the superior performance of our semantic surfel-mapping approach, which we call SuMa++, compared to purely geometric surfel-based mapping and the mapping results by removing all potentially moving objects based on class labels.

3.1 Semantic Surfel-Based SLAM

As illustrated in Figure 3.2, we build our semantic SLAM approach based on SuMa [15] pipeline. We extend it by integrating semantic information provided by a semantic segmentation neural network, named RangeNet++ [120], 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.

3.1.1 Surfel-Based Mapping

We have introduced SuMa [15] in Section 2.3. Here we only briefly recap the main concepts relevant to our semantic-based approach.

Following SuMa, we denote the transformation of a point in a point cloud $\boldsymbol{p}_A \in \mathcal{P}_A$ in coordinate frame A to a point $\boldsymbol{p}_B \in \mathcal{P}_B$ in coordinate frame B by $\boldsymbol{T}_{BA} \in \mathbb{R}^{4\times 4}$, such that $\boldsymbol{p}_B = \boldsymbol{T}_{BA}\boldsymbol{p}_A$. Let $\boldsymbol{R}_{BA} \in \mathrm{SO}(3)$ and $\boldsymbol{t}_{BA} \in \mathbb{R}^3$ denote the corresponding rotational and translational part of transformation $\boldsymbol{T}_{BA}$. We call the coordinate frame at timestep t as C_t. Each variable in coordinate frame

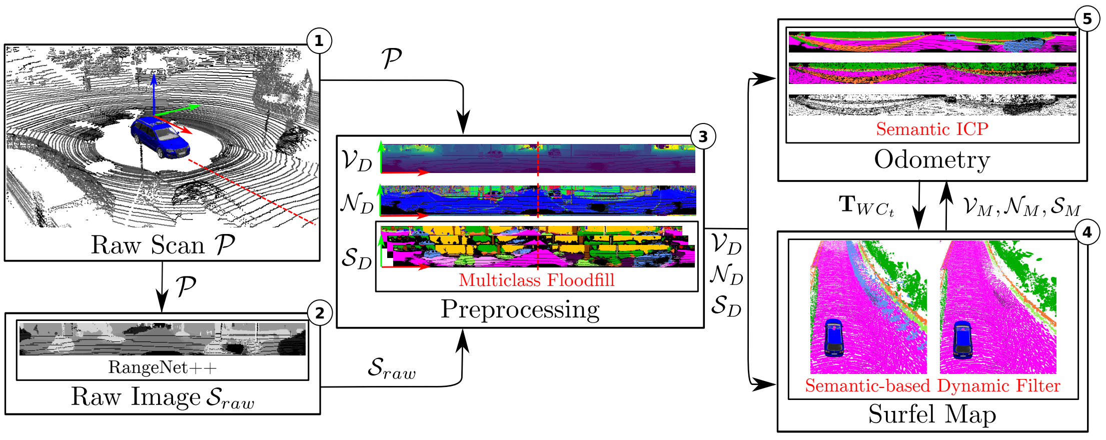


Figure 3.2: Pipeline overview of our proposed approach. We integrate semantic predictions into the SuMa pipeline in a compact way: (1) The input is only the LiDAR scan $\mathcal{P}$. (2) Before processing the raw point clouds $\mathcal{P}$, we first use a semantic segmentation from RangeNet++ to predict the semantic label for each point and generate a raw semantic image $\mathcal{S}_{\text{raw}}$. (3) Given the raw image, we generate a refined semantic image $\mathcal{S}_D$ in the preprocessing module using multiclass 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.

C_t is associated to the world frame W by a pose $T_{WC_t} \in \mathbb{R}^{4\times4}$, transforming the observed point cloud into the world coordinate frame. SuMa first generates a spherical projection of the point cloud $\mathcal{P} = \{\boldsymbol{p} \in \mathbb{R}^3\}$ at timestep t, the so-called vertex image $\mathcal{V}_D$, which contains a group of vertices $\mathcal{V}_D = \{\boldsymbol{u} \in \mathbb{R}^3\}$. Each vertex $\boldsymbol{u}$ stores the 3D coordinates $(x, y, z)^\top$ of the corresponding LiDAR point $\boldsymbol{p}$. The vertex image is then used to generate a corresponding normal image $\mathcal{N}_D = \{\boldsymbol{n} \in \mathbb{R}^3\}$. 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 $T_{C_{t-1}C_t}$ and consequently T_{WC_t} by chaining all pose increments.

The map is represented by surfels, where each surfel is defined by a position $\boldsymbol{v}_s \in \mathbb{R}^3$, a normal $\boldsymbol{n}_s \in \mathbb{R}^3$, 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 [178] to determine if a surfel is considered stable or unstable.

3.1.2 Semantic Segmentation

For each frame, we use RangeNet++ [120] to predict a semantic label for each point and generate a semantic image $\mathcal{S}_D$. As described in Section 2.4, RangeNet++ semantically segments a range image generated by a spherical projection of each

Algorithm 2: Flood-fill for refining $\mathcal{S}_D$.

```
Input: semantic image $\mathcal{S}_{\text{raw}}$ and the corresponding vertex image $\mathcal{V}_D$
Output: refined image $\mathcal{S}_D$
1  Let $\mathcal{N}_{\boldsymbol{s}}$ be the set of neighbors of a pixel in a semantic image $\boldsymbol{s} \in \mathcal{S}$
     within a filter kernel of size $d$. $\epsilon_{\text{rej}}$ is the rejection threshold. $\mathbf{0}$
     represents an empty pixel with label 0.
2  foreach $\boldsymbol{s}_u \in \mathcal{S}_{\text{raw}}$ do
3      Let $\mathcal{S}_{\text{raw}}^{\text{eroded}}(\boldsymbol{u}) = \boldsymbol{s}_u$
4      foreach $\boldsymbol{n} \in \mathcal{N}_{\boldsymbol{s}_u}$ do
5          if $y_{\boldsymbol{s}_u} \neq y_{\boldsymbol{n}}$ then
6              Let $\mathcal{S}_{\text{raw}}^{\text{eroded}}(\boldsymbol{u}) = \mathbf{0}$
7              break
8          end
9      end
10 end
11 foreach $\boldsymbol{s}_u \in \mathcal{S}_{\text{raw}}^{\text{eroded}}$ do
12     Let $\mathcal{S}_D(\boldsymbol{u}) = \boldsymbol{s}_u$
13     foreach $\boldsymbol{n} \in \mathcal{N}_{\boldsymbol{s}_u}$ do
14         if $\|\boldsymbol{u} - \boldsymbol{u}_{\boldsymbol{n}}\|_2 < \epsilon_{\text{rej}} \|\boldsymbol{u}\|_2$ then
15             Let $\mathcal{S}_D(\boldsymbol{u}) = \boldsymbol{n}$ if $y_{\boldsymbol{s}_u} = 0$
16             break
17         end
18     end
19 end
```

laser scan. 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 assign each surfel the inferred semantic label y_u of the corresponding vertex $\boldsymbol{u}$ and the confidence $P(y_u \mid \boldsymbol{u})$ of that label given by the semantic segmentation network.

3.1.3 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 Algorithm 2. It is inside the preprocessing module, which uses depth information from the vertex image $\mathcal{V}_D$ to refine the semantic image $\mathcal{S}_D$.

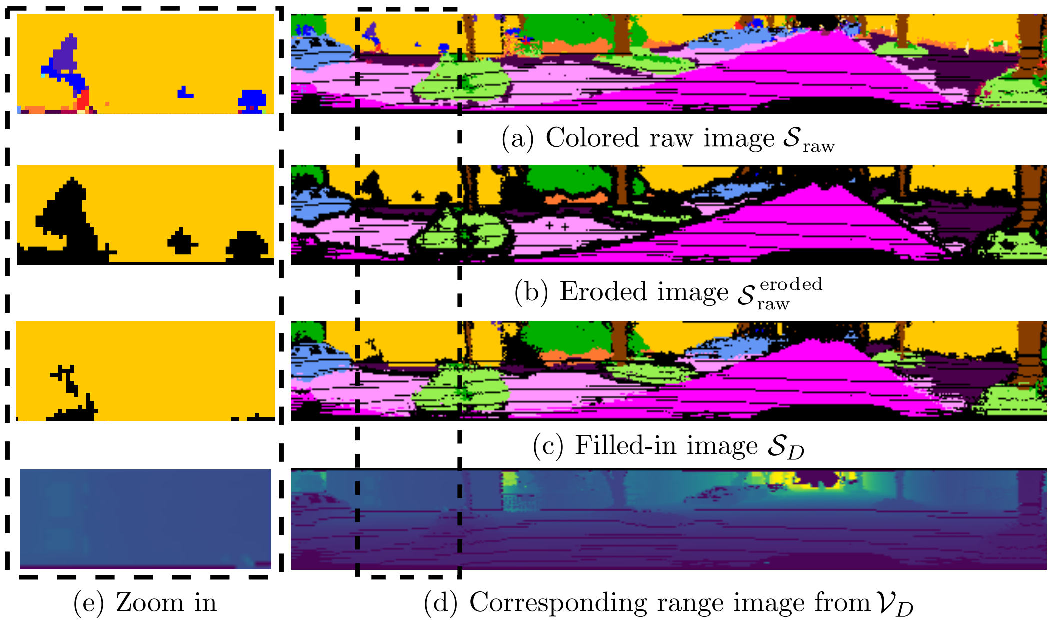

(a) Colored raw image $\mathcal{S}_{\text{raw}}$
(b) Eroded image $\mathcal{S}_{\text{raw}}^{\text{eroded}}$
(c) Filled-in image $\mathcal{S}_D$
(e) Zoom in
(d) Corresponding range image from $\mathcal{V}_D$

Figure 3.3: Visualization for each step of the proposed flood-fill algorithm. Given the (a) raw semantic image $\mathcal{S}_{\text{raw}}$, we first use an erosion to remove boundary labels and small areas of wrong labels resulting in (b) the eroded image $\mathcal{S}_{\text{raw}}^{eroded}$. (c) We then finally fill-in eroded labels with neighboring labels to get a more consistent result $\mathcal{S}_{\mathcal{D}}$. Black points represent empty pixels with label 0. (d) Shows the depth and (e) the details inside the areas with the dashed borders.

The input to the flood-fill is the raw semantic image $\mathcal{S}_{\text{raw}}$ generated by the RangeNet++ and the corresponding vertex image $\mathcal{V}_D$. The value of each pixel in the image $\mathcal{S}_{\text{raw}}$ is a semantic label. The corresponding pixel in the vertex image contains the 3D coordinates of the nearest 3D point in the LiDAR coordinate system. The output of the approach is the refined semantic image $\mathcal{S}_D$.

Considering that the prediction uncertainty of object boundaries is higher than that of the center of an object [102], 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 image $\mathcal{S}_{\text{raw}}^{\text{eroded}}$. Combining this image with the depth information generated from the vertex image $\mathcal{V}_D$, we then fill-in the eroded image. 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 ϵ_{rej}.

Figure 3.3 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 Figure 3.3(e).

3.1.4 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, 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:

$$\begin{aligned} l_s^{(t)} = l_s^{(t-1)} & \\ & + \text{odds}\left(p_{\text{stable}} \exp\left(-\frac{\alpha^2}{\sigma_\alpha^2}\right) \exp\left(-\frac{d^2}{\sigma_d^2}\right)\right) \\ & - \text{odds}(p_{\text{prior}}) - \text{odds}(p_{\text{penalty}}), \end{aligned} \tag{3.1}$$

where $\text{odds}(p) = \log(p\,(1-p)^{-1})$, 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 $\boldsymbol{n}_s$ 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 as introduced in Section 2.3.

Instead of using semantic information, Pomerleau et al. [138] propose 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 Figure 3.4, we illustrate the effect of our filtering method compared to naively removing all surfels from classes corresponding to movable objects. When

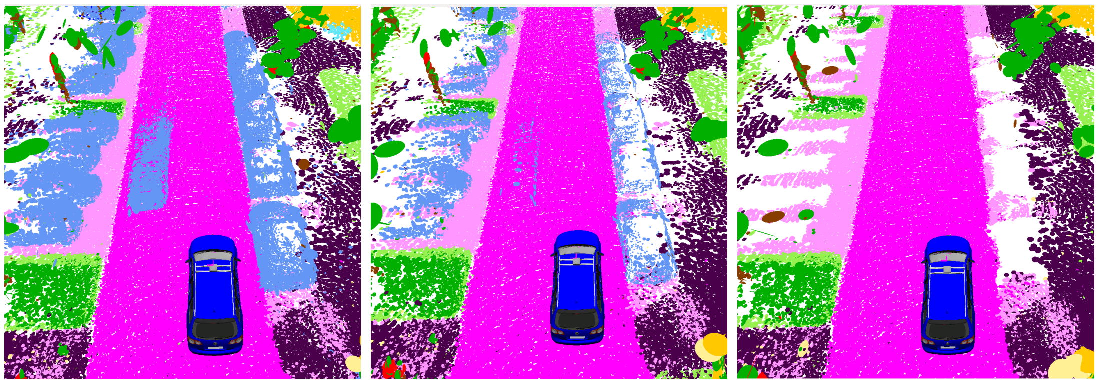

(a) SuMa (b) SuMa++ (c) SuMa_nomovable

Figure 3.4: 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.

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.

3.1.5 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:

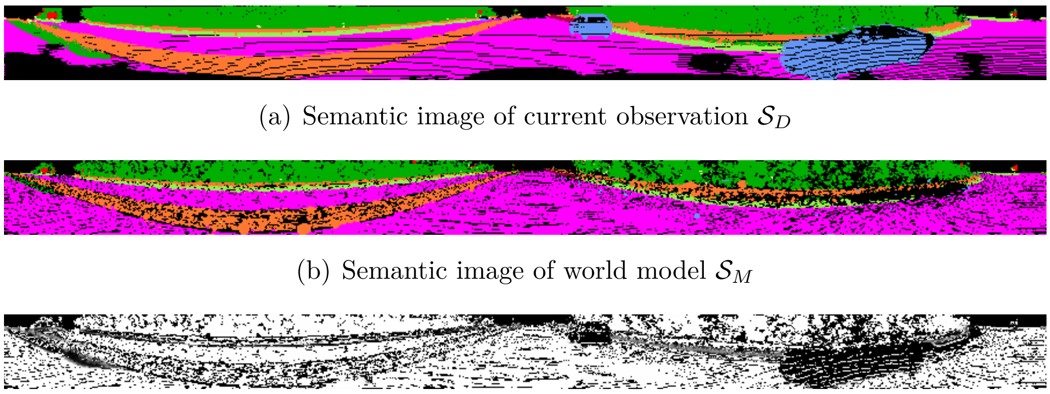
$$E(\mathcal{V}_D, \mathcal{V}_M, \mathcal{N}_M) = \sum_{u \in \mathcal{V}_D} w_u \underbrace{\left(\boldsymbol{n}_u^\top \left(T^{(k)}_{C_{t-1}C_t} \boldsymbol{u} - \boldsymbol{v}_u \right) \right)^2}_{r_u}, \tag{3.2}$$

where r_u and w_u are the corresponding residual and weight, respectively. Each vertex $\boldsymbol{u} \in \mathcal{V}_D$ is projectively associated to a reference vertex $\boldsymbol{v}_u \in \mathcal{V}_M$ and its normal $\boldsymbol{n}_u \in \mathcal{N}_M$ via

$$\boldsymbol{v}_u = \mathcal{V}_M \left(\Pi \left(T^{(k)}_{C_{t-1}C_t} \boldsymbol{u} \right) \right), \tag{3.3}$$

$$\boldsymbol{n}_u = \mathcal{N}_M \left(\Pi \left(T^{(k)}_{C_{t-1}C_t} \boldsymbol{u} \right) \right). \tag{3.4}$$

For the minimization, we use Gauss-Newton and determine increments δ by



(a) Semantic image of current observation $\mathcal{S}_D$

(b) Semantic image of world model $\mathcal{S}_M$

(c) Visualized weights image

Figure 3.5: Visualization of semantic ICP. (a) semantic image $\mathcal{S}_D$ for the current laser scan, (b) corresponding semantic image $\mathcal{S}_M$ rendered from the model, (c) the weights image during the ICP. The darker the pixel, the lower the weight of the corresponding pixel.

iteratively computing:

$$\delta = \left(\mathsf{J}^\top \mathsf{W} \mathsf{J}\right)^{-1} \mathsf{J}^\top \mathsf{W} \boldsymbol{r}, \tag{3.5}$$

where $\mathsf{W} \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing weights w_u for each residual r_u, $\boldsymbol{r} \in \mathbb{R}^n$ is the stacked residual vector, and $\mathsf{J} \in \mathbb{R}^{n\times 6}$ the Jacobian of $\boldsymbol{r}$ 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 $w_u^{(k)}$ for the residual $r_u^{(k)}$ in iteration k as follows:

$$w_u^{(k)} = \rho_{\text{Huber}}\left(r_u^{(k)}\right)\ C_{\text{semantic}}(\mathcal{S}_D(\boldsymbol{u}), \mathcal{S}_M(\boldsymbol{u}))\ \mathbb{I}\left\{l_s^{(k)} \geq l_{stable}\right\}, \tag{3.6}$$

where $\rho_{\text{Huber}}(r)$ corresponds to the Huber norm, given by:

$$\rho_{\text{Huber}}(r) = \begin{cases} 1 & \text{, if } |r| < \delta \\ \delta|r|^{-1} & \text{, otherwise.} \end{cases} . \tag{3.7}$$

For each vertex $\boldsymbol{u}$, the corresponding semantic image stores both the semantic class y_u and the estimated probability of that label $P(y_u \mid \boldsymbol{u})$, leading to $\mathcal{S}_D(\boldsymbol{u}) = (y_u, P(y_u \mid \boldsymbol{u}))$. The same holds for the rendered semantic image of the map. The semantic compatibility $C_{\text{semantic}}\big((y_u, P(y_u \mid \boldsymbol{u})), (y_{v_u}, P(y_{v_u} \mid \boldsymbol{v}_u))\big)$ is defined as:

$$C_{\text{semantic}}(\cdot,\cdot) = \begin{cases} P(y_u \mid \boldsymbol{u}) & \text{, if } y_u = y_{v_u} \\ 1 - P(y_u \mid \boldsymbol{u}) & \text{, otherwise.} \end{cases} , \tag{3.8}$$

which is using the probability 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.

Figure 3.5 shows the weighting for a highway scene with two moving cars visible in the scan, see Figure 3.5(a). Note that our filtering of dynamics using the semantics, as described in Section 3.1.4, removed the moving cars from the map, see Figure 3.5(b). Therefore, we can also see a low weight corresponding to lower intensity in Figure 3.5(c), since the classes of the observation and the map disagree.

3.2 Experimental Evaluation

Our experimental evaluation supports 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 [67], 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 with 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 with an RTK-GPS.

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 [13] 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).

Table 3.1: Odometry results on KITTI Road dataset

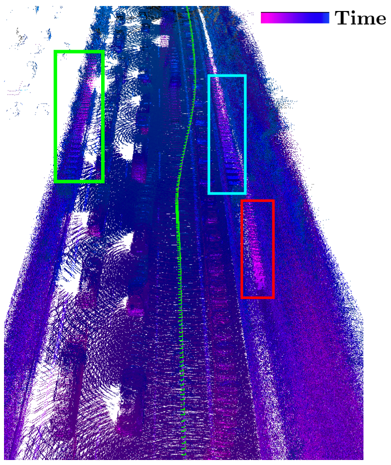
Sequence	Environment	Approach		
		SuMa	SuMa_nomovable	SuMa++
30	country	0.38/0.96	0.39/0.97	**0.38**/**0.90**
31	country	1.54/2.02	1.66/2.13	**1.19**/**2.02**
32	country	1.38/1.70	1.63/1.76	**1.00**/**1.57**
33	highway	**1.61**/**1.79**	1.72/1.80	1.67/1.87
34	highway	0.79/1.17	0.70/1.14	**0.60**/**1.09**
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 deg 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.

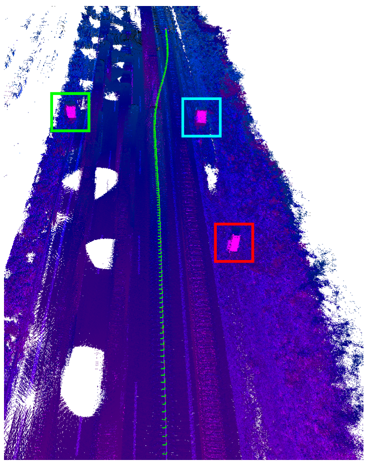
3.2.1 KITTI Road Sequences

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 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 inconsistencies in the generated map.

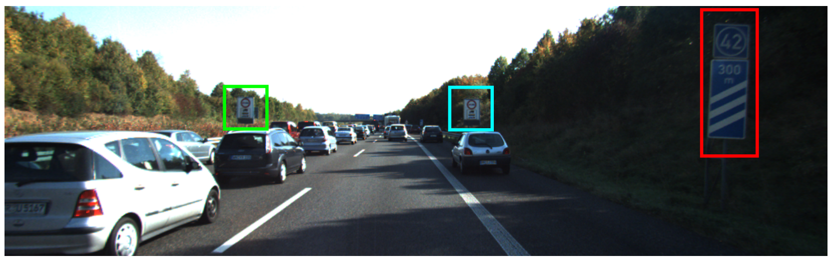
Figure 3.6 shows an example generated with SuMa and the proposed SuMa++. In the case of the purely geometric approach, we clearly see that the pose cannot be correctly estimated, since the highlighted traffic signs show up at different



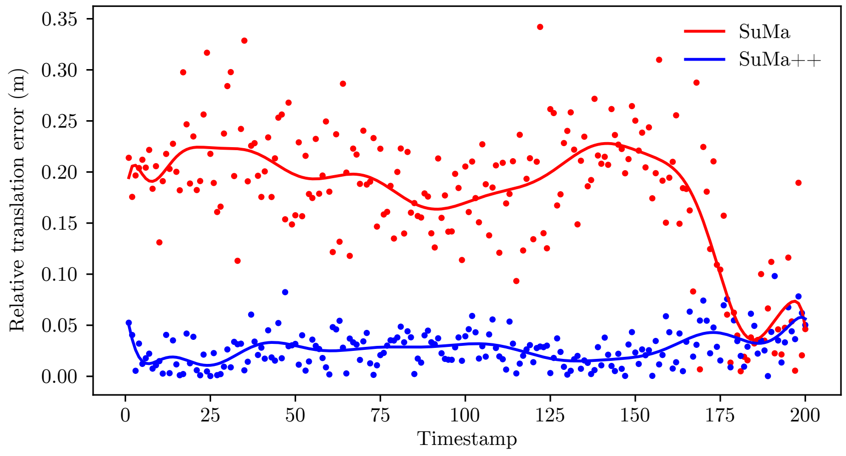
(a) SuMa



(b) SuMa++

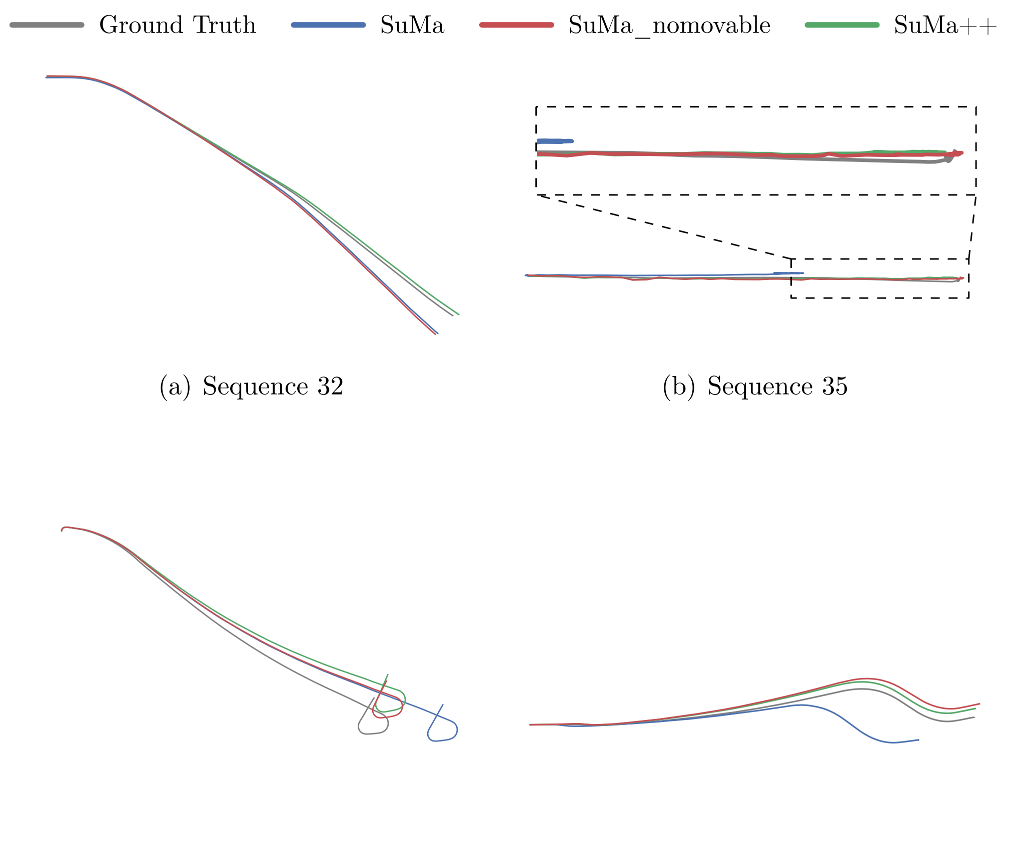


(c) Corresponding front-view camera image



(d) Relative translation error plot for each time step

Figure 3.6: 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 at each timestamp, and the curves are polynomial fitting results of those dots for better visualization and comparison.



(a) Sequence 32

(b) Sequence 35

(c) Sequence 40

(d) Sequence 41

Figure 3.7: Trajectories of different methods test on KITTI road dataset.

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 environment with many outliers caused by moving objects.

Table 3.1 shows the relative translational and relative rotational error and Figure 3.7 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 mov-

Table 3.2: Odometry results on KITTI odometry benchmark (training set)

Sequence	Environment	Approach		
		SuMa	SuMa_nomovable	SuMa++
00*	urban	0.23/0.68	22.0/58.0	**0.22**/**0.64**
01	highway	0.54/1.70	0.57/1.70	**0.46**/**1.60**
02*	urban	0.48/1.20	25.0/63.0	**0.37**/**1.00**
03	country	0.50/0.74	**0.45**/**0.67**	0.46/**0.67**
04	country	0.27/0.44	**0.26**/**0.37**	**0.26**/**0.37**
05*	country	**0.20**/0.43	14.0/36.0	**0.20**/**0.40**
06*	urban	0.30/0.54	0.22/0.47	**0.21**/**0.46**
07*	urban	0.54/0.74	0.21/**0.34**	**0.19**/**0.34**
08*	urban	0.38/1.20	13.0/32.0	**0.35**/**1.10**
09*	urban	**0.22**/0.62	13.0/45.0	0.23/**0.47**
10	country	0.32/0.72	12.0/19.0	**0.28**/**0.66**
Average		0.36/0.83	9.24/23.3	**0.29**/**0.70**

Relative errors averaged over trajectories of 100 to 800 m length: relative rotational error in deg 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.

ing objects can also have negative effects on the pose estimation performance in urban environments.

3.2.2 KITTI Odometry Benchmark

The second experiment is designed to show that our method performs superior compared to simply removing certain semantic classes from the observations. This evaluation is performed on the KITTI Odometry Benchmark[1].

Table 3.2 shows the relative translational and relative rotational errors. In most sequences, we can see a similar performance of SuMa++ outperforming its geometric and simply removing certain semantic classes counterparts. 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

[1]KITTI Odometry: `http://www.cvlibs.net/datasets/kitti/eval_odometry.php`.

surfels in the map that are actually static. Second, the removal of parked cars is problematic 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.

3.3 Conclusion

In this chapter, we presented a novel approach for building semantic maps for autonomous driving enabled by a LiDAR-based semantic segmentation of the point cloud which does not require any other types of sensor data. We exploit this information to improve pose estimation accuracy in otherwise ambiguous and challenging situations. In particular, our method exploits semantic consistency between scans and the map to filter out dynamic objects and provide high-level constraints during the ICP process. This allows us to successfully combine semantic and geometric information based solely on 3D LiDAR range scans to achieve considerably better pose estimation accuracy than the purely geometric approach. We evaluated our approach first on the KITTI road dataset showing that our approach is able to generate consistent maps even in situations with many moving objects. We also provide the evaluation on the KITTI Vision Benchmark dataset showing the advantages of our approach in comparison to purely geometric approaches and methods that simply remove certain semantic classes from the observations.

Chapter 4

Deep Learning-Based LiDAR Loop Closing and Localization

In the previous chapter, we introduced a semantic-based LiDAR SLAM system to build semantic maps for autonomous driving in dynamic environments. Despite encouraging results achieved by our method, there are several avenues for further exploiting semantic information for SLAM and localization. For example, in this chapter, we design a novel neural network that uses multiple information generated from LiDAR as input, including semantics, to better find the loop closures for SLAM and also localize the vehicle in a given map.

Nowadays, LiDAR sensors are key components of the sensor suite of autonomous vehicles that allows them to perceive and navigate the world. Especially mapping and localization systems can leverage the range information provided by LiDAR sensors covering the 360 deg surroundings of the vehicle. As discussed in the previous chapter, accurate odometry estimation enhanced with semantic information allows us to build locally consistent maps. However, the estimated poses evidently drift over time due to the incremental estimation process and sensor noises. 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 chapter, we propose 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 an autonomous vehicle. It can be used to tackle loop closing and global localization tasks for autonomous driving as shown in Fig-

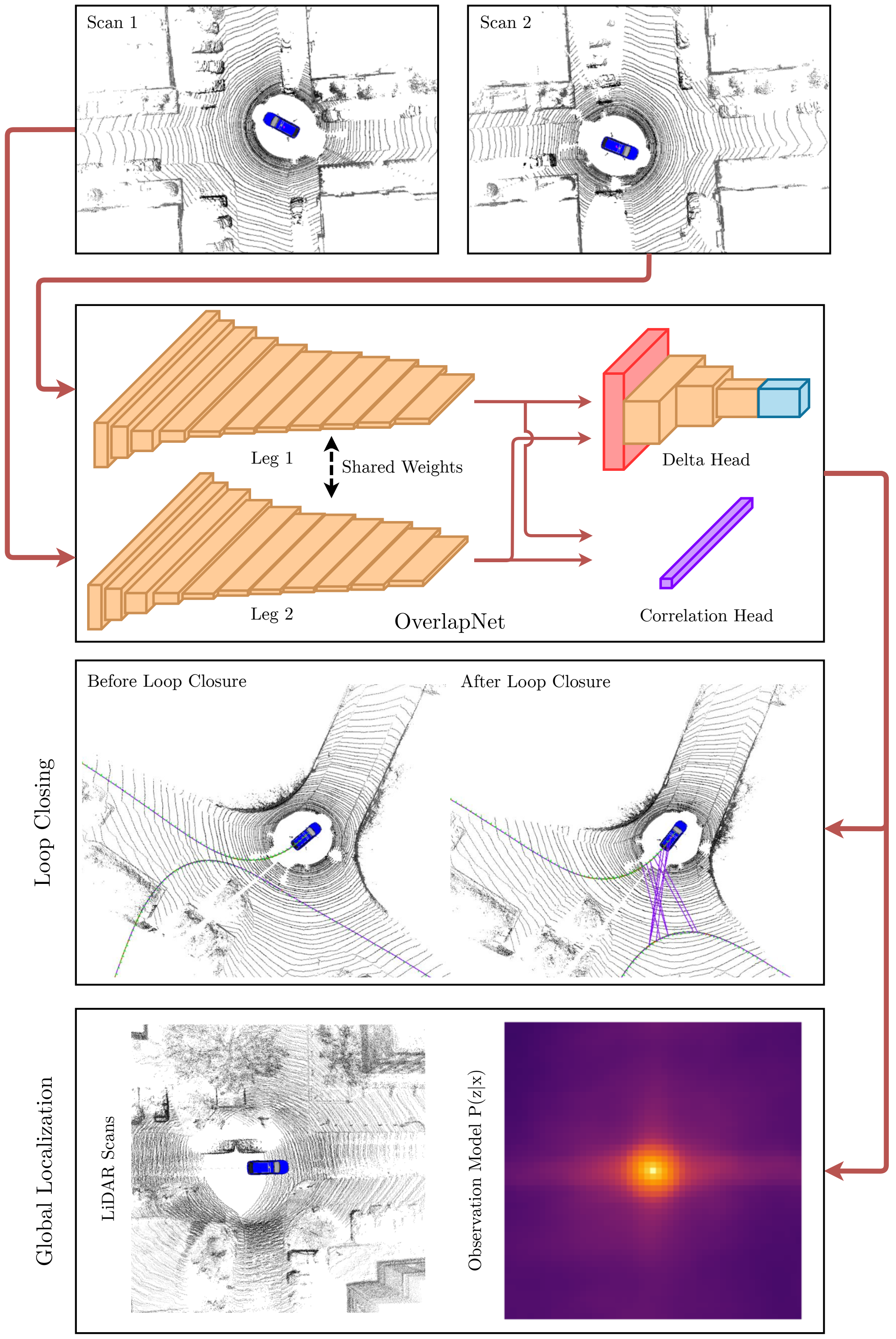


Figure 4.1: Overview: We use a siamese network with two LiDAR scans as input. The result of OverlapNet consisting of the overlap percentage and yaw angle offset can be used for loop closing and/or global localization.

ure 4.1. 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 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 algorithm [19] 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 chapter is a deep neural network, OverlapNet, 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 by a semantic segmentation network 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 lightweight. We furthermore test our method in different applications for autonomous vehicles. We integrate our OverlapNet into our SLAM system 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 approach for updating the importance weights of the particles.

We thoroughly evaluate our method for different LiDAR perception tasks for autonomous vehicles. To test LiDAR-based loop closing, we train the proposed OverlapNet on parts of the KITTI odometry dataset [67] 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 [130], 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 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.

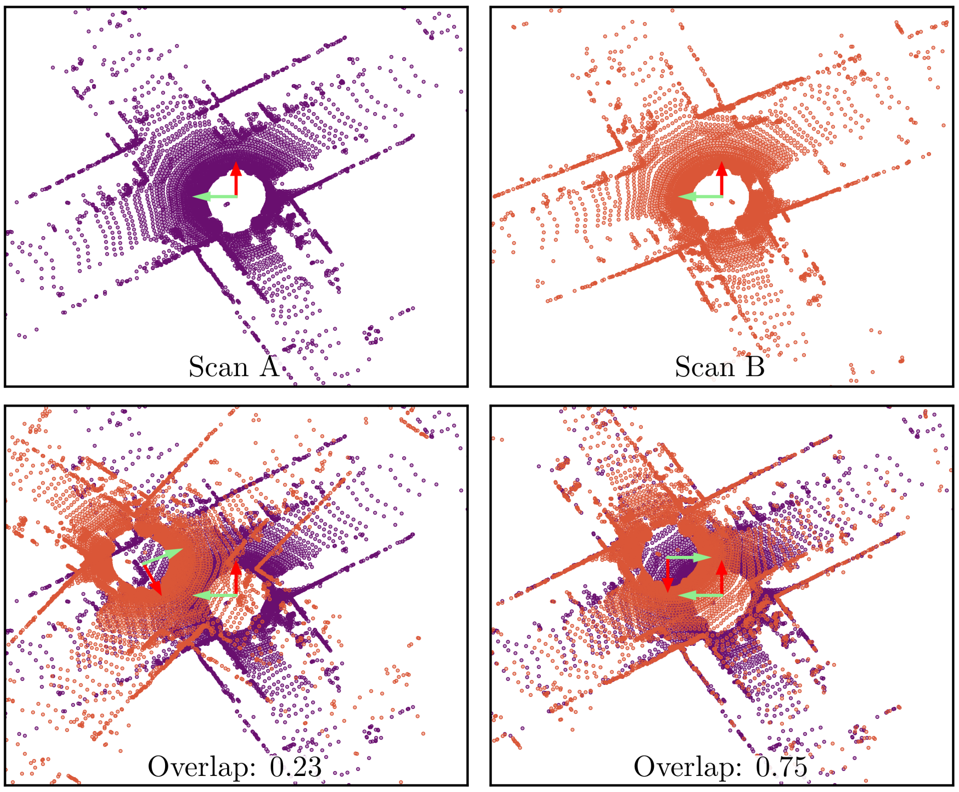


Figure 4.2: Overlap of two scans (scan A and B) at an intersection but computed with different relative transformations. The overlap depends on the relative transformation and larger overlap values often correspond to better alignment between the point clouds. Our approach can predict the overlap *without* knowing the relative transformation between the scans.

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.

4.1 OverlapNet

The idea of overlap that we are using here has its originates from the photogrammetry and computer vision community [79]. 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 chapter, we use the idea of overlap for range images generated from LiDAR scans. As illustrated in Figure 4.2, when the vehicle drives back to an intersection from the opposite direction, different relative transformations between two scans lead to different overlap values. Larger overlap values often correspond to better alignment between the point clouds. We propose a novel neural network called OverlapNet that can predict the overlap *without* knowing the relative transformation between the scans, which can quantify the quality of matches and is a useful tool for LiDAR-based loop closure detection and localization.

4.1.1 Definition of the Overlap Between LiDAR Scans

We use spherical projections of LiDAR scans as input data as presented in Section 2.1, which is often used to speed up computations [15, 20, 35]. We project the point cloud $\mathcal{P} = \{\boldsymbol{p}_i\}, i \in \{1, \dots, N\}$ to a so-called vertex image $\mathcal{V}$, where each pixel stores to the nearest 3D point. Each point $\boldsymbol{p}_i = (x, y, z)^\top$ is converted via the function $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates and finally to image coordinates $(u, v)^\top$ by Equation (2.2) resulting in the vertex image $\mathcal{V}$.

For the LiDAR scans $\mathcal{P}_1$ and $\mathcal{P}_2$, we generate the corresponding vertex images $\mathcal{V}_1$ and $\mathcal{V}_2$. 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 $\mathsf{T}_{WC_t} \in \mathbb{R}^{4\times4}$. Given the poses T_{WC_1} and T_{WC_2}, we can reproject scan $\mathcal{P}_1$ into the other's vertex image $\mathcal{V}_2$ and generate a reprojected vertex image $\mathcal{V}_1'$:

$$\mathcal{V}_1' = \Pi\left(\mathsf{T}_{WC_1}^{-1}\mathsf{T}_{WC_2}\mathcal{P}_1\right). \tag{4.1}$$

We calculate the absolute difference of all corresponding vertices 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 $\epsilon_{\text{overlap}}$ relative to all valid entries, i.e., the overlap $O_{C_1C_2}$ of two LiDAR scans C_1, C_2 is defined as:

$$O_{C_1C_2} = \frac{\sum_{(u,v)} \mathbb{I}\Big\{\|\mathcal{V}_1'(u,v) - \mathcal{V}_2(u,v)\|_2 \leq \epsilon_{\text{overlap}}\Big\}}{\min\left(\text{valid}(\mathcal{V}_1'), \text{valid}(\mathcal{V}_2)\right)}, \tag{4.2}$$

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 pixels might have a valid LiDAR measurement associated after the projection.

We use Equation (4.2) only for creating training data, i.e., only positive examples of correct loop closures get a non-zero overlap assigned using the relative

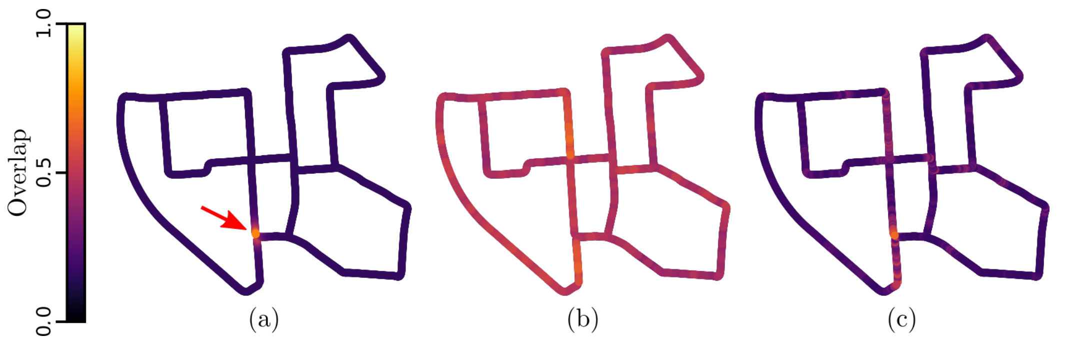


Figure 4.3: Overlap estimations of one scan to all others. (a) The red arrow points out the position of the query scan. (b) We directly use Eq. (4.2) to estimate the overlap between two LiDAR scans without knowing the accurate relative poses. Therefore, we apply different orientations, e.g., every 30 deg rotation around the vertical axis, and use the maximum as the final estimate. It is hard to decide which pairs of scans are true loop closures, since most estimations of Eq. (4.2) 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.

poses between scans, as shown in Figure 4.3. Thus, the proposed approach can learn not only high overlap values for nearby scans, but also is able 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 Equation (4.2) 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 Equation (4.2) assuming the relative pose to be the identity and applying different orientations, e.g., every 30 deg rotation around the vertical axis, and using the maximum over all these overlaps as an estimate. Figure 4.3 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 Equation (4.2), many scans which are far away getting 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.

4.1.2 Overlap Network Architecture

A visual overview of our proposed OverlapNet is depicted in Figure 4.4. We exploit multiple cues, which can be generated from a single LiDAR scan, including

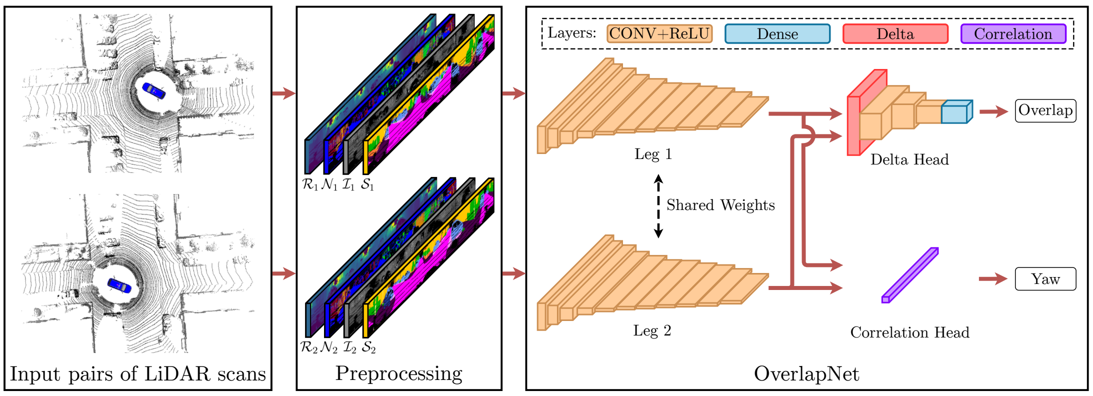

Input pairs of LiDAR scans
$\mathcal{R}_1\ \mathcal{N}_1\ \mathcal{I}_1\ \mathcal{S}_1$
$\mathcal{R}_2\ \mathcal{N}_2\ \mathcal{I}_2\ \mathcal{S}_2$
Preprocessing
Layers: CONV+ReLU Dense Delta Correlation
Leg 1
Shared Weights
Leg 2
Delta Head
Overlap
Correlation Head
Yaw
OverlapNet

Figure 4.4: 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.

depth, normal, intensity, and semantic class probability information. The depth information is stored in the range image $\mathcal{R}$, which defines one input channel. We use neighborhood information of the vertex image to generate a normal image $\mathcal{N}$, 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 image $\mathcal{I}$. The point-wise semantic class probabilities are computed using RangeNet++ [120] and we represent them as a semantic image $\mathcal{S}$. 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 $64 \times 900 \times 8$, where 64 and 900 are the height and width of the range image.

Our proposed OverlapNet is a siamese network architecture, which consists of two legs sharing weights and two heads that use the same pair of feature volumes generated by the two legs [23]. The trainable layers are listed in Table 4.1.

4.1.2.1 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 the range projection we used is cyclic, which means a change in the

Table 4.1: 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

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 range images. 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 deg in the case of our leg architecture.

4.1.2.2 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 Figure 4.5, computes all possible absolute differences of all pixels. It takes the output feature volumes $\mathcal{L}^l \in \mathbb{R}^{H\times W\times C}$ from the two legs $l \in \{0,1\}$ as input. These are stacked in a tiled tensor $\mathcal{T}^l \in \mathbb{R}^{HW\times HW\times C}$ as follows:

$$\mathcal{T}^0(iW+j,k,c) = \mathcal{L}^0(i,j,c)\,, \tag{4.3}$$

$$\mathcal{T}^1(k,iW+j,c) = \mathcal{L}^1(i,j,c)\,, \tag{4.4}$$

with $k = \{0,\ldots,HW-1\}$, $i = \{0,\ldots,H-1\}$ and $j = \{0,\ldots,W-1\}$.

Note that $\mathcal{T}^1$ is transposed with respect to $\mathcal{T}^0$, as depicted in the middle of Figure 4.5. After that, all differences are calculated by element-wise absolute

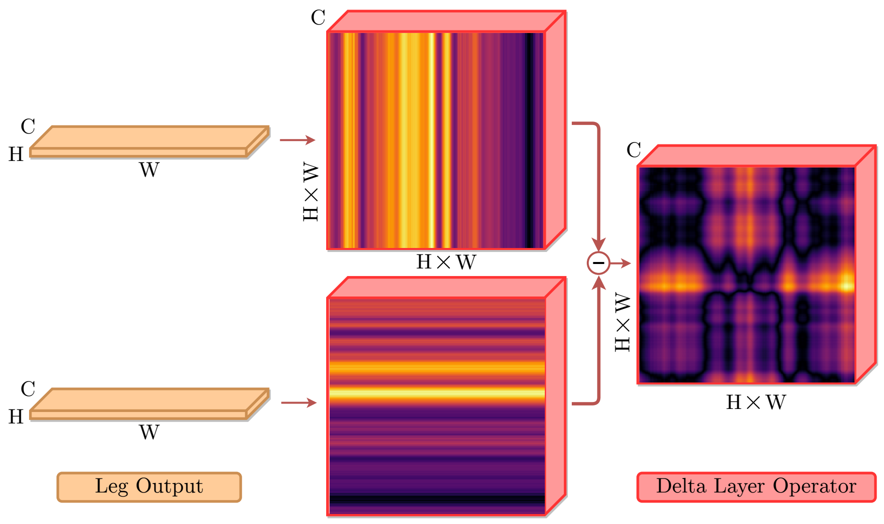


Figure 4.5: The delta layer: Computation of pairwise differences is efficiently performed by concatenating the feature volumes and transposition of one concatenated feature volume.

differences between $\mathcal{T}^0$ and $\mathcal{T}^1$.

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.

4.1.2.3 Correlation Head

The correlation head [126] 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 deg resolution.

4.1.3 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, each training sample consists of $(I_1,\ I_2,\ Y_O,\ Y_Y)$, 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 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 δ:

$$L\left(I_1, I_2, Y_O, Y_Y\right) = L_O\left(I_1, I_2, Y_O\right) + \delta L_Y\left(I_1, I_2, Y_Y\right). \tag{4.5}$$

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:

$$L_O\left(I_1, I_2, Y_O\right) = \operatorname{sigmoid}\left(s\left(\left|\hat{Y}_O - Y_O\right| + a\right) - b\right), \tag{4.6}$$

with $\operatorname{sigmoid}(v) = (1 + \exp(-v))^{-1}$, 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

$$L_Y\left(I_1, I_2, Y_Y\right) = \sum\nolimits_{i=\{1,\ldots,N\}} \mathrm{H}\left(Y_Y^i, \hat{Y}_Y^i\right), \tag{4.7}$$

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. The term $\hat{Y}_Y = f_C\left(f_L\left(I_1\right), f_L\left(I_2\right)\right)$ 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 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 Section 4.4.3.3. More uncertain estimations also decrease the accuracy of pose estimation building on top of OverlapNet and the corresponding experimental results can be found in Section 4.4.4.2.

4.2 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 may be 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 it is a direct measure for the number of corresponding points, which can be exploited by ICP employed by most SLAM systems. 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.

4.2.1 SLAM Pipeline

In line with Chapter 3, we use the surfel-based mapping system SuMa [15] as our SLAM pipeline and integrate OverlapNet in SuMa replacing its original heuristic loop closure detection method. As already introduced in Section 2.3, we only summarize here the key steps of SuMa relevant to our approach and refer for more details to Section 2.3.

Our OverlapNet uses the same vertex image $\mathcal{V}_D$ and normal image $\mathcal{N}_D$ as used in SuMa. 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 $\mathsf{T}_{C_{t-1}C_t}$ and consequently $\mathsf{T}_{WC_t}$by chaining all pose increments. Therefore, each vertex $\boldsymbol{u} \in \mathcal{V}_D$ is projectively associated to a reference vertex $\boldsymbol{v}_u \in \mathcal{V}_M$. Given this association information, SuMa estimates the transformation between scans by incrementally minimizing the point-to-plane error, see Section 2.3 for details.

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 Section 4.4.4.

4.2.2 Covariance Propagation for Geometric Verification

SuMa's loop closure detection uses a fixed search radius. In contrast, we suggest using the covariance of the pose estimate and error propagation to automatically adjust the search radius as detailed below.

We assume a noisy pose $\mathsf{T}_{C_{t-1}C_t}$ with mean $\bar{\mathsf{T}}_{C_{t-1}C_t}$ and covariance $\Sigma_{C_{t-1}C_t}$. We estimate the covariance matrix following Huber [78] by

$$\Sigma_{C_{t-1}C_t} = \frac{1}{K}\frac{E}{N-M}\left(\mathsf{J}^\top \mathsf{W} \mathsf{J}\right)^{-1}, \tag{4.8}$$

where K is the correction factor of the Huber robustized covariance estimation, E is the sum of the squared weighted point-to-plane errors (the sum of squared weighted residuals) given the pose $\mathsf{T}_{C_{t-1}C_t}$, see Equation (2.8), N is the number of correspondences, $M = 6$ is the dimension of the transformation between two 3D poses. $\mathsf{W} \in \mathbb{R}^{N\times N}$ is a diagonal matrix containing weights w_u for each residual r_u, and $\mathsf{J} \in \mathbb{R}^{N\times 6}$ the Jacobian of $\boldsymbol{r}$ with respect to the pose increment.

To estimate the propagated uncertainty during the incrementally pose estimation, we can update the mean and covariance as follows:

$$\bar{\mathsf{T}}_{WC_t} = \bar{\mathsf{T}}_{WC_{t-1}} \bar{\mathsf{T}}_{C_{t-1}C_t} \,, \tag{4.9}$$

$$\Sigma_{WC_t} \approx \Sigma_{WC_{t-1}} + \mathsf{J}_{C_{t-1}C_t} \Sigma_{C_{t-1}C_t} \mathsf{J}^{\top}_{C_{t-1}C_t} \,, \tag{4.10}$$

where $\mathsf{J}_{C_{t-1}C_t}$ is the Jacobian of Equation (4.9).

Since we use the Mahalanobis distance D_M as a probabilistic distance measure between two poses, we make use of Lie algebra to express T as a 6D vector $\boldsymbol{\xi} \in \mathrm{se}(3)$ using $\boldsymbol{\xi} = \log \mathsf{T}$, yielding

$$D_M\left(\mathsf{T}_{C1}, \mathsf{T}_{C2}\right) = \sqrt{\Delta\boldsymbol{\xi}^{\top}_{C1C2} \Sigma^{-1}_{C1C2} \Delta\boldsymbol{\xi}_{C1C2}} \,. \tag{4.11}$$

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 $\mathsf{T}_{WC_{j^*}}$ using the frame-to-model ICP. For the ICP initialization, we use the yaw angle offset estimated from OverlapNet while keeping other parameters 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 $\{t+1, \ldots, t+\Delta_{\text{verification}}\}$, which ensures that we only add consistent loop closures to the pose graph.

4.3 OverlapNet for Global Localization

As introduced in Section 2.2, we use the Monte-Carlo localization or MCL framework to achieve global localization. For global localization in an MCL framework [44], 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.

4.3.1 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 F using the input tensor 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 F during localization on the map. The feature 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.

4.3.2 Monte-Carlo Localization

As presented in Section 2.2, MCL is one of the classic frameworks for localization. In our setup, each particle represents a hypothesis for the robot's or autonomous vehicle's 2D pose $\boldsymbol{x}_t = (x, y, \theta)_t^\top$ 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 $\boldsymbol{u}_t$. The expected observation from the predicted pose of each particle is then compared to the actual observation $\boldsymbol{z}_t$ acquired by the robot to update the particle's weight based on the observation model.

MCL realizes a recursive Bayesian filtering scheme of the form:

$$p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) = \eta \; p(\boldsymbol{z}_t \mid \boldsymbol{x}_t) \int p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}) \; p(\boldsymbol{x}_{t-1} \mid \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t-1}) \; d\boldsymbol{x}_{t-1}, \tag{4.12}$$

where η is the normalization constant resulting from Bayes rule, $p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ is the motion model, and $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$ is the observation model. We split the observation model into two parts:

$$p(\boldsymbol{z}_t \mid \boldsymbol{x}_t) = p_L\left(\boldsymbol{z}_t \mid \boldsymbol{x}_t\right) \; p_O\left(\boldsymbol{z}_t \mid \boldsymbol{x}_t\right) , \tag{4.13}$$

where $\boldsymbol{z}_t$ is the observation at time t, $p_L\left(\boldsymbol{z}_t \mid \boldsymbol{x}_t\right)$ is the probability encoding the location $(x, y)^\top$ agreement between the current query LiDAR scan and the virtual

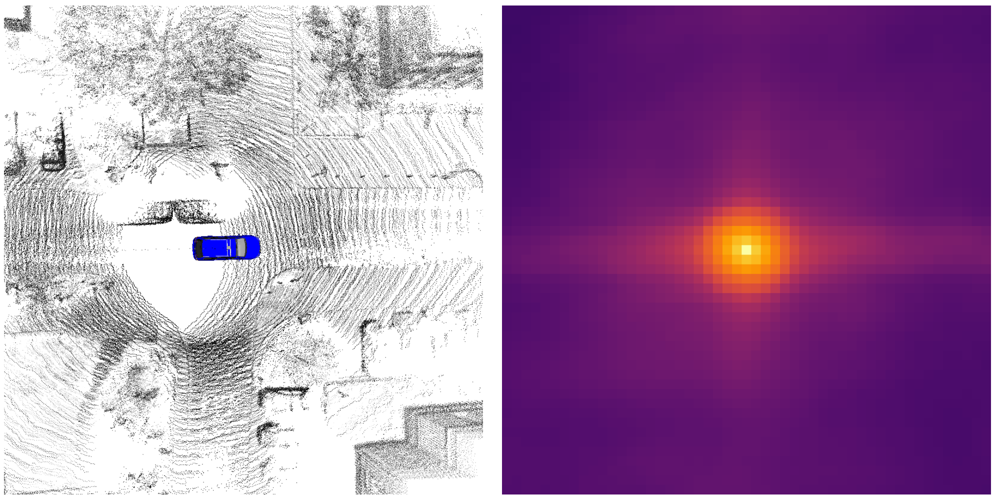


Figure 4.6: Overlap-based observation model describing $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$ for MCL. Local heatmap of the scan at the car's position with respect to the map. Lighter shades correspond to higher probabilities.

scan at the nearest grid position and $p_O\left(\boldsymbol{z}_t \mid \boldsymbol{x}_t\right)$ is the probability encoding the yaw angle θ agreement between the same pairs of scans.

4.3.3 OverlapNet-Based Observation Model

Given a particle i with the state estimate $(x^{(i)}, y^{(i)}, \theta^{(i)})^\top$, the overlap estimates encode the location agreement between the query LiDAR scan and virtual scans of the grid cells where particles are located. It can be directly used as the probability:

$$p_L\left(\boldsymbol{z}_t \mid \boldsymbol{x}_t\right) \propto f\left(\boldsymbol{z}_t, \boldsymbol{z}^{(i)}; \boldsymbol{w}\right) \ , \tag{4.14}$$

where f corresponds to the neural network providing the overlap estimate between the input scans $\boldsymbol{z}_t$, $\boldsymbol{z}^{(i)}$ and $\boldsymbol{w}$ is the pre-trained weights of the network. $\boldsymbol{z}_t$ and $\boldsymbol{z}^{(i)}$ are the current query scan and a virtual scan at one $(x, y)^\top$ location respectively.

For illustration purposes, Figure 4.6 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 $(x^{(i)}, y^{(i)}, \theta^{(i)})^\top$, 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:

$$p_O\left(\boldsymbol{z}_t \mid \boldsymbol{x}_t\right) \propto \exp\left(-\frac{1}{2}\frac{\left(g\left(\boldsymbol{z}_t, \boldsymbol{z}^{(i)}; \boldsymbol{w}\right) - \theta_i\right)^2}{\sigma_\theta^2}\right), \tag{4.15}$$

where g corresponds to the neural network providing the yaw angle estimation between the input scans $\boldsymbol{z}_t, \boldsymbol{z}^{(i)}$ and $\boldsymbol{w}$ is the pre-trained weights of the network. $\boldsymbol{z}_t$ and $\boldsymbol{z}^{(i)}$ are the current query scan and a virtual scan of one particle respectively. σ_θ is the variance for orientation probability.

When generating the virtual scans of the grid map, all virtual scans will be set facing along the absolute 0 deg 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. Equation (4.15) 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 $(x, y, \theta)^\top$.

4.4 Experimental Results

The experimental evaluation is designed to evaluate our approach. They support the claims we made in the introduction of this chapter, which are: 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.

4.4.1 Implementation Details

We provide the parameters used in the proposed method in Table 4.1 and Table 4.2 for the purpose of reproducibility to the experimental results. Table 4.1

Table 4.2: Hyper parameters of our overlap-based MCL

Parameter	Description	Value
h	Height of range image	64
w	Width of range image	900
$d_{\max}$	Maximum range	75 m
$\epsilon_{\text{overlap}}$	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	Variance for orientation probability	10 deg
$\epsilon_{\text{converge}}$	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 δ	5

shows the configuration of the network architecture and Table 4.2 shows hyper parameters used in the proposed method. As shown in Table 4.2, for generating the range image following the SuMa setup [15], 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 Equation (4.2), we use $\epsilon_{\text{overlap}} = 1$ 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, Equation (4.5), we set $\delta = 5$. For the overlap loss, Equation (4.6), we use $a = 0.25, b = 12$, and scale factor $s = 24$. For the calculation of yaw angle probability, Equation (4.15), we set the variance $\sigma_\theta = 10$. We use the a threshold of $\epsilon_{\text{converge}} = 5\,m$ to decide whether the global localization converges or not.

4.4.2 Datasets

For SLAM and loop closing, we train and evaluate our approach on the KITTI odometry benchmark [67] as also used in the previous chapter. It provides 11 sequences (00-10) with ground truth poses covering different types of environments, e.g., urban, country, and highway. We use sequences 03-10 for training, sequence 02 is used for validation and sequence 00 for evaluation. In sequence 00, the vehicle starts to re-enter the visited area in the city after the 170 s. Therefore we use the data collected in the first 170 s to generate the map, i.e., a database for loop closing. The remaining point clouds are used for localization queries. The query point clouds are sampled to be at least 3 m apart, as the same setup introduced in the state-of-the-art baseline method by Schaupp [155].

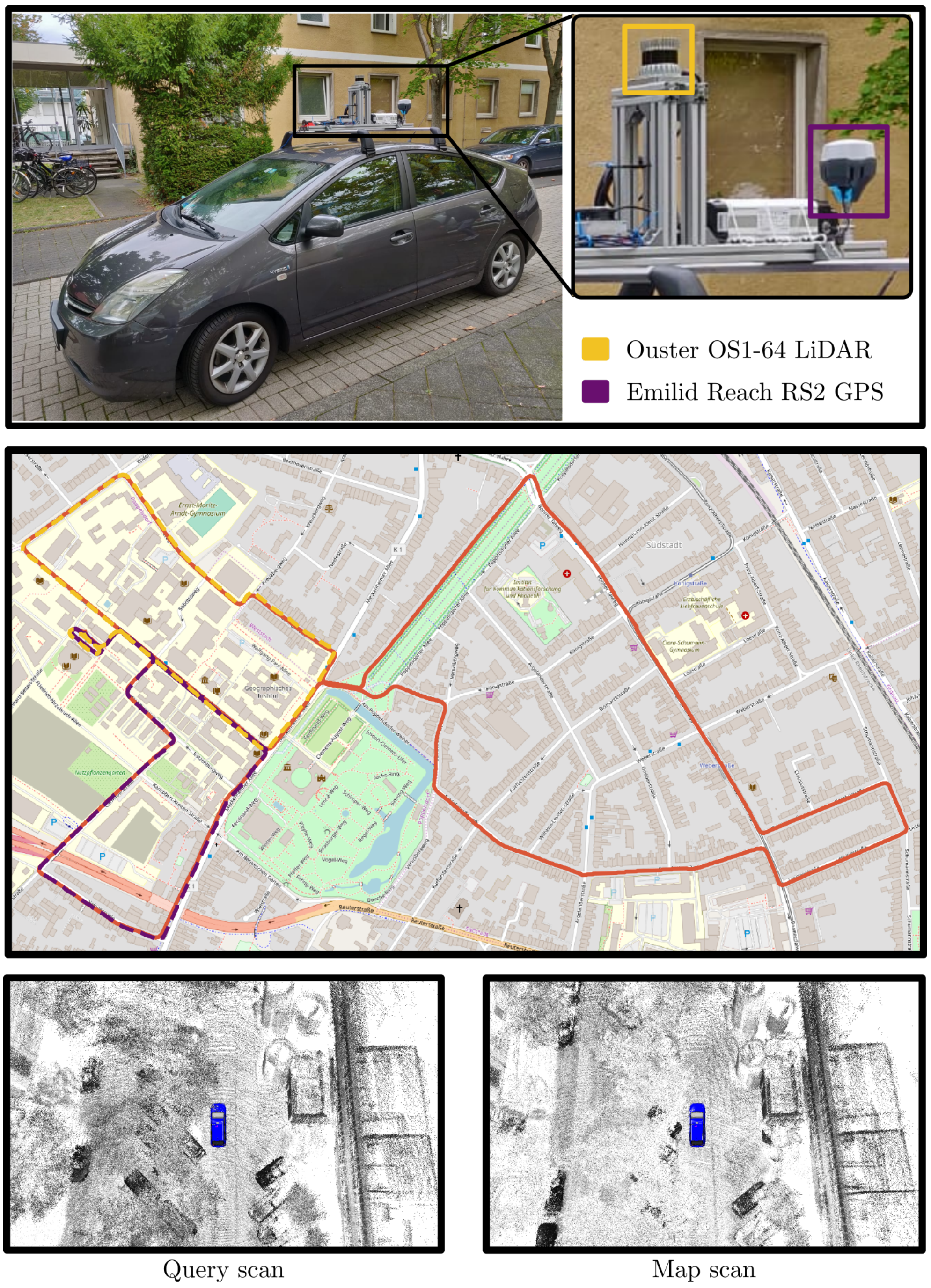


Figure 4.7: Illustration of our IPB-Car dataset. Upper: sensor setup used for data recording. Middle: trajectories of the dataset used in this chapter, overlayed on Open-StreetMap. The orange trajectory represents the sequence used to generate a map for localization. The yellow and purple trajectories 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 in different places.

To evaluate the generalization capabilities of our method, we also test it on the Ford campus dataset [130], 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 evaluating global localization, we use our own IPB-Car dataset, collected with our self-developed sensor platform illustrated in Figure 4.7. 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 [15], combined with a GPS using SAPOS correction 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 sequences 00 and 01 for localization. As can be seen from Figure 4.7, 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.

4.4.3 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 [204], LocNet++ [155] and OREOS [155], and also the traditional hand-crafted feature-based method FPFH [149]. 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

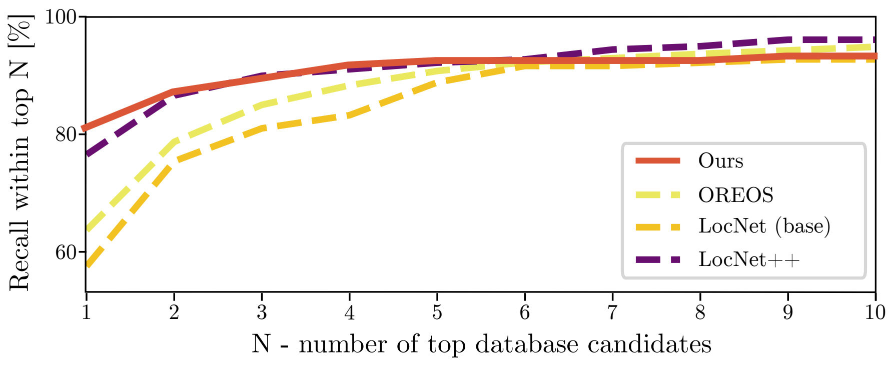


Figure 4.8: Place recognition performance on KITTI sequence 00.

Table 4.3: Yaw estimation errors

Approach	Mean [deg]	std [deg]	Recall [%]
FPFH	13.28	32.19	97
OREOS	12.67	15.23	100
Ours	**1.13**	**3.34**	**100**

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 [155].

4.4.3.1 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 Figure 4.8. 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.

4.4.3.2 Yaw Angle Estimation

Table 4.3 summarizes the yaw angle errors on KITTI sequence 00. We can see that our method outperforms the other methods in terms of mean errors 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 FPFH-based method that sometimes fails.

The superior performance can be mainly attributed to the correlation head

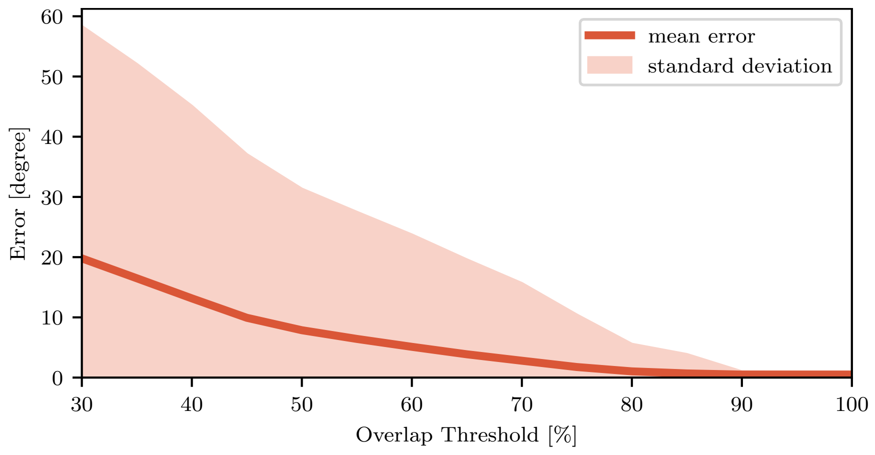


Figure 4.9: Overlap and yaw estimation relationship.

Table 4.4: Ablation study on usage of input modalities.

$\mathcal{R}$	$\mathcal{N}$	$\mathcal{I}$	$\mathcal{S}$	overlap AUC	overlap F1	yaw [deg] Mean	yaw [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

$\mathcal{R}$: range depth; $\mathcal{N}$: normal; $\mathcal{I}$: intensity; $\mathcal{S}$: semantic information.
AUC: area under the curve, F1: balanced F-score

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.

4.4.3.3 Relationship Between Overlap and Yaw Estimations

Figure 4.9 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 1 deg.

4.4.3.4 Ablation Study on Input Modalities

An ablation study on the usage of different inputs is shown in Table 4.4. As can be seen, when employing more input modalities, the proposed method becomes 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, our OverlapNet performs 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 makes the input data more distinguishable when the car drives in symmetrical environments.

4.4.4 OverlapNet-Based Loop Closing

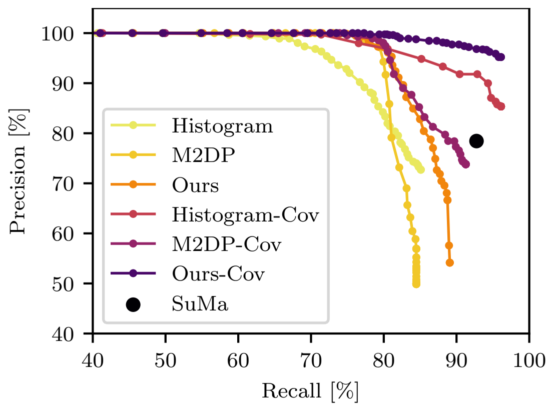
In the following experiments, we investigate the loop closing performance of our approach and compare it to existing methods. Different from the general place recognition, loop closure detection typically assumes that robots revisit places during the mapping while moving with uncertain odometry. Therefore, the prior information about robot poses extracted from the pose graph is available for 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 most recent 100 scans.
- For each query scan, we only consider the best candidate in 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 Figure 4.1. 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 Section 4.4.4.2.

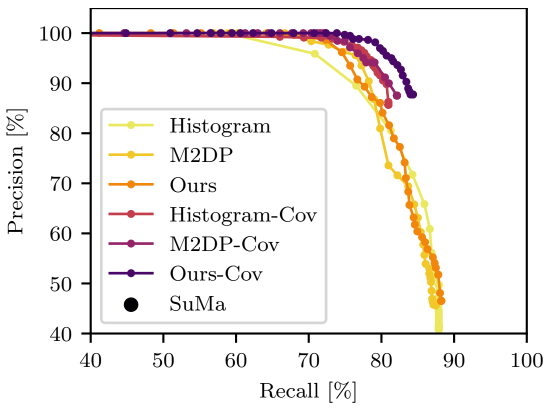
We evaluate OverlapNet on both the KITTI dataset and the Ford campus datasets to showcase the generalization capabilities of the approach.

4.4.4.1 Quantitative Analysis

Figure 4.10 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 (AC-TH), with three state-of-the-art approaches: a histo-gram-based approach (Histogram) by Röhling et al. [144], M2DP by He et al. [75], and the



(a) KITTI sequence 00



(b) Ford campus sequence 00

Figure 4.10: Precision-Recall curves of different approaches.

original heuristic approach of SuMa by Behley and Stachniss [15]. 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, called Ours-Cov, which uses covariance propagation 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 baselines with the proposed covariance-based method, named Histogram-Cov and M2DP-Cov respectively.

Table 4.5 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 datasets. 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, called 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 4.6. We compare our best model using all available cues and two heads labeled as AC-TH to a variant which only uses a basic multilayer perceptron as the head named MLPOnly which consists of two hidden fully connected layers

Table 4.5: 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**

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), does not have 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).

4.4.4.2 Application of 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 4.11 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 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

Table 4.6: Comparison with our variants.

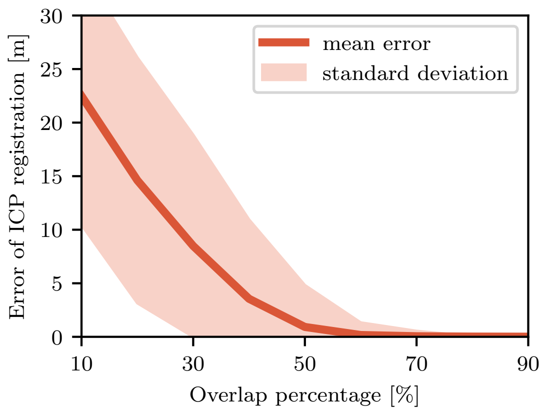
Dataset	Variant	AUC	F1 score
KITTI	MLPOnly	0.58	0.65
	DeltaOnly	0.85	0.88
	Ours (AC-TH)	0.87	0.88
	Ours-Cov (AC-TH)	**0.96**	**0.96**
Ford Campus	Ours (GO)	0.85	0.84
	Ours-Cov (GO)	**0.85**	**0.88**

closure: larger overlap values result in better registration results of the involved ICP.

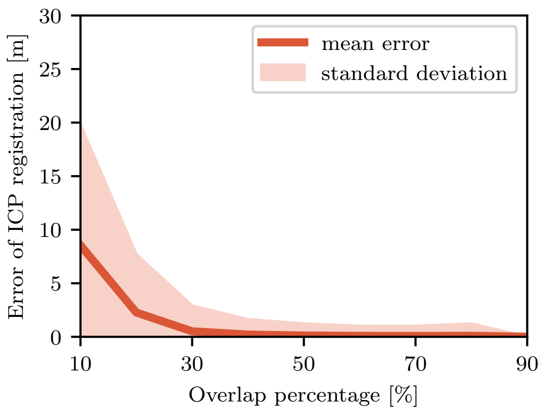
To analyze the correlation between the overlap and the 6 degree of freedom pose, we show the statistics with all sequences of both KITTI and Ford Campus datasets in Figure 4.12. As can be seen, with overlap values larger than 30%, the relative differences in roll, pitch are much smaller than that in 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 deg. 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 offsets of other orientation angles are small and can typically be handled by the following ICP that is used in general loop closing.

4.4.4.3 Improving SLAM Results

This experiment supports our claim that our method is able to improve the overall SLAM result. Figure 4.13 shows the odometry results on KITTI sequence 02. The color in Figure 4.13 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,



(a) With identity as initial guess



(b) With yaw angle as initial guess

Figure 4.11: 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.

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 Figure 4.1.

4.4.5 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 framework is 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.

4.4.5.1 Global Localization Baseline Methods

We compare our observation model with two baseline observation models: the typical beam-end model as introduced by Thrun et al. [178] and a histogram-based model derived from the work of Röhling et al. [144].

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 chapter, we implement the beam-end model with a down-sampled point cloud map using voxelization with a resolution of 10 cm.

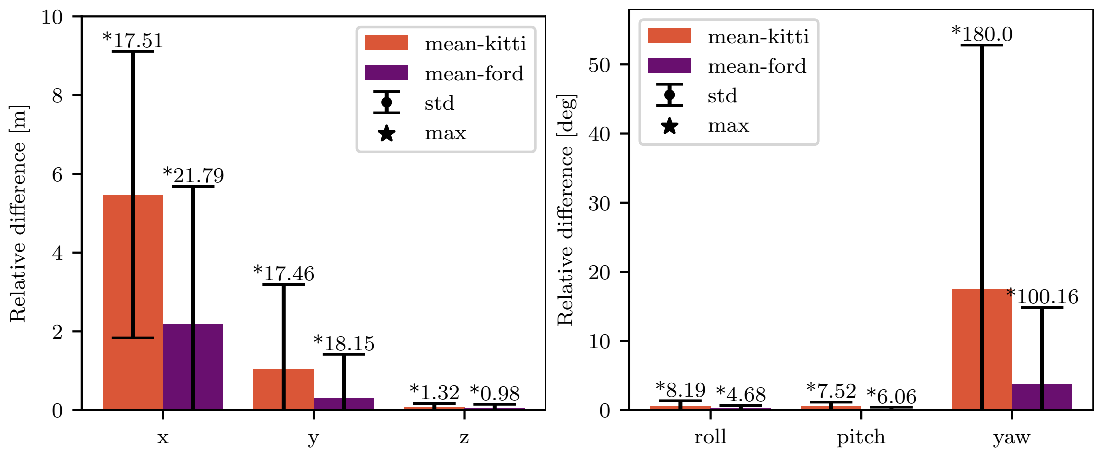


Figure 4.12: Statistics on relative 6 degree of freedom pose (x, y, z, roll, pitch, yaw) between pairs of scans with overlap larger than 30%. Results on KITTI and Ford campus datasets are shown.

Our second baseline for comparison uses the model proposed by Röhling et al. [144], which uses 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 we 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 current query scan and the virtual scans of grids where the particles locate. Then, we use the same Wasserstein distance [49] to measure the similarity between them and update the weights of particles as follows:

$$p(\boldsymbol{z}_t \mid \boldsymbol{x}_t) \propto d\left(h(\boldsymbol{z}_t), h(\boldsymbol{z}^{(i)})\right), \tag{4.16}$$

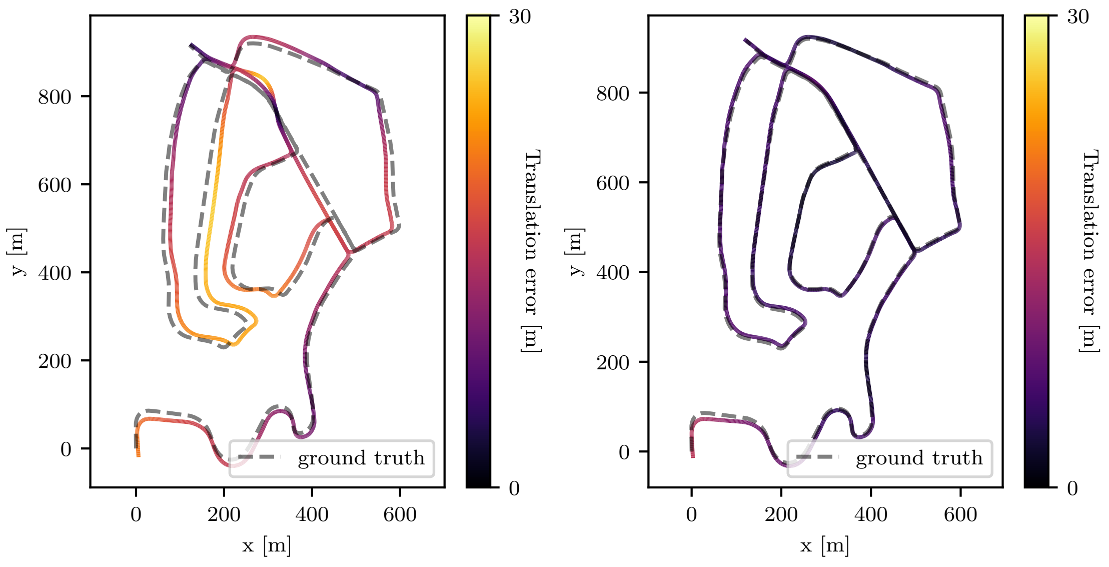
where d represents the Wasserstein distance between histograms $h(\boldsymbol{z}_t), h(\boldsymbol{z}^{(i)})$ of LiDAR scan $\boldsymbol{z}_t, \boldsymbol{z}^{(i)}$.

4.4.5.2 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 Figure 4.14. 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 Figure 4.15. The x-axis represents



(a) SuMa (b) Ours-Cov

Figure 4.13: Qualitative result on KITTI sequence 02 comparing SuMa to our approach.

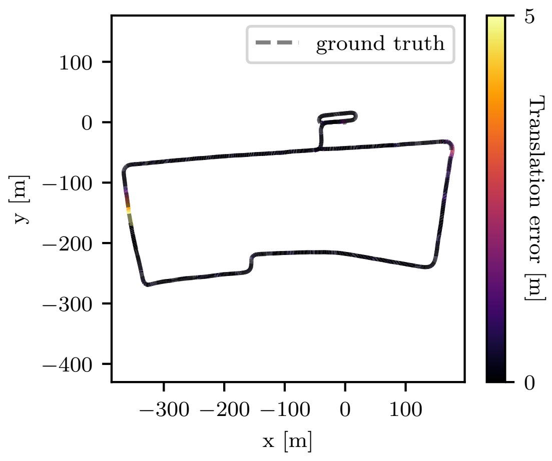
the number of particles used during localization, while the y-axis is the success rate of different setups. The success rate for a specific setup of one method is calculated using the number of success cases divided by the total number of the tests. To decide whether one test is successful or not, we check the location error 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 $N = \{1\,000;\ 5\,000;\ 10\,000;\ 50\,000;\ 100\,000\}$. For each setup, we sample 10 trajectories and perform global localization.

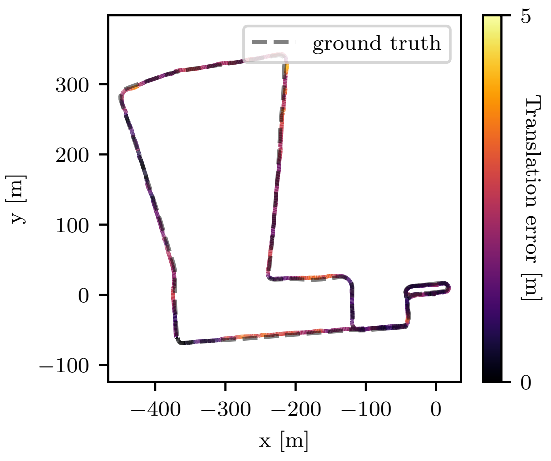
Quantitative results of localization accuracy are shown in Table 4.7. The upper part shows the location error of all methods tested with both sequences. The location error is defined as the root mean square error (RMSE) of each test in terms of the Euclidean error computed in (x, y) with respect to the 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



(a) Localization trajectory from sequence 00



(b) Localization trajectory from sequence 01

Figure 4.14: Localization results of our method with 10, 000 particles on two sequences recorded with the setup depicted in Figure 4.7. Shown are the ground truth trajectory (black) and our estimated trajectory using our observation model (red).

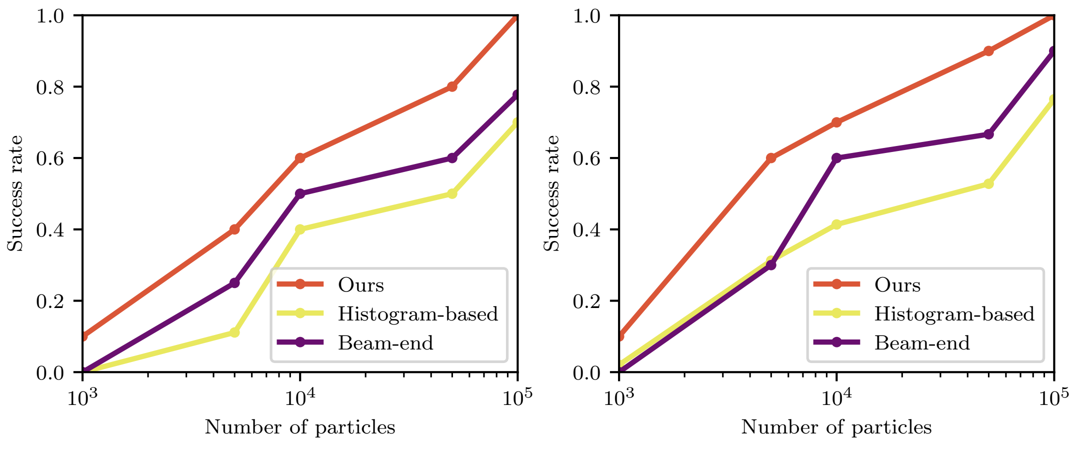
Table 4.7: Localization results on IPB-Car dataset

Sequence	Location error [meter]		
	Beam-end	Histogram	Ours
0	0.92 ± 0.27	1.85 ± 0.34	**0.81** ± **0.13**
1	**0.67** ± **0.11**	1.86 ± 0.34	0.88 ± 0.07

Sequence	Yaw angle error [deg]		
	Beam-end	Histogram	Ours
0	1.87 ± 0.47	3.10 ± 3.07	**1.74** ± **0.11**
1	2.10 ± 0.59	3.11 ± 3.08	**1.88** ± **0.09**

the proposed method faster than the 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 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 yielding 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.



(a) sequence 00 (b) sequence 01

Figure 4.15: Success rate of the different observation models for 10 globalization 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 converging to the ground truth location within 5 m.

4.4.6 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. 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 operates online, 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 Figure 4.16. This is a fairer way to compare the computational cost of different methods, since our neural network-based method uses a GPU to concurrently update 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 takes around 43 s to

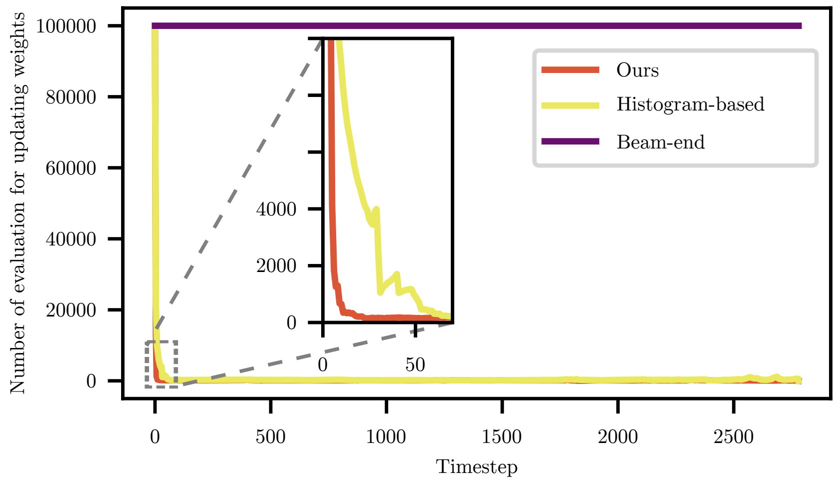


Figure 4.16: Number of observation model evaluations for updating the weights at each timestep with 100 000 particles. The beam end model needs to be evaluated for each and every particle individually. The histogram-based method is more computationally efficient, while our proposed method still needs the fewest evaluations.

process one scan. After convergence, the proposed method takes only 1 s on average to process one scan with 10 000 particles.

4.5 Conclusion

In this chapter, we proposed a modified Siamese network, OverlapNet, to estimate the similarity between pairs of LiDAR scans recorded by autonomous vehicles. 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 an MCL framework.

We evaluated our approach on multiple autonomous driving 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.

Part II

Task-Specific Semantics for Autonomous Vehicle Navigation

Chapter 5

Moving Object Segmentation for LiDAR SLAM

So far, we introduced in Part I how to use multiclass semantics generated from a segmentation network to improve the performance of LiDAR perception tasks like SLAM and localization for autonomous vehicles. Despite significant improvement in the performance of LiDAR SLAM and localization that can be achieved by exploiting multiclass semantics, accurate and reliable multiclass semantics are not always available. It is not only due to the lack of training data, but also because of the high difficulty of the multiclass semantic segmentation task itself, which makes the multiclass segmentation networks unable to generalize well in different environments.

In this part, instead of exploiting multiclass semantics, we introduce to use certain semantic classes that may be more useful than others in specific tasks and, at the same time, more easily to obtain. For example, for LiDAR-based SLAM, static or dynamic scene understanding can be leveraged to improve the performance of pose estimation and static map generation. While for localizing an autonomous vehicle reliably and precisely in an urban environment, pole-like objects are useful landmarks due to their local distinctiveness, natural availability, and long-term stability.

In this chapter, we address the problem of moving object segmentation in 3D LiDAR data at the sensor frame rate in urban environments. 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. Thus, accurate and reliable moving object segmentation (MOS) in sensor data at frame rate is a crucial capability for most autonomous mobile systems. Depending on the application domain and chosen sensor setup, moving object segmentation can be a challenging task.

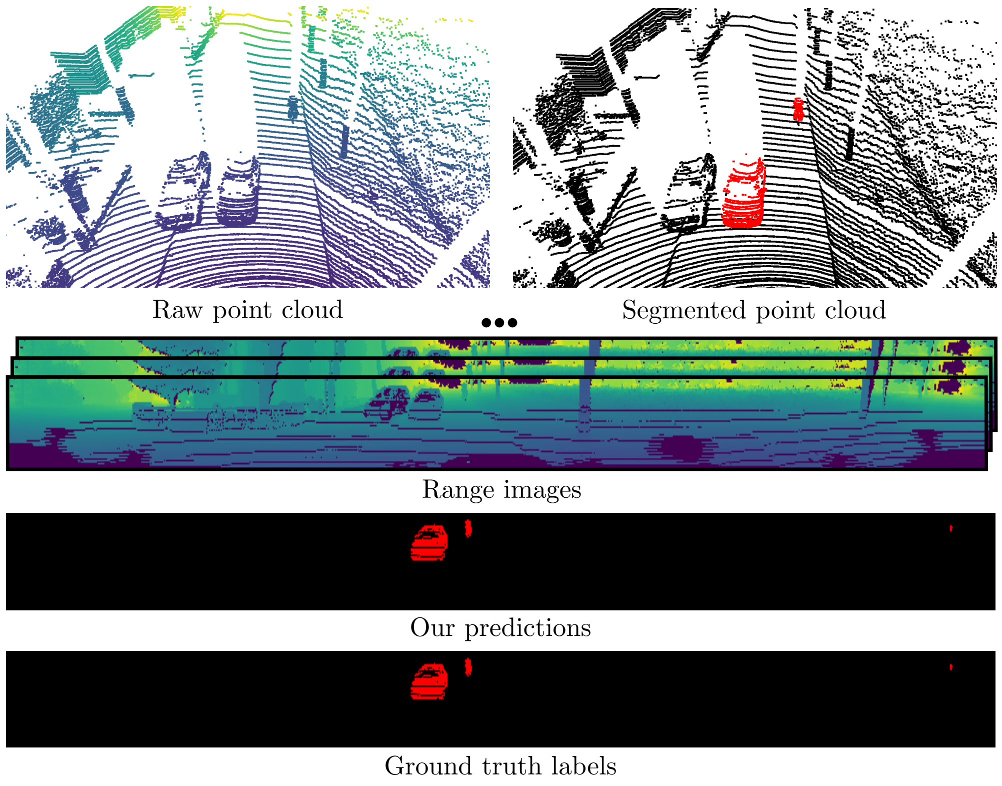


Figure 5.1: Moving object segmentation using our approach. Our method can detect and segment the currently moving objects given sequential point cloud data exploiting its range projection. Instead of detecting all *potentially movable* objects such as vehicles or humans, our approach distinguishes between *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.

Instead of detecting all *potentially movable* objects such as vehicles or humans, we aim at separating the *actually moving* objects, such as driving cars or walking pedestrians, from static or non-moving objects such as buildings, parked cars, etc. See Figure 5.1 for an example scene and our segmentation. Actually moving objects are colored in red. We propose a novel approach based on convolutional neural networks to explicitly address the MOS problem for 3D LiDAR scans. As introduced in Section 2.1, we stick to the LiDAR range image representation, which is a natural representation of the scan from a rotating 3D LiDAR such as a Velodyne or Ouster sensor. Based on this comparably lightweight representation, we can directly exploit the existing range-image-based semantic segmentation networks, such as RangeNet++ by Milioto et al. [120], SalsaNext by Cortinhal et al. [38], and MINet by Li et al. [101] 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 and static objects. 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.

The main contribution of this chapter is a novel method based on CNNs, which generates for each range measurement in the current scan a label indicating if it belongs to a moving object or not. It uses range images generated from 3D LiDAR scans together with the residual images generated from past scans as inputs. By combining range images and residual images, our network exploits the temporal information and can differentiate between moving and static objects as shown in Figure 5.1. For training, we reorganize the SemanticKITTI [13] 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: (i), 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. (ii), 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, including a hidden test set, based on the SemanticKITTI dataset and release the source code of our approach.

5.1 LiDAR-Based Moving Object Segmentation

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. Figure 5.2 shows a conceptual overview of our proposed method. To achieve online MOS, we first project the point clouds into range image representation (see Section 2.1). To separate moving and non-moving objects, we then exploit sequential information computing residuals between the current and the previous scans. We finally concatenate them together with the range information as the input for a segmentation CNN. In addition, we propose a novel MOS benchmark based on SemanticKITTI to train and evaluate MOS methods.

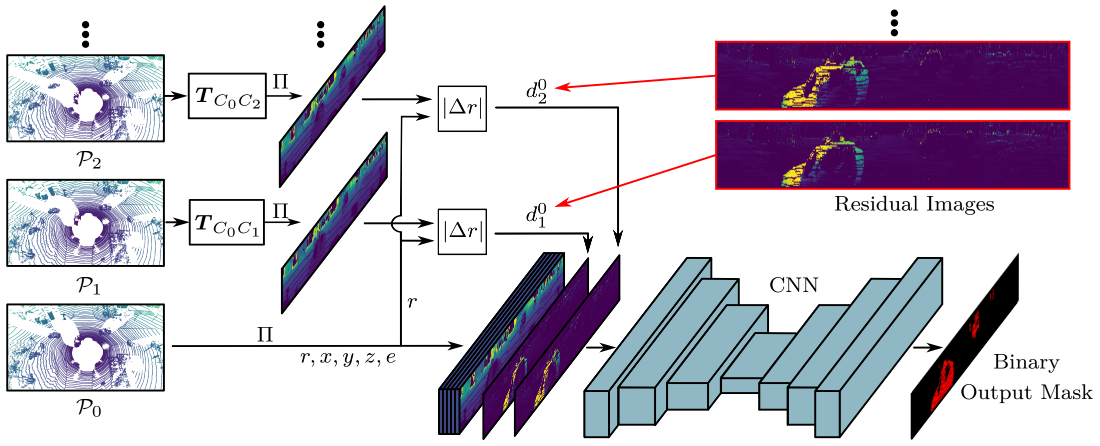


Figure 5.2: Overview of our method. We use the range projection-based representation of LiDAR scans to achieve online moving object segmentation. Given the current scan $\mathcal{P}_0$, we generate residual images from previous scans $\{\mathcal{P}_i\}_{i=1}^N$ to explore the sequential information. This is by transforming them to the current viewpoint with a homogeneous transformation matrix $\mathcal{T}_i$ 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.

5.1.1 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 $\mathcal{P}_j = \{\boldsymbol{p}_i \in \mathbb{R}^4\}$ with M points represented as homogeneous coordinates, i.e., $\boldsymbol{p}_i = (x, y, z, 1)^\top$. We call the current LiDAR scan by C_0 and the sequence of N previously scans by C_j with $1 < j < N$. The estimated N consecutive relative transformations from the SLAM / odometry approach, $\mathcal{T}_{C_0C_1}, \ldots, \mathcal{T}_{C_{N-1}C_N}$, between the $N+1$ scanning poses, represented as transformation matrices, i.e., $\mathcal{T} \in \mathbb{R}^{4\times4}$, 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

$$\mathcal{P}^{k\to l} = \{\mathcal{T}_{C_lC_k}\boldsymbol{p}_i \mid \boldsymbol{p}_i \in \mathcal{P}_k\}, \quad \text{with } \mathcal{T}_{C_lC_k} = \prod_{j=k}^{l+1} \mathcal{T}_{C_{j-1}C_j}. \tag{5.1}$$

5.1.2 Residual Images

In line with previous chapters, we use a range image representation of a point cloud. Inspired by Wang et al. [189], 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 the 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 the transformation estimates as defined in Equation (5.1). Next, the transformed past scans $\mathcal{P}^{k\to l}$ are re-projected into the current range image view using Equation (2.2). We compute the residual $d_i^{k\to l}$ for each pixel i by computing the normalized absolute difference between the ranges of the current frame and the transformed frame by

$$d_i^{k\to l} = \frac{|r^i - r_i^{k\to l}|}{r_i}\,, \tag{5.2}$$

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 use the normalized residual to keep the input data within the same range. 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 Figure 5.3. 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 desirable 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 the range image provides spatial information and residual images encode temporal information. Each pixel $(u_i, v_i)^\top$ in the fused range image then contains a vector $(x_i, y_i, z_i, r_i, e_i, d_i^{1\to 0}, ..., d_i^{j\to 0}, ..., d_i^{N\to 0})^\top$. $(x_i, y_i, z_i, r_i, e_i)^\top$ are the 3D coordinates, range, and intensity values. $d^{j\to 0}$ is the residual image calculated between the last j^{th} frame and the current frame.

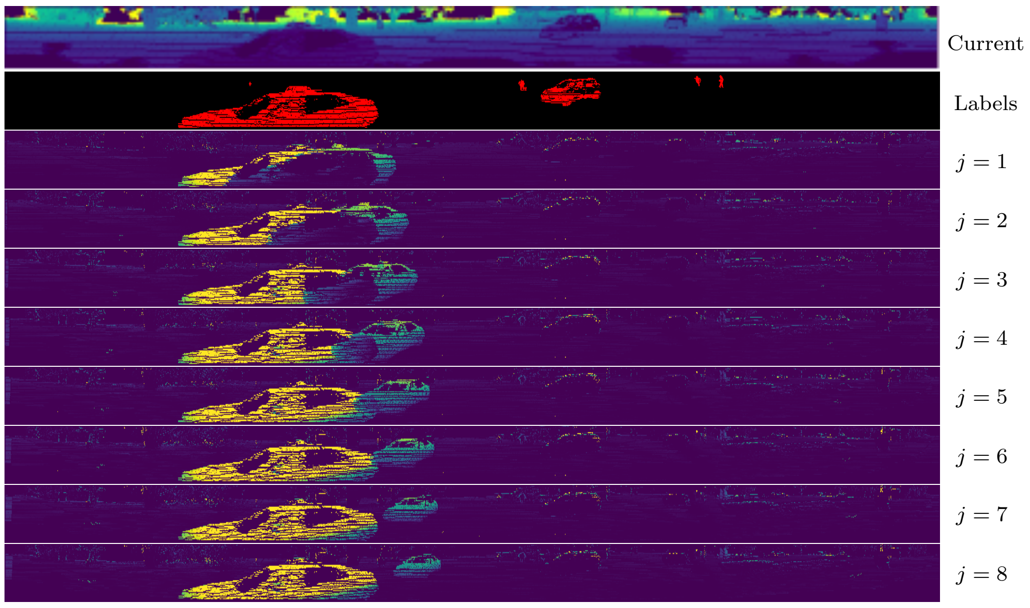


Figure 5.3: Residual images, where j means the residual image generated between the current frame and the last j-th frame. We can see the continuous discrepancy in the residual images due to the motion of the moving car.

5.1.3 Range Projection-Based Segmentation CNNs

In this chapter, 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 [38], RangeNet++ [120], and MINet [101], for MOS. SalsaNext and RangeNet++ are encoder-decoder architectures with a solid performance and MINet uses a lightweight 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 network and back projection (see Section 2.4). All methods are state-of-the-art range projection-based LiDAR semantic segmentation networks, comparably lightweight, 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 [38, 101, 120].

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. Using our proposed residual image approach, all segmentation networks show a large improvement in moving object segmentation. 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.

5.1.4 Moving Object Segmentation Benchmark

Large datasets for LiDAR-based odometry, object detection, and tracking, like the KITTI Vision Benchmark [67], and semantic segmentation, panoptic segmentation, and scene completion like SemanticKITTI [13] 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 of 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 [58] over moving objects, which is given by

$$\mathrm{IoU} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}\,, \tag{5.3}$$

where TP, FP, and FN correspond to the number of true positive, false positive, and false negative predictions for the moving class.

5.2 Experimental Evaluation

This chapter 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, now available online[1]. We use the odometry information provided by SemanticKITTI, which are estimated with a LiDAR-based SLAM system, SuMa [15]. 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

[1]See http://bit.ly/mos-benchmark for more information.

Table 5.1: Evaluating our method with three different networks in terms of IoU

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

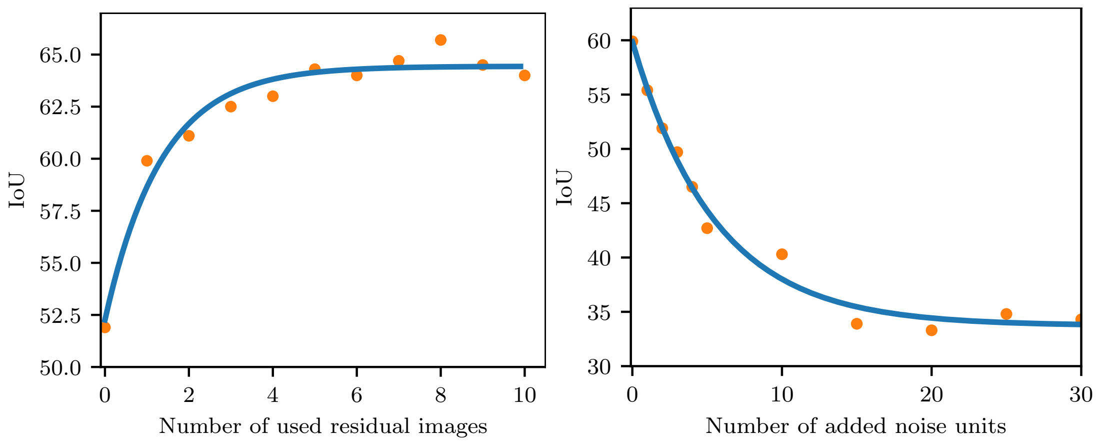


Figure 5.4: Ablation studies. The left figure shows the ablation study on the MOS performance vs. the number of residual images N. The right figure shows the ablation study on the MOS performance vs. the number of added noise units to the poses during the inferring.

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 [38, 101, 120].

5.2.1 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 Table 5.1. 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 without using the relative transformation, which results in 2×5 input channels, as each image contains the coordinates $(x, y, z)^\top$, 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 Section 5.1.2.

As can be seen in Table 5.1, RangeNet++ and SalsaNext show a fair 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 Figure 5.5. 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. [120].

As shown in Figure 5.4, 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 poses estimated by SLAM in $(x, y, \theta)^\top$ with multiples of $(0.1\,\text{m}, 0.1\,\text{m}, 1\,\text{deg})$ 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, $(2\,\text{m}, 2\,\text{m}, 20\,\text{deg})$, the noisy residual images do not further influence the network, and the MOS performance will not become worse.

Table 5.2: LiDAR-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
Spsequencenet	43.2
KPConv	60.9
Ours (based on SalsaNext/N = 1)	52.0
Ours (based on SalsaNext/N = 8 + Semantics)	**62.5**

Bold numbers indicate best performance in terms of IoU.

5.2.2 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.

We analyze multiple alternative approaches. We start using an existing semantic segmentation network, e.g., SalsaNext [38], 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 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 Residual. We use 1 m as the threshold to determine moving and non-moving. Inspired by Yoon et al. [206], we also re-implement the pure geometric heuristic-based method using residual information together with free space checking and region growing, named Residual+RG.

We furthermore compare our method also to the state-of-the-art scene flow

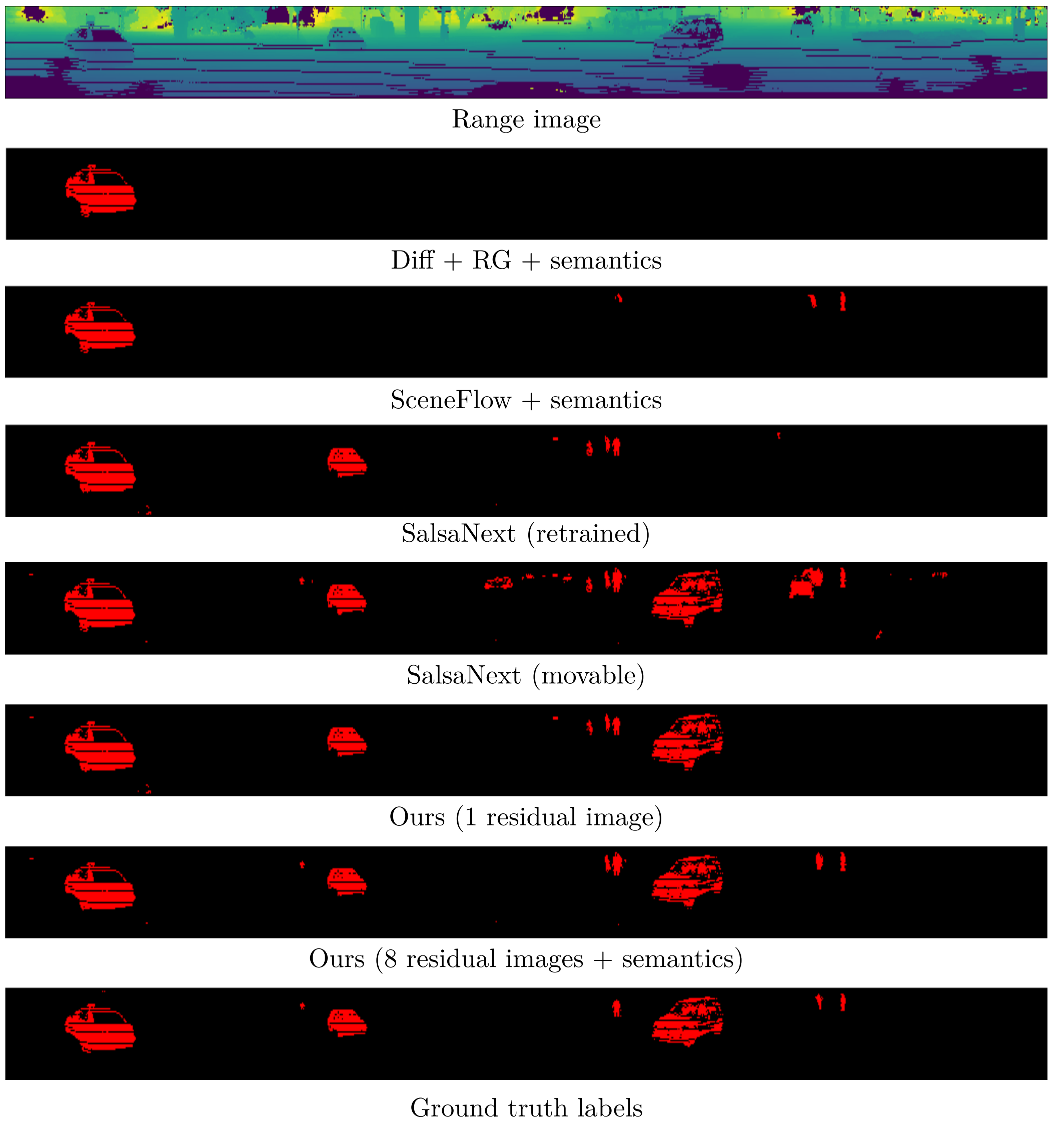
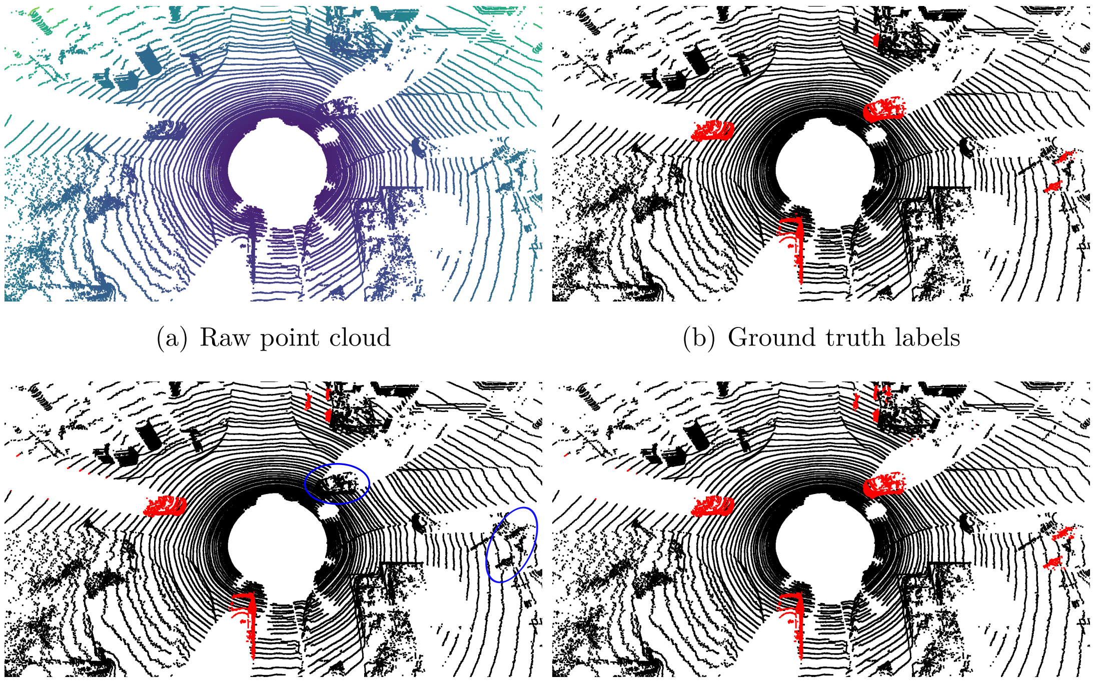


Figure 5.5: Qualitative results with range projections, where red pixels correspond to moving objects.

method, FlowNet3D [105], 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 1 m 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, KPConv [177] and Spsequencenet [159], 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


(a) Raw point cloud

(b) Ground truth labels

(c) SalsaNext (retrained)

(d) Ours

Figure 5.6: 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. Blue circles highlight wrong predictions.

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 Table 5.2, 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 the 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.

Figure 5.5 and Figure 5.6 show the qualitative results on range images and LiDAR scans respectively in a very challenging situation, where the car is at the

Table 5.3: 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**

In line with Table 3.2: relative rotational error in deg per 100 m / relative translational error in %.

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.

5.2.3 Applications

Two obvious applications of our proposed method are LiDAR-based odometry or SLAM and 3D mapping. Here, we show the effectiveness of our method by using the MOS predictions as masks for removing effectively all points belonging to moving objects in the input LiDAR scans. 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. Note that, here we show two direct use cases of our MOS approach without any further optimizations employed.

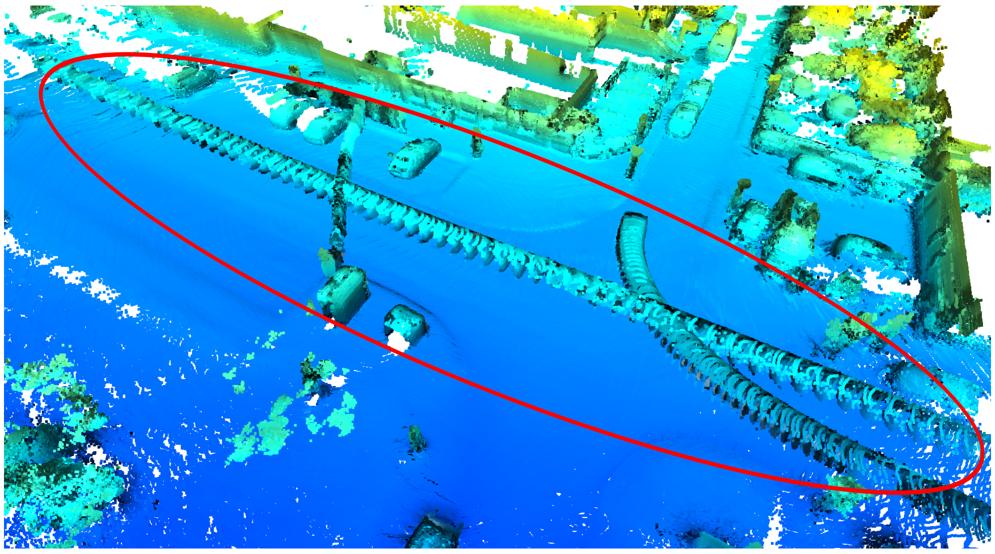
5.2.3.1 Odometry/SLAM

For the LiDAR-based odometry experiments, we use an off-the-shelf SLAM approach SuMa [15] 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 SuMa introduced in Section 2.3, and our semantic-enhanced approach, SuMa++ introduced in Chapter 3. We evaluate these odometry methods, SuMa, SuMa++, and SuMa+MOS on the KITTI odometry benchmark [67].

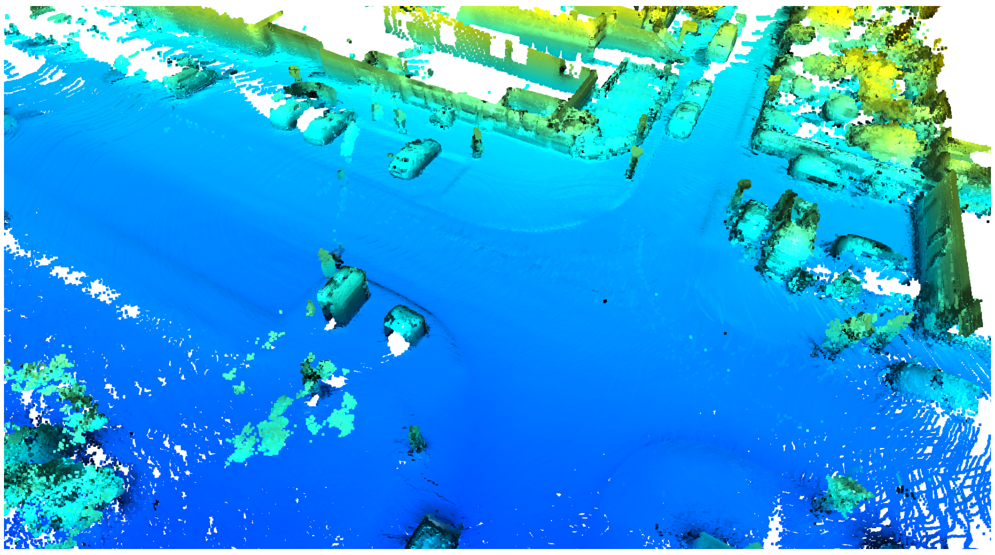
The quantitative results are shown in Table 5.3. We can see that, by simply applying our MOS predictions as a preprocessing mask, the odometry results are improved in both the KITTI training and test data and even slightly better than the well-designed semantic-enhanced SuMa.

5.2.3.2 3D Mapping

As shown in Figure 5.7, 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. As can be seen, there are moving objects present that



(a) Raw point clouds



(b) Point clouds with moving segments removed

Figure 5.7: 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. Red circle highlights artifactes caused by moving objects.

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.

5.2.4 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 the original one, specifically 75 ms for RangeNet++, 42 ms for SalsaNext, and 21 ms for MINet. In the 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 scan. 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.

5.3 Conclusion

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 chapter, 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, buildings, roads, etc., our approach accurately segments the scene into moving and static objects, i.e., 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 publish it to allow other researchers to compare their approaches transparently. The link to the benchmark is `http://bit.ly/mos-benchmark`.

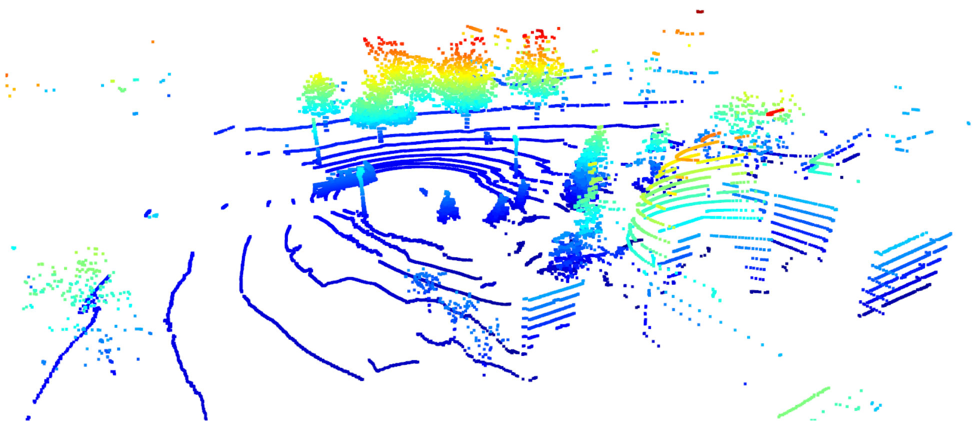
Chapter 6

Pole-Like Object Detection for LiDAR Localization

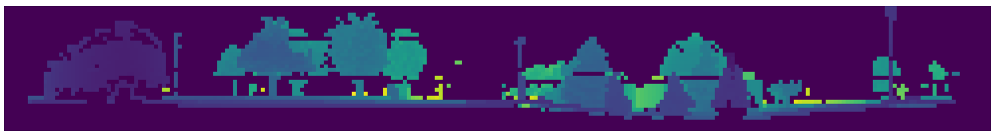
DIFFERENT from SLAM, which benefits more from distinguishing moving and non-moving objects, for localization, pole-like objects, such as traffic signs, poles, lamps, etc., are frequently used as landmarks in urban environments due to their local distinctiveness and long-term stability. In this chapter, we present a novel, accurate, and fast pole extraction approach that runs online and has little computational demands such that this information can be used for a localization system.

Robust and reliable localization is a basic requirement for an autonomous robot. The accurate estimation of the robot's pose helps to avoid collisions, navigate in a goal-directed manner, follow the traffic lanes, and perform other tasks. Reliability means here that the robot should adapt to changes in the environment, such as different weather conditions [28], day and night [180], or seasonal changes [110].

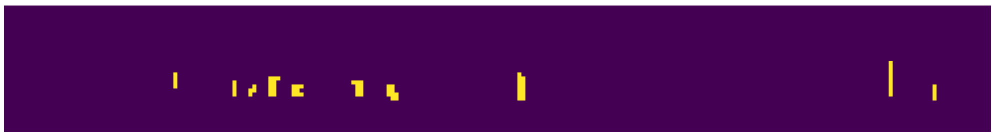
GPS or GNSS-based localization systems are robust to appearance changes of the environment. However, in urban areas, they may suffer from low availability due to building and tree occlusions. Additionally, map-based approaches are needed for precise and reliable localization of mobile robots. Multiple different types of sensors have been used to build the map of the environments, such as LiDAR scanners [15, 52, 185], monocular cameras [125], stereo cameras [40], etc. Among them, LiDAR sensors are more robust to the illumination changes and multiple LiDAR-based effective and efficient mapping approaches have been proposed. However, these approaches often need large amounts of memory due to their map representations, thus cannot generalize easily to large-scale scenes. If only specific features are used to build the map, such as traffic signs, trunks and other pole-like structures, the map size can be reduced significantly.



(a) Current scan



(b) Current range image



(c) Pole extraction result by our approach

Figure 6.1: Visualization of range image and pole extraction results. (a) On the top is the raw LiDAR scan. (b) The corresponding range image generated from this scan is in the middle. (c) The bottom is the pole extraction result based on the range image.

The main contribution of this chapter is a novel range image-based pole extractor for long-term localization of autonomous mobile systems. Instead of using directly the raw point clouds obtained from 3D LiDAR sensors, we stick to the use of range images for pole extraction. Operating on range image representation is considerably faster than on the raw 3D point cloud. Besides, range image keeps the neighborhood information implicitly in its 2D structure and we can use this information for segmentation. As shown in Figure 6.1, in the mapping phase, we first project the raw point cloud into a range image and then extract poles from that. After obtaining the position of poles in the range image, we use the ground truth poses of the robot to reproject them into the global coordinate system to build a global map. During localization, we first use our OverlapNet introduced in Chapter 4 to find possible locations on the global map/database. We sample particles for each hypothesis and then utilize Monte Carlo localization for updating the importance weights of the particles by matching the poles detected from online sensor data with the poles in the global map. In the end, the particles converge to the correct location, and our approach achieves global localization.

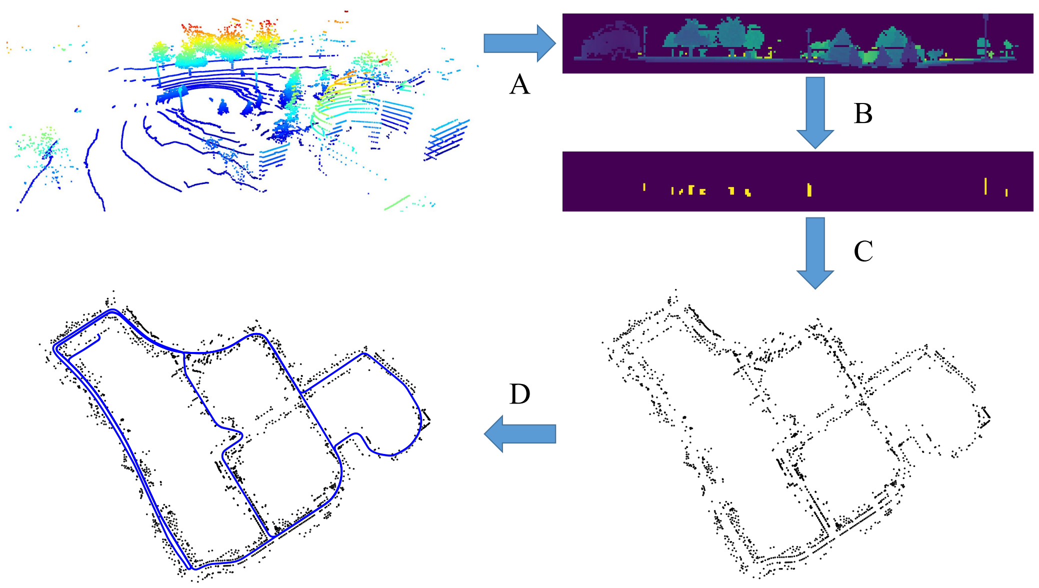


Figure 6.2: Overview of our approach. A. We project the LiDAR point cloud into a range image and B. extract poles in the image. C. Based on the extracted poles, we then build a 2D global pole map of the environment. D. We finally propose a pole-based observation model for MCL to localize the robot in the map.

In sum, we make three key claims: Our approach is able to (i) extract poles in the environment more reliably compared to the baseline method, (ii) as a result, achieves better localization performance in different environments, and (iii) runs online for pose tracking. These claims are backed up by the chapter and our experimental evaluation.

6.1 Pole-Based LiDAR Localization

In this chapter, we propose a range image-based pole extractor for long-term localization using a 3D LiDAR sensor. As shown in Figure 6.2, we first project the LiDAR point cloud into a range image and extract poles from it. Based on the proposed pole extractor, we then build a global pole map of the environment. In the localization phase, we first use our OverlapNet to find possible locations by comparing the current range image to those stored in the database. We then extract poles online in the current scan using the same extractor and use a novel pole-based observation model for Monte Carlo localization.

6.1.1 Pole Extraction

The key idea of our approach is to use range images generated from LiDAR scans for pole extraction. We utilize the same spherical projection for range image generation as used in the previous chapters. We extract poles based on the range images using a heuristic approach. The general intuition behind our pole extraction algorithm is that the range values of the poles are usually significantly smaller than the background. Based on this idea and as specified in Algorithm 3, our first step is to cluster the pixels of the range image into different small regions based on their range values. We first pass through all pixels in the range image, from top to bottom, left to right.

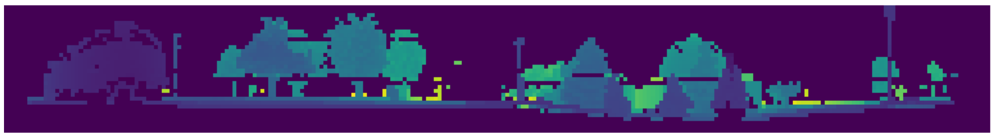
As introduced in Section 2.1, each pixel $(u_i, v_i)^\top$ in the range image stores the corresponding 3D point $\boldsymbol{p} = (x, y, z)^\top$, and the range information is computed by $r = \|\boldsymbol{p}_i\|_2$. We put all pixels with valid range data in an open set $\mathcal{O}$. For each pixel, we check its neighbors including the left, right and below ones. If there exists a neighbor with a valid value and the range difference between the current pixel and its neighbor is smaller than a threshold ϵ_d, we add the current pixel to a cluster $\mathcal{C}$ and remove it from the open set $\mathcal{O}$. We do the same check iteratively with the neighbors until no neighbor pixel meets the above criteria resulting in a cluster of pixels. After checking all the pixels in $\mathcal{O}$, we get a set $\mathcal{B}$ of clusters and each cluster represents one object. If the number of pixels in one cluster is smaller than a threshold ϵ_n, we regard it as an outlier and ignore it.

The next step is to extract poles from these objects using geometric constraints. To this end, we exploit both the range information and the 3D coordinates stored in each pixel. We first check the aspect ratio of each cluster. Since we are only interested in pole-like objects, whose height is usually larger than its width, we therefore discard clusters with aspect ratio $h/w < 1$. Another heuristic we use is the fact that a pole usually stands alone and has a significant distance from background objects. N_{SmallR} is the number of points in cluster $\mathcal{C} \in \mathcal{B}$ whose range value is smaller than its neighbor outside $\mathcal{C}$, we throw away the cluster if N_{SmallR} is smaller than δ_1 times the number of all points in the cluster.

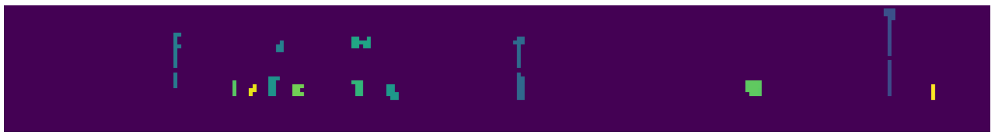
To exploit the 3D coordinates $(x, y, z)^\top$ of each pixel, we calculate the length of each cluster and only take a cluster as a pole candidate if $\max(z) - \min(z) > \epsilon_h$. Besides, we are only interested in poles whose height is higher than H_a. Based on experience, we also set the lowest position of the pole as H_b to filter outliers. For each pole candidate, we then fit a circle using the x and y coordinates of all points in the cluster and get the center and the radius of that pole. Since the generation of range image loses some point cloud information, thus influences fitting accuracy. We deal with this problem by finding all points inside the circle plus a small distance and refitting the circle using these enriched points. We filter out the candidates with too small or too large radiuses R_a, R_b respectively and candidates that connect to other objects by checking the free space around them. After the above steps, we finally extract the positions and radiuses of poles. As an example, Figure 6.3 visualizes the intermediate results on each step of our geometric pole extractor.

Algorithm 3: Range image-based pole extraction.

```
Input: Range image $\mathcal{I}_{\text{range}}$
Output: Pole parameters $\mathcal{L}$ with circle centers and radiuses
1  Let $\mathcal{O}$ be the set of all valid pixel coordinates in $\mathcal{I}_{\text{range}}$. $\epsilon_d$ is the distance
     threshold to find neighbors, $\epsilon_n$ is the pixel count threshold, and $\epsilon_h$ is the
     object height threshold. $H_a$ and $H_b$ are the height lower and upper bounds.
     $R_a$ and $R_b$ are the radius lower and upper bounds. $\delta_1$ and $\delta_2$ are scale factors.
2  while $\mathcal{O} \neq \emptyset$ do
3      create a new $\mathcal{C}$ in $\mathcal{B}$; $\boldsymbol{p} \leftarrow \mathcal{O}[0]$
4      $\mathcal{N} \leftarrow$ Neighbor($\boldsymbol{p}$)
5      while $\mathcal{N} \neq \emptyset$ do
6          $\boldsymbol{n} \leftarrow \mathcal{N}[0]$
7          if $\boldsymbol{n} \in \mathcal{O}$ and Distance $(\boldsymbol{n}, \boldsymbol{p}) < \epsilon_d$ then
8              $\mathcal{C} \leftarrow \mathcal{C} \cup \{\boldsymbol{p}\}$; $\boldsymbol{p} = \boldsymbol{n}$
9              $\mathcal{N} \leftarrow \mathcal{N} \cup$ Neighbor($\boldsymbol{n}$)
10         end
11     end
12     $N_{\text{pixel}} \leftarrow$ the number of pixels in $\mathcal{C}$
13     if $N_{\text{pixel}} < \epsilon_n$ then
14         $\mathcal{B} \leftarrow \mathcal{B} - \mathcal{C}$
15     end
16 end
17 foreach $\mathcal{C} \in \mathcal{B}$ do
18     $w, h \leftarrow$ Width($\mathcal{C}$), Height($\mathcal{C}$)
19     $N_{\text{SmallR}} \leftarrow$ the number of pixels in $\mathcal{C}$ whose range value is smaller than its
         neighbor outside $\mathcal{C}$
20     if $h/w < 1$ or $N_{\text{SmallR}} < \delta_1\ |\mathcal{C}|$ then
21         $\mathcal{B} \leftarrow \mathcal{B} - \mathcal{C}$
22     end
23 end
24 foreach $\mathcal{C} \in \mathcal{B}$ do
25     $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \leftarrow$ 3D coordinates of pixels in $\mathcal{C}$
26     if max($\boldsymbol{z}$) $> H_a$ and min($\boldsymbol{z}$) $< H_b$ and (max($\boldsymbol{z}$) $-$ min($\boldsymbol{z}$)) $> \epsilon_h$ then
27         $x_{\mathcal{C}}, y_{\mathcal{C}}, r_{\mathcal{C}} \leftarrow$ FitCircle($\boldsymbol{x}, \boldsymbol{y}$)
28         $N_{\text{Free}} \leftarrow$ the number of the pixels in a small free space outside the
             radius of the pole
29         if $r_{\mathcal{C}} < R_a$ and $r_{\mathcal{C}} > R_b$ and $N_{\text{Free}} < \delta_2\ |\boldsymbol{z}|$ then
30             $\mathcal{L} \leftarrow \mathcal{L} \cup \{x_{\mathcal{C}}, y_{\mathcal{C}}, r_{\mathcal{C}}\}$
31         end
32     end
33 end
```


(a) LiDAR range image

(b) Clustering results


(c) Pole candidates

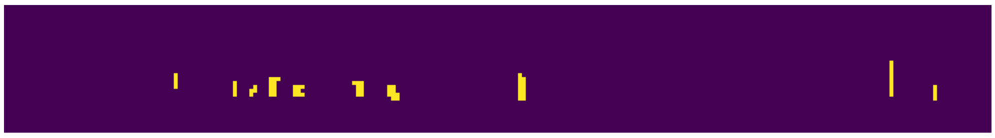

(d) Final pole results

Figure 6.3: Visualization of results on each step of our pole extractor. (a) The first image shows the original range image. (b) The second represents the clustering result. (c) The third shows the pole candidates after applying geometric constraints. (d) The last one is the final pole extraction result.

6.1.2 Mapping

To build the 2D global pole map for localization, we follow the same setup as introduced by Schaefer et al. [153], splitting the ground truth trajectory into shorter sections with equal lengths. Since the provided poses are not very accurate for mapping [153], instead of aggregating a noisy submap, we only use the middle LiDAR scan of each section to extract poles. We merge multiple overlapped pole detections by averaging over their centers and radiuses and apply a counting model to filter out the dynamic objects. Only those poles that appear multiple times in continuous sections are regarded as real poles.

To achieve global localization, we also store the leg features of all the map scans generated by our OverlapNet as a place recognition database. Each leg feature is a lightweight 1D vector describing one map scan. Therefore, the descriptor-based database is also relatively much smaller than the raw point cloud map. Similar to our overlap-based global localization presented in Section 4.3, we use our OverlapNet to estimate the similarities between the current query scan and the map scans to find the possible locations as the initial hypotheses. Unlike overlap-localization, we only store the features generated from the real map scans obtained by the LiDAR sensor but not from the rendered synthetic scans. We use the descriptor database only to find initial global hypotheses to sample particles and use the pole map for final accurate pose tracking.

6.1.3 Pole-Based Localization

We achieve global localization in two steps. In the first step, we use our OverlapNet to generate a descriptor for the current range image and find the most N_o similar places in the database to initialize the particles. For each location hypothesis of these top N_o similar places, we sample N_p particles. Then, we propose a novel pole-based observation model to update the importance weights of particles and integrate it into the MCL framework to achieve accurate global localization.

As presented in Section 2.2, MCL is one of the classic frameworks for localization and commonly implemented using a particle filter. It realizes a recursive Bayesian filter estimating a probability density $p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t})$ over the pose $\boldsymbol{x}_t$ given all observations $\boldsymbol{z}_{1:t}$ and motion controls $\boldsymbol{u}_{1:t}$ up to time t. This posterior is updated as follows:

$$p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) = \eta \; p(\boldsymbol{z}_t \mid \boldsymbol{x}_t) \int p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1}) \; p(\boldsymbol{x}_{t-1} \mid \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t-1}) \; d\boldsymbol{x}_{t-1}, \tag{6.1}$$

where η is a normalization constant, $p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ is the motion model, $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$ is the observation model, and $p(\boldsymbol{x}_{t-1} \mid \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t-1})$ is the probability distribution for the prior state $\boldsymbol{x}_{t-1}$.

In our case, each particle represents a hypothesis for the 2D pose $\boldsymbol{x}_t = (x, y, \theta)_t^\top$ of the robot at time t. When the robot moves, the pose of each particle is updated based on a motion model with the control input $\boldsymbol{u}_t$ or the odometry measurements. For the observation model, the weights of the particles are updated based on the difference between expected observations and actual observations. The observations are the positions of the poles. We match the online observed poles with the poles in the map via nearest-neighbor search using a k-d tree [17]. The likelihood of the j-th particle is then approximated using a Gaussian distribution:

$$p\left(\boldsymbol{z}_t \mid \boldsymbol{x}_t\right) \propto \prod_{i}^{N_m} \exp\left(-\frac{1}{2} \frac{d\left(\boldsymbol{z}_t^i, \boldsymbol{z}^{(j)i}\right)^2}{\sigma_d^2}\right), \tag{6.2}$$

where d corresponds to the difference between the online observed pole $\boldsymbol{z}_t^i$ and matched pole in the map $\boldsymbol{z}^{(j)i}$ given the position of the particle j. N_m is the number of matches of current scan. If the query pole and its nearest neighbor are not overlapped, we directly multiple a fixed penalty. Otherwise, we use the Euclidean distance between the pole center positions to measure this difference.

6.2 Experimental Evaluation

The main focus of this work is an accurate and efficient pole extractor for long-term LiDAR localization. We present our experiments to show the capabilities of our method and to support our key claims, that our method is able to: (i) better extract poles compared to the baseline method, (ii) as a result, achieve better performance on long-term localization in different environments, and (iii) achieve online operation for pose tracking.

6.2.1 Pole Extractor Performance

The first experiment evaluates the pole extraction performance of our approach and supports the claim that our range image-based method outperforms the baseline method in pole extraction.

There are few public datasets available to evaluate pole extraction performance. To this end, we label the poles in session 2012-01-08 of the NCLT dataset by hand and release this dataset for public research use. For the reason that the original NCLT ground truth poses are inaccurate [153], the aggregated point cloud is a little blurry. Therefore, to create the ground truth pole map of the environment, we partition the ground truth trajectory into shorter segments of equal length. For each segment, we aggregate the point cloud together and use Open3D [214] to render and label the pole positions. We only label those poles with high certainty and ignore those blurry ones. Besides our own labeled data, we also reorganize the SemanticKITTI [13] dataset sequence 00-10 by extracting the pole-like objects, e.g., traffic signs, poles, and trunk, and then clustering the point clouds to generate the ground truth pole instances. We evaluate our method and compare it to the state-of-the-art pole-based LiDAR localization method proposed by Schaefer et al. [153] in both datasets.

During the matching phase, we find the matches via nearest-neighbor search using a k-d tree with 1 m distance bounds, which is the same way as used in our proposed pole-based observation model. Table 6.1 summarizes the precision, recall, and F1 score of our method and Schaefer et al. [153] compared to the ground truth pole map on both the NCLT dataset and SemanticKITTI dataset. As can be seen, our method has better performance in terms of all metrics compared to the baseline method. The higher recall and precision results indicate that our method extracts more poles in both environments with better accuracy.

6.2.2 Localization Performance

The second part of experiments is presented to support the claim that our approach achieves higher accuracy on localization in different environments. To as-

Table 6.1: Pole extraction accuracy on NCLT and KITTI datasets.

Dataset	Method	Precision	Recall	F1 Score
NCLT	Schaefer [153]	0.52	0.66	0.58
	Ours	**0.53**	**0.71**	**0.61**
Semantic KITTI	Schaefer [153]	0.62	0.38	0.46
	Ours	**0.68**	**0.44**	**0.52**

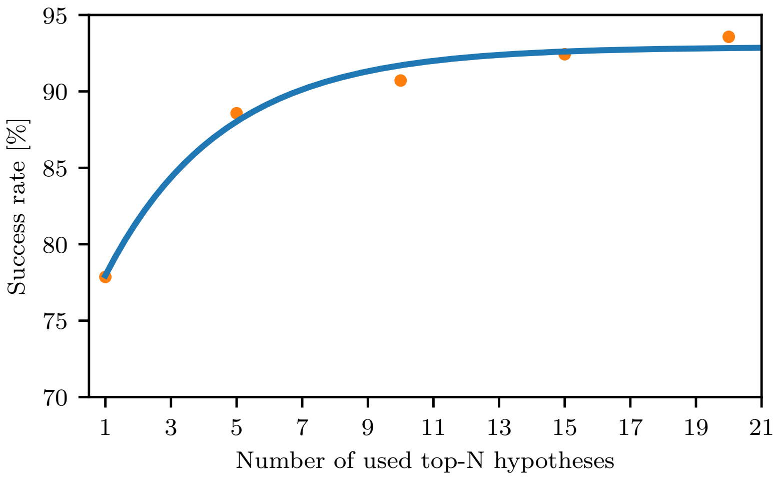


Figure 6.4: Pole-based localization success rate. We vary the used number of hypotheses and sample 200 particles around each hypothesis. We test each setup 10 times on all the test sequences of NCLT and calculate the average success rate.

sess the localization reliability and accuracy of our method, we use three datasets for evaluation, the NCLT dataset [28], and MulRan dataset [90]. Note that, these two datasets are collected in different environments (U.S., Korea) with different LiDAR sensors (Velodyne HDL-32E, Ouster OS1-64). In the NCLT and MulRan dataset, the robot passes through the same place multiple times with month-level temporal gaps, hence ideal to test the long-term localization performance. We compare our methods to both the pole-based method from Schaefer et al. [153] and our previously proposed range image-based method [36], named RangeMCL. We use the SemanticKITTI dataset to evaluate the pole extraction performance but do not use it for localization, since there is no long-term repetitive visiting of the same places. For our pole-based method and the method by Schaefer et al. [153], we use our OverlapNet to provide the initial hypotheses for particle initialization, while RangeMCL also achieves global localization using range images via particle filter but without using pole information.

The NCLT dataset contains 27 sessions with an average length of 5.5 km and an average duration of 1.3 h over the course of 15 months. The data was recorded by onboard sensors mounted on a mobile robot. The dataset covers different times over a year, different weather and seasons, including both indoor and outdoor environments, and also lots of dynamic objects. The trajectories of different sessions have a large overlap. Therefore, it is an ideal dataset for testing long-term localization in urban environments.

We first build the map following the setup introduced by Schaefer et al. [153], which uses the laser scans and the ground truth poses of the first session. Since in later sessions, the robot sometimes moves into unseen places for the first session, we therefore also use those scans whose position is 10 m away from all previously visited poses to build the map. We also use such map scans to train our OverlapNet and build the feature database for providing initial global hypotheses.

During localization, we first use the top $N_o = 20$ place candidates provided by our OverlapNet. Then, we uniformly sample $N_p = 200$ positions around each location candidate within a 2.5 m circle. The orientations are uniformly sampled from -5 to 5 deg. We resample particles if the ratio of effective particles is less than 0.5. To get the pose estimation, we use the average poses of the best 10 % of the particles. We compare the localization results after the particles converge successfully. We consider one run as a success if all particles converge into the correct location.

Figure 6.4 shows the success rate with using different numbers of top-N hypotheses we use for the MCL initialization. We vary the used number of hypotheses and sample $N_p = 200$ particles around each hypothesis. We test each setup 10 times on all the test sequences of NCLT and calculate the average success rate. As can be seen, when using $N_o = 20$ place candidates, our method has a high success rate to find the correct global pose of the robot. After converging, we use only 1000 particles for pose tracking to ensure the online operation. We use this setup for all experiments below.

6.2.2.1 Localization on the NCLT Dataset

Table 6.2 shows the position and orientation errors for every session. We run the localization 10 times and compute the average means and RMSEs to the ground truth trajectory. The results show that our method surpasses Schaefer et al. [153] in almost all sessions with an average error of 0.17 m. Besides, in session 2012-02-23, the baseline method fails to localize resulting in an error of 2.47 m, while our method never loses track of the robot position (Figure 6.5). This is because our pole extractor can robustly extract poles and reliably localize the robot even in an environment with fewer poles.

Session Date	f_{map} [%]	Δ_{pos} [m]		RMSE_{pos} [m]		Δ_{ang} [deg]		RMSE_{ang} [deg]	
		Schaefer's	Ours	Schaefer's	Ours	Schaefer's	Ours	Schaefer's	Ours
2012-01-08	100.0	0.13	**0.12**	0.18	**0.15**	0.66	**0.63**	0.86	**0.81**
2012-01-15	8.5	0.16	**0.15**	0.23	**0.20**	0.76	**0.75**	1.00	**0.98**
2012-01-22	5.1	0.17	**0.15**	0.22	**0.19**	0.94	**0.91**	1.29	**1.24**
2012-02-02	0.4	0.16	**0.14**	0.21	**0.17**	0.72	**0.70**	0.98	**0.92**
2012-02-04	0.1	0.14	**0.13**	0.20	**0.17**	0.68	**0.67**	0.90	**0.88**
2012-02-05	0.5	0.15	**0.14**	0.21	**0.20**	**0.69**	**0.69**	0.95	**0.94**
2012-02-12	0.8	0.27	**0.25**	1.01	**1.00**	0.80	**0.79**	1.04	**1.02**
2012-02-18	0.8	0.15	**0.13**	0.22	**0.18**	0.70	**0.68**	0.94	**0.91**
2012-02-19	0.1	0.15	**0.14**	0.19	**0.18**	0.70	**0.69**	0.94	**0.92**
2012-03-17	0.0	0.15	**0.14**	0.19	**0.17**	0.83	**0.80**	1.06	**1.03**
2012-03-25	0.0	0.20	**0.18**	0.26	**0.24**	1.42	**1.38**	1.84	**1.79**
2012-03-31	0.0	0.14	**0.14**	0.18	**0.18**	0.75	**0.73**	0.97	**0.94**
2012-04-29	0.0	0.17	**0.15**	0.25	**0.22**	0.83	**0.82**	1.08	**1.07**
2012-05-11	5.5	0.16	**0.13**	0.23	**0.16**	0.77	**0.75**	1.00	**0.97**
2012-05-26	0.4	0.16	**0.14**	0.22	**0.18**	0.69	**0.67**	0.89	**0.87**
2012-06-15	0.4	0.18	**0.15**	0.24	**0.19**	0.66	**0.65**	0.88	**0.87**
2012-08-04	0.3	0.21	**0.17**	0.34	**0.23**	0.88	**0.84**	1.14	**1.09**
2012-08-20	3.8	0.19	**0.16**	0.26	**0.21**	0.71	**0.69**	0.94	**0.91**
2012-09-28	0.3	0.21	**0.17**	0.31	**0.24**	0.73	**0.72**	0.95	**0.94**
2012-10-28	1.4	0.22	**0.19**	0.34	**0.28**	0.69	**0.68**	0.92	**0.91**
2012-11-04	2.5	0.26	**0.21**	0.46	**0.32**	0.75	**0.72**	1.00	**0.97**
2012-11-16	2.7	0.40	**0.30**	0.72	**0.44**	1.47	**1.40**	2.03	**1.92**
2012-11-17	0.4	0.24	**0.20**	0.38	**0.32**	0.69	**0.68**	0.96	**0.95**
2012-12-01	0.0	0.27	**0.23**	0.49	**0.43**	0.67	**0.66**	0.93	**0.89**
2013-01-10	0.0	0.22	**0.19**	0.28	**0.23**	0.69	**0.63**	0.91	**0.81**
2013-02-23	0.0	2.47	**0.24**	5.48	**0.57**	1.08	**0.59**	1.77	**0.85**
2013-04-05	0.0	0.37	**0.30**	0.92	**0.87**	0.65	**0.64**	**1.03**	1.04
Average		0.28	**0.17**	0.53	**0.29**	0.80	**0.76**	1.08	**1.02**

Table 6.2: Results of our experiments with the NCLT dataset compared to Schaefer [153], averaged over ten localization runs per session. The variables Δ_{pos} and Δ_{ang} denote the mean absolute errors in position and heading, respectively, RMSE_{pos} and RMSE_{ang} represent the corresponding root mean squared errors, while f_{map} denotes the fraction of LiDAR scans per session used to build the reference map.

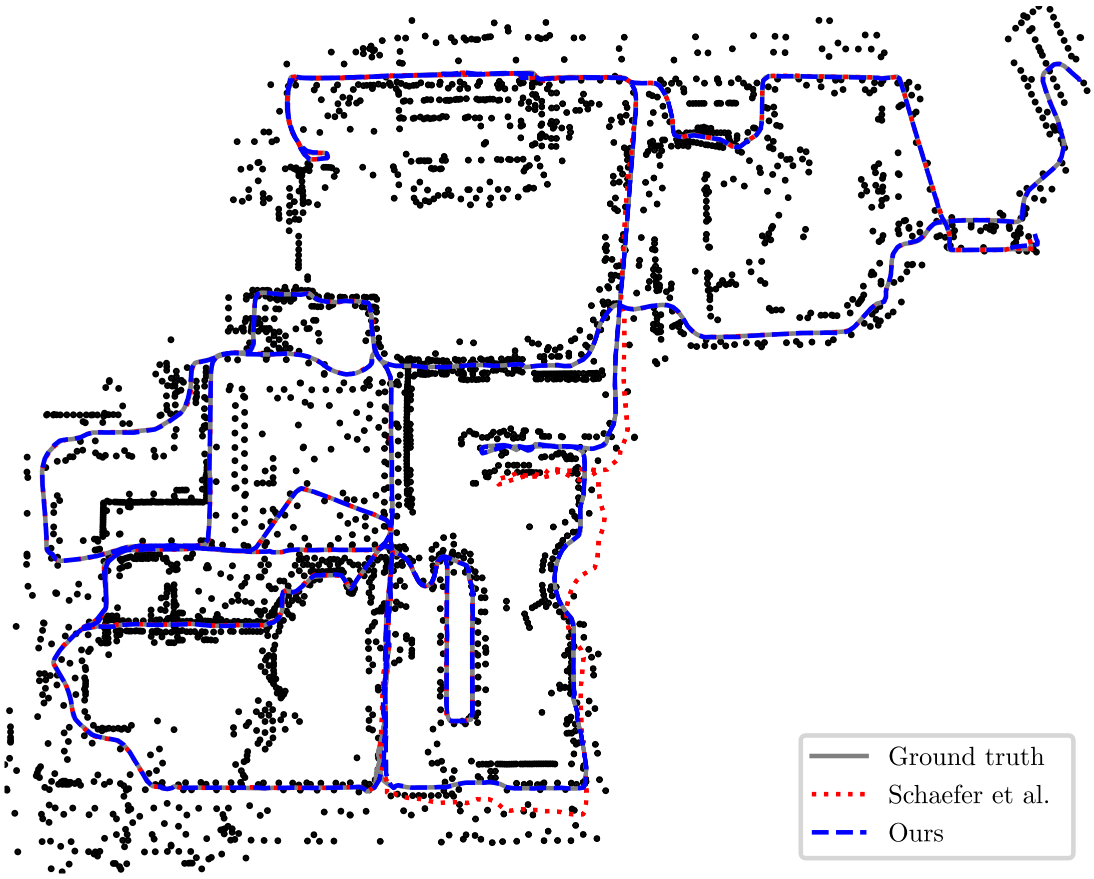


Figure 6.5: Comparison of localization results of Schaefer et al. [153] and our method in session 2012-02-23 on the NCLT dataset. The black dots are the poles on the map. The grey line is the ground truth trajectory. The blue line is our result and the red one is of the baseline method. As can be seen, Schaefer's method loses track of the robot in some places, while our method always tracks the correct robot poses with respect to the ground truth.

6.2.2.2 Localization on the MulRan Dataset

In the MulRan dataset, we use KAIST 02 sequence (collected on 2019-08-23) to build the global map and use KAIST 01 sequence (collected on 2019-06-20) for localization. We use the same setup as used in the experiments conducted on the NCLT dataset for mapping and localization. Table 6.3 shows the location and yaw angle RMSE errors on the MulRan dataset. As can be seen, our method consistently achieves a better performance than both baseline methods [36, 153].

6.2.3 Runtime

This experiment has been conducted to support the claim that our approach runs online at the sensor frame rate for pose tracking. For the initialization time cost by OverlapNet with 4000 particles is around 2 s, and for more details of time

Table 6.3: Localization results on MulRan dataset.

	Schaefer [153]	RangeMCL [36]	Ours
RMSE_{pos} [m]	1.82	0.83	**0.48**
RMSE_{ang} [deg]	0.56	3.14	**0.27**

cost by OverlapNet components, we refer to Chapter 4. In this section, we focus more on the online pose tracking runtime cost. We compare our method to the baseline method proposed by Schaefer et al. [153]. As reported in their paper, the baseline method takes an average of 1.33 s for pole extraction. Our method only needs 0.09 *s* for pole extraction and all MCL steps take less than 0.1 s yielding a run time faster than the LiDAR frame rate of 10 Hz.

6.3 Conclusion

Reliable and accurate localization is crucial for mobile autonomous systems. Pole-like objects, such as traffic signs, poles, lamps, etc., are ideal landmarks for localization in urban environments due to their local distinctiveness and long-term stability. In this paper, we present a heuristic, yet accurate, and fast pole extraction approach that runs online and has little computational demands such that this information can be used for a localization system. Our method performs all computations directly on range images generated from 3D LiDAR scans, which avoids processing 3D point cloud explicitly and enables fast pole extraction for each scan. We test the proposed pole extraction and localization approach on different datasets with different LiDAR scanners, routes, and seasonal changes. The experimental results show that our approach outperforms other state-of-the-art approaches, while running online for pose tracking. Besides, we release our pole dataset generated from the SemanticKITTI dataset to the public for evaluating the performance of pole extractors here: `https://github.com/PRBonn/pole-localization`.

Part III

Automatic Labeling for Task-Specific Semantic Segmentation

Chapter 7

Automatic Labeling for Moving Object Segmentation

As discussed in Part II, specific semantic categories in the scene can be more useful than others depending on the underlying application scenarios. For example, for LiDAR SLAM, distinguishing moving and non-moving objects can be more important than identifying different classes of objects, as shown in Chapter 5. In contrast, to localize precisely and reliably, pole-like objects can be used as good landmarks due to their local distinctiveness and long-term stability, as presented in Chapter 6.

With the advent of deep learning techniques, neural networks can provide accurate point-wise semantic predictions. Task-specific semantics can be addressed by semantic segmentation networks, such as using moving object segmentation, or in brief, MOS to distinguish moving and non-moving objects or using pole-like object segmentation to separate poles and non-pole backgrounds [33, 50, 190]. Despite the good performance that such deep neural networks can achieve, they rely on the diversity and amount of labeled training data that may be costly to obtain. With a large amount of publicly available LiDAR datasets nowadays [12, 25, 66], labeling LiDAR data is still a tedious process and one of the bottlenecks in supervised learning. Automatic label generation can alleviate this problem by exploiting the temporal-spatial dependence of the recorded sensor data. Especially for task-specific semantics, after specifying and simplifying the categories of semantics for specific tasks such as MOS and pole segmentation, we turn the challenging multiclass semantic segmentation problem into an easier point-wise binary classification task, which enables automatic label generation to be feasible.

In the final part of this thesis, we focus on methods that can automatically generate labels to train segmentation neural networks for LiDAR perception tasks. As shown in Figure 7.1, we use geometric-based methods to automatically gener-

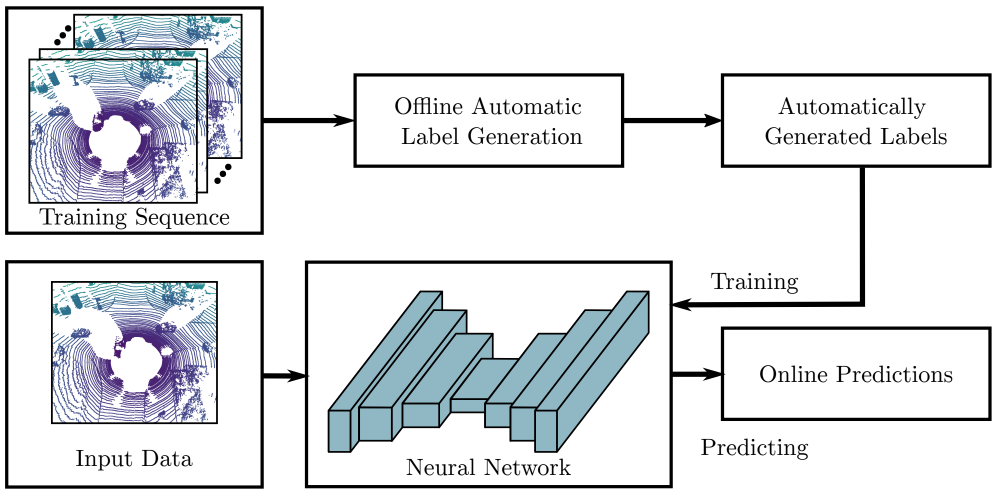


Figure 7.1: Pipeline overview of our proposed auto-labeling method. Our method generates training data offline without human annotations. It can leverage temporal-spatial information from sequences of LiDAR data. After training, the network can be used for online perception tasks of autonomous vehicles such as moving object segmentation and pole-like object detection.

ate training data offline, leveraging temporal-spatial information from sequences of LiDAR data and without human annotations. Based on our automatic labeling methods, we can generate a large number of training data on different datasets collected in different environments, which boosts the segmentation neural networks' performance and enables better generalization across different environments. After training the network, it can be deployed for online perception tasks of autonomous vehicles. We provide two examples of our automatic labeling methods for moving object segmentation in Chapter 7 and pole-like object segmentation in Chapter 8.

7.1 Automatic Labeling for LiDAR-MOS

Moving object segmentation is one of the techniques to separate the *actually moving* objects such as driving cars and pedestrians from static or non-moving objects such as buildings, parked cars, etc. This is an important processing step needed in many applications, such as predicting the future state of the surroundings [117], collision avoidance [134], or robot path planning [95]. This knowledge can also improve and robustify pose estimation, sensor data registration, and simultaneous localization and mapping [35]. Thus, accurate and reliable MOS available at frame rate is relevant for most autonomous mobile systems.

3D LiDAR-based moving object segmentation (LiDAR-MOS) is challenging due to the distance-dependent sparsity and uneven distribution of the range measurements. In Chapter 5, we propose a deep neural network, called LMNet, to achieve LiDAR-MOS faster than sensor frame rate by exploiting sequential LiDAR range images. Such supervised learning-based methods rely on often manually labeled data, which is often limited in size and cannot easily be generated for new or unseen environments. Automatic label generation can alleviate this problem by exploiting the temporal-spatial dependence of the recorded sequential data and, compared to online operation, that the data can be processed in batches. For example, the labels at a specific timestamp can be easier determined when exploiting preceding and succeeding scans compared to online MOS.

The main contribution of this chapter is a novel modularized approach to generate MOS labels in 3D LiDAR scans automatically. Our approach first exploits occupancy-based dynamic object removal techniques to detect possible dynamic objects coarsely. We then cluster the proposals into instances and track them using a Kalman filter. Based on the tracked trajectories, we label the actually moving objects (driving cars, pedestrians, etc.) as moving. In contrast, the non-moving objects, including parked cars, are labeled as static. Based on the labels automatically generated offline, we train the LiDAR-MOS network LMNet [33]. Note that no manually labeled data or other sensor information is needed for training. Once having a sequence of LiDAR scans, our method can automatically generate MOS labels.

For evaluation, we automatically label LiDAR scans in the training sequences of the KITTI odometry dataset [66] using different methods and compare the quality of generated MOS labels using SemanticKITTI [13, 33] ground truth LiDAR-MOS labels. Next, we automatically generate more labels on additional data from KITTI, i.e., the KITTI road dataset, to train LMNet further. Compared to the network trained on manually labeled ground truth data, the evaluation results suggest that the network trained on labels automatically generated by our method achieves similar performance, and is superior when using more automatically generated data from additional scans. Furthermore, our approach generalizes well to different, unseen environments, which we show for MOS on three different datasets.

In sum, we make the following claims for our approach: (i) Compared to existing methods, our approach generates better labels for LiDAR-MOS. (ii) Based on our generated labels, a network achieves similar performance compared to the same network trained with manual labels on the same data and better performance with additional automatically labeled training data. (iii) Our method generates effective labels for different LiDAR scanners and in different environments.

As shown in Figure 7.2, our approach consists of five serialized modules. Our proposed method only uses sequential LiDAR scans as input, and first uses a LiDAR odometry / SLAM approach to estimate the poses for each scan. With

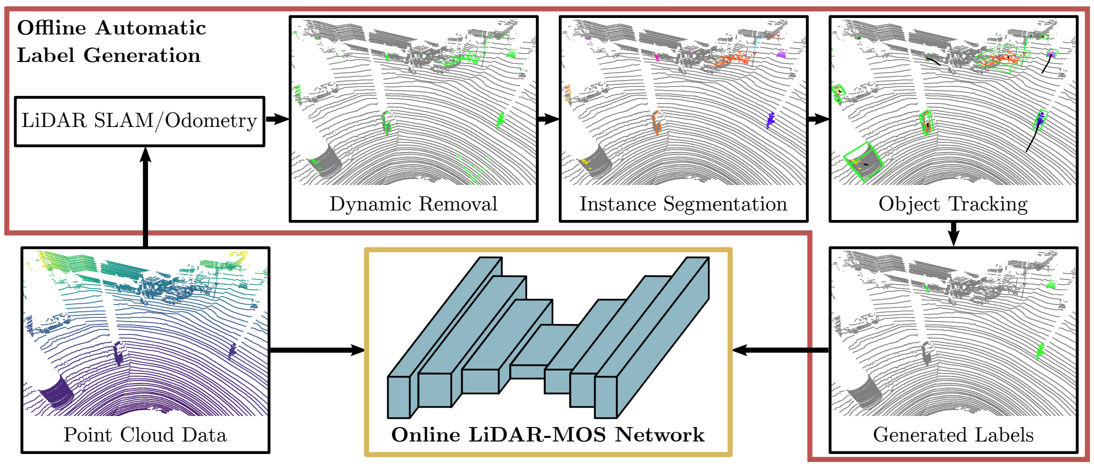


Figure 7.2: Overview of our method. It uses sequential LiDAR scans as input, and first conducts a LiDAR odometry / SLAM step to estimate the poses. With the estimated poses, it then applies a map cleaning method to coarsely detect the moving objects (colored in green). A clustering method is then used to extract instances (in different colors) based on the detected moving object proposals. After that, a multi-object tracking method is applied to associate instances (with bounding boxes) and decide the final labels of instances based on the tracked trajectories (colored in black). With the generated labels, we train LMNet that can be later deployed for online LiDAR-MOS.

the estimated poses, we then apply a map cleaning method to coarsely detect moving objects. We then use a clustering method to extract instances based on the detected moving object proposals. After that, we apply a multi-object tracking method to associate instances and determine the final labels of instances based on the tracked trajectories. Once we have generated the labels offline, we train LMNet that can be later deployed on an autonomous vehicle to perform LiDAR-MOS online in an unseen environment.

7.1.1 LiDAR Odometry

Instead of exploiting ground truth poses or information from other sensors, such as RTK-GPS, our method uses only sequential LiDAR scans as input. We use an off-the-shelf SLAM approach, SuMa [15] as introduced in Chapter 2.3, to estimate the pose of each LiDAR scan, but other systems might be used instead. SuMa exploits a spherical projection of the point cloud and estimates the relative pose between the current LiDAR scan and the maintained world model via projective ICP. In the following, we denote the estimated absolute pose of a scan at time t by $T_t \in \mathbb{R}^{4\times 4}$.

Note that there might be noise in the estimated poses, which may influence the performance of visibility or ray tracing-based methods [88, 135]. With increasing noise, more objects will be detected as moving due to the misalignment caused by inaccurate poses. However, this does not affect our method, since we use the estimated poses for map cleaning and generate only coarse dynamic object proposals, which are later verified and typical false positives are filtered out via tracking.

7.1.2 Coarse Dynamic Object Removal

Different from Chapter 3 using multiclass semantic information to pre-define movable objects, e.g., cars, pedestrians, here we only exploit the sequential temporal-spatial information to coarsely detect the moving objects in a class-agnostic manner. Since the sequential LiDAR data is available for offline label generation, we can detect dynamic objects with an aggregated map, which is also referred as map cleaning. Thus, this happens offline and *not* during the online operation. In this work, we use ERASOR [104], a state-of-the-art map cleaning approach. It first aggregates all local LiDAR points to obtain a single point cloud map $\mathcal{M}$:

$$\mathcal{M} = \bigcup_{t \in \mathcal{T}} \{ \boldsymbol{T}_t \boldsymbol{p} \mid \boldsymbol{p} \in \mathcal{P}_t \} \,, \tag{7.1}$$

where $\mathcal{T} = \{1, 2, \dots, N\}$ is the set of timestamps and N is the number of scans. $\mathcal{P}_t = \{\boldsymbol{p}_j\}_{j=1}^{N_t}$ is the scan at time t with N_t points. The transformations $\boldsymbol{T}_t$ are the aforementioned estimated poses from the SLAM method.

Let $\hat{\mathcal{M}}$ be the estimated static map, where the problem of map cleaning is defined as follows:

$$\hat{\mathcal{M}} = \mathcal{M} - \bigcup_{t \in \mathcal{T}} \hat{\mathcal{M}}_{\text{dyn},t} \,, \tag{7.2}$$

where $\hat{\mathcal{M}}_{\text{dyn},t}$ refers to the estimated dynamic points at timestamp t. In this work, we are interested in the set $\hat{\mathcal{M}}_{\text{dyn},t}$ of moving objects instead of the static map $\hat{\mathcal{M}}$.

ERASOR determines dynamic points by checking the discrepancy between transformed points $\mathcal{P}'_t = \{ \boldsymbol{T}_t \boldsymbol{p} \mid \boldsymbol{p} \in \mathcal{P}_t \}$ and $\mathcal{M}$, i.e., if an object in the map $\mathcal{M}$ can be found in the same position of $\mathcal{P}'_t$. Instead of checking the whole query scan $\mathcal{P}'_t$ at the same time t, it divides the volume of interest into small cells over azimuthal and radial directions. For each cell, it calculates the pseudo occupancy by subtracting the maximum and minimum height of the points inside the cell, and then labels the cell as dynamic if the ratio of pseudo occupancy between the corresponding pair of cells in the query $\mathcal{P}'_t$ and map $\mathcal{M}$ is larger than a threshold. In the end, it reverts the ground points as static in the labeled dynamic cells. For more details, we refer to the original paper [104].

Even though such map cleaning methods can distinguish moving objects from the static map, we observe a substantial number of false positive detections, possibly caused by noisy points or inaccuracies in the estimated poses. Since the goal of ERASOR is to obtain a static map, it is reasonable for map cleaning methods to be more aggressive and remove some points to guarantee a clean map. Moreover, exploiting both, past and future data sequentially, there are also redundant observations. Thus, it may not influence the mapping results when cleaning methods remove more points that belong to static objects.

However, the objective of MOS training data generation is different from that of map cleaning. Thus, we aim to accurately separate actual moving objects, e.g., driving cars, from static or non-moving objects, e.g., buildings, parked cars.

7.1.3 Class-Agnostic Instance Segmentation

To compute accurate segments of moving objects, we apply instance segmentation on the dynamic proposals $\hat{\mathcal{M}}_{\text{dyn},t}$ provided by the map cleaning method. The goal of a segmentation $\mathcal{G}$ is to partition the point cloud into disjoint subsets:

$$\mathcal{G} = \bigcup_{k \in \{1,\ldots,N_S\}} \mathcal{G}_k \,, \tag{7.3}$$

where $\mathcal{G}_k \subset \hat{\mathcal{M}}_{\text{dyn},t}$ is a segment, and N_S denotes the number of segments. Every point $\boldsymbol{p} \in \hat{\mathcal{M}}_{\text{dyn},t}$ exists in one and only one segment $\mathcal{G}_k$, i.e., $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \forall\, i \neq j$.

There are many methods for class-agnostic segmentation [16, 20, 27]. In this section, we choose HDBSCAN [27], which is an extension of DBSCAN [26]. DBSCAN is density-based and provides a clustering hierarchy from which a simplified tree of significant clusters can be constructed. HDBSCAN performs DBSCAN over varying density thresholds $\epsilon_{\text{density}}$ and integrates the result yielding a clustering that gives the best score over $\epsilon_{\text{density}}$. This allows HDBSCAN to find clusters of varying densities and to be more robust to parameter selection.

Once we obtain the final clustering results using HDBSCAN, we generate a bounding box $\boldsymbol{b}_k = (\boldsymbol{c}_k, \theta, l, w, h, s)^\top$ for each segment $\mathcal{G}_k$, including the center coordinates $\boldsymbol{c}_k \in \mathbb{R}^3$, the length l, width w, and height h, heading angle θ, and confidence score s. We filter out segments that are too small or too large, i.e., segments with less than $N_{\min}$ points or with a maximum side length of the bounding box larger than a threshold ϵ_{size} and get the final set of instances $\mathcal{B} = \{\boldsymbol{b}_m\}_{m=1}^{N_B}$ with $N_B \leq N_S$.

7.1.4 Multiple Dynamic Object Tracking

After clustering, there can still be static objects inside the set of instances $\mathcal{B}$. To verify the real dynamic objects, we use a multi-object tracking method proposed by Weng et al. [194] to obtain trajectories of object instances and determine the label for each tracked instance based on its movement.

To this end, we use multiple extended Kalman filters to track instance bounding boxes. For the motion model, we first compensate the ego-motion using the poses estimated by the SLAM system. Then, for each instance, we apply a constant velocity model as the state prediction. For the observation model, we find associations between instances in consecutive scans. We compute a cost matrix $\mathsf{C} \in \mathbb{R}^{N_B^t \times N_B^{t-1}}$ between all N_B^t instances in the LiDAR scan at timestamp t and all N_B^{t-1} instances still tracked in the previous scan at timestamp $t-1$, by means of the similarity. The association problem can be formulated as a bipartite graph matching problem that can be solved using the Hungarian method [93], an optimal assignment method to determine the pairs of associations between currently detected and previously tracked instances based on a fixed cost function.

To compute the instance similarity, we take three different geometric features into account, the center distance, overlapping bounding box volumes, and change of the volume between each pair of instances based on their bounding boxes. Each entry of the cost matrix C is calculated as the association cost by a linear combination of these three features between a new instance i and the prediction of a previous tracked instance j:

$$\mathsf{C}_{i,j} = \alpha_d\, c_d + \alpha_o\, c_o + \alpha_v\, c_v\,, \tag{7.4}$$

$$c_d = \|\boldsymbol{c}_i - \boldsymbol{c}_j\|_2\,, \tag{7.5}$$

$$c_o = 1 - \mathrm{IoU}(\boldsymbol{b}_i, \boldsymbol{b}_j)\,, \tag{7.6}$$

$$c_v = 1 - \frac{\min(\boldsymbol{v}_i, \boldsymbol{v}_j)}{\max(\boldsymbol{v}_i, \boldsymbol{v}_j)}\,, \tag{7.7}$$

where $c_d, c_o, c_v \in \mathbb{R}$ are the cost for the center distance, overlapping volume, and change of the volume, and $\alpha_d, \alpha_o, \alpha_v \in \mathbb{R}$ are corresponding importance weights. The center of the bounding box $\boldsymbol{b}_k$ is denoted as $\boldsymbol{c}_k$ and $\|\cdot\|_2$ is the Euclidean distance. The intersection over union $\mathrm{IoU}\,(\cdot)$ between two instance bounding boxes is used to account for the volume overlap with $v_k = l \times w \times h$ representing the volume of the bounding box $\boldsymbol{b}_k$.

There are multiple challenging cases during tracking. For example, newly detected objects might not be tracked in the previous scans, a tracked instance might leave the scene, or a tracked instance might be occluded or missed. To avoid wrong associations, we add a new EKF in case of newly detected objects by determining $\mathrm{Flag}_{\mathrm{add}}$:

$$\mathrm{Flag}_{\mathrm{add}} = (c_d > \epsilon_d) \vee (c_o > \epsilon_o) \vee (c_v > \epsilon_v)\,, \tag{7.8}$$

where $\epsilon_d, \epsilon_o, \epsilon_v$ are the thresholds for the three instance features respectively determined based on the validation data. We store the deactivated trajectories for a fixed number of timestamps n_{old}. We associate or re-identify instances if the similarity between a newly detected object and the still tracked or deactivated one is high, resulting in $\text{Flag}_{\text{add}} = \textit{false}$. For re-identification, the motion model is applied continuously also to the deactivated targets (not matched with any newly detected instance) to estimate their position in case of occlusions or missed detections.

After tracking, we label an instance as moving if its accumulated trajectory is larger than its maximum side length of the associated bounding box. We label the instance point-wise, which means all points inside the bounding box will be labeled as moving once we determine the instance as moving. In case of misdetection or occlusion, we can still generate temporally consistent labels by re-identifying instances using the EKFs.

7.1.5 Training a Neural Network for Online LiDAR-MOS

After running our pipeline, we obtain binary labels for all points. Based on that, we can train our LiDAR-MOS network LMNet [33] as introduced in Chapter 5, which can be later deployed for online LiDAR-MOS in unseen environments. LMNet exploits sequential range images from a LiDAR sensor as an intermediate representation combined with a typical encoder-decoder CNN and runs faster than the frame rate of the sensor. By using residual images, our network obtains temporal information and can differentiate between moving and static objects and achieves state-of-the-art performance on the LiDAR-MOS task.

Instead of using the manual labels provided by our LiDAR-MOS benchmark, we train our LMNet with the automatically generated labels by our proposed off-line pipeline. Since our method can generate labels automatically for sequential LiDAR data, we can train LMNet with more data than that provided by the LiDAR-MOS benchmark [33], which further boosts the performance of our network for online LiDAR-MOS. Furthermore, the proposed auto-labeling method can also generate labels for LiDAR data from other datasets, which were collected from different environments. Using these additional generated labels, our LMNet can better generalize in different environments.

7.1.6 Parameters and Implementation Details

Our method consists of individual modules that are carried out sequentially. We use SuMa [15] with the default parameters as the LiDAR SLAM approach. We then use ERASOR [104] with the default parameters to coarsely detect the dynamic objects. Based on the dynamic object proposals, we apply HDBSCAN [27] with multiple density thresholds $\epsilon_{\text{density}} = \{2.0, 1.0, 0.5, 0.25\}$ to generate dynamic object instances. After filtering out instances with less than $N_{\min} = 5$ points or with a maximum side length larger than $\epsilon_{\text{size}} = 20\,\text{m}$, we track the remaining instances using a multi-object tracking method [194]. We use the same importance weights for three different association terms, i.e., $\alpha_d = \alpha_o = \alpha_v = 1$, and keep track of deactivated instances for $n_{\text{old}} = 5$ timestamps. The thresholds are $\epsilon_d = 2\,\text{m}$, $\epsilon_o = 0.95$, and $\epsilon_v = 0.7$. For training LMNet [33], we use the default parameters with 8 residual images without semantic information and train for 150 epochs.

7.2 Experimental Evaluation

We present our experiments to illustrate the capabilities of our automatic label generation method for LiDAR-MOS. They furthermore support our key claims, that: (i) Our approach generates better labels for moving object segmentation using only 3D LiDAR scans; (ii) Based on our automatically generated labels, the network achieves similar performance in LiDAR-MOS compared to the one trained with manual labels, and better performance with additional automatically generated training data; (iii) Our method generates LiDAR-MOS labels for different LiDAR data collected from different environments.

7.2.1 Experimental Setup

In line with Chapter 5, we use the intersection-over-union (IoU) metric over moving objects to quantify the LiDAR-MOS performance, which is given by

$$\text{IoU}_{\text{MOS}} = \frac{\text{TD}}{\text{TD} + \text{FD} + \text{FS}}, \tag{7.9}$$

where TD, FD, and FS correspond to the number of true dynamics, false dynamics, and false statics points.

We evaluate our method on four different datasets. The first one is the LiDAR-MOS benchmark [33] of SemanticKITTI [12], which separates all classes into moving and static. It contains 22 sequences, from 00 to 07 and 09 to 10 for training with ground truth labels, 08 for validation, and 11 to 21 for testing. In this work, we first evaluate the automatic label generation methods on the training data in Section 7.2.2. After generating the labels automatically, we re-train the LMNet with the generated labels and test the re-trained model on the hidden test set. Since there is more unlabeled LiDAR data available in the original KITTI dataset [66], we generate more labels automatically and train the network with extra data. We also test the further trained model on the hidden test set, see Section 7.2.3. To evaluate the generalization ability of the proposed method, we also test our approach on three other datasets, Apollo [106], MulRan [90], and IPB-Car [30, 32], shown in Section 7.2.4. KITTI and Apollo have been

Table 7.1: Evaluation on LiDAR-MOS automatic label generation.

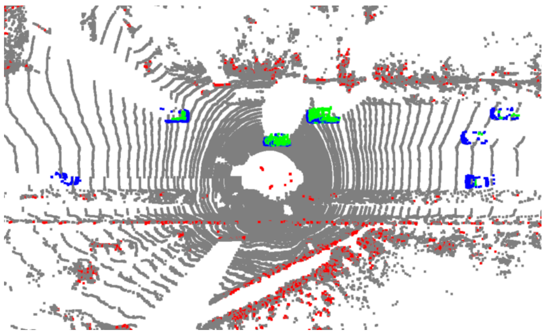
	IoU_{MOS}
Octomap-based [4]	13.6
Removert [88]	15.7
ERASOR [104]	19.1
Pfreundschuh's [135]	34.5
Yoon's [206]	15.1
Ours	**74.2**

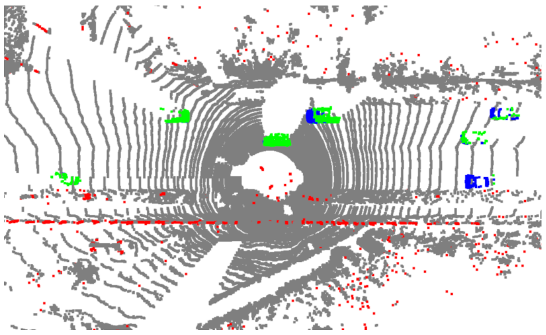
recorded with Velodyne HDL-64E LiDAR scanners, while IPB-Car and MulRan offer LiDAR data from Ouster sensors.

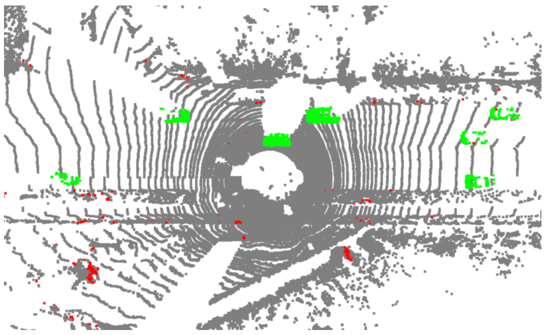
7.2.2 Automatic Data Labeling Results

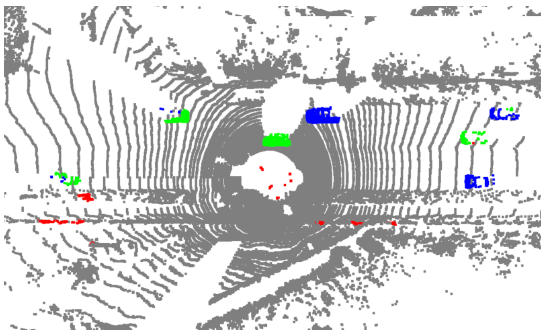
The experiment in this section supports our claim that our approach generates better labels for moving object segmentation using only 3D LiDAR scans than baseline methods. We test multiple methods that can distinguish moving and non-moving objects on 3D LiDAR data, as shown in Table 7.1. We compare our method to map cleaning-based methods like Removert [88], ERASOR [104], and an Octomap-based [4] method. Those methods are open-source and have been evaluated on the KITTI dataset. Therefore, we directly use the provided implementation with the default parameters. We also re-implement two other methods whose source code is not publicly available. The first one is proposed by Yoon et al. [206] and detects dynamic objects in LiDAR scans exploiting geometric heuristics, including residual and region growth, named Yoon's. The other one is the label generation method proposed by Pfreundschuh et al. [135], which uses range images for visibility checking and later uses clustering to verify candidates by voting, named Pfreundschuh's. We implement that method using the range image-based visibility checking part from Removert, the HDBSCAN for clustering, and the same voting as used in the original paper. Our method is denoted as Ours. All methods only use geometric information without any learning or semantic information, which makes it possible to generate labels fully unsupervised.

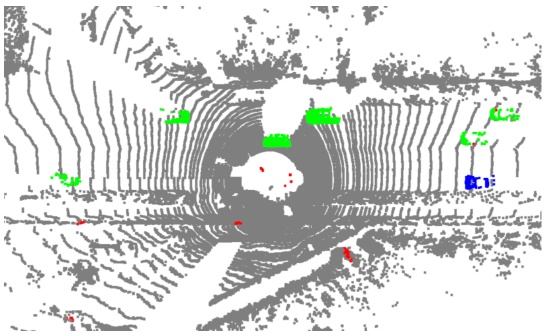
As shown in Table 7.1, our method outperforms other baseline methods on generating LiDAR-MOS labels in terms of IoU over moving objects, which is in line with the qualitative results shown in Figure 7.3. As can be seen, the map-cleaning-based methods, Octomap-based, Removert, and ERASOR, show many false dynamics (colored in red), while other label generation methods, Yoon's and Pfreundschuh's, often miss moving objects leading to many false static points (col-


(a) Octomap-based [4]


(b) Removert [88]


(c) Erasor [104]


(d) Yoon's [206]


(e) Pfreundschuh's [135]

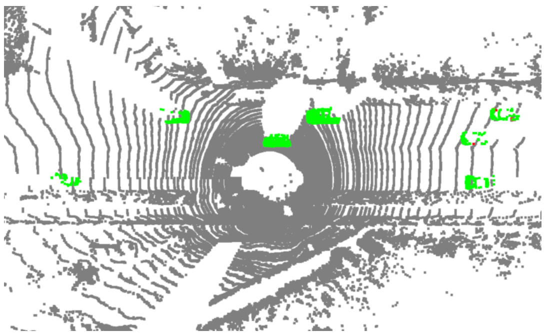

(f) Ours

Figure 7.3: Qualitative results of (a) Octomap-based [4], (b) Removert [88], (c) Erasor [104], (d) Yoon's [206], (e) Pfreundschuh's [135], and (f) ours. The green points represent the true dynamics (TD), red points are false dynamics (FD), blue points are false statics (FS), and the gray background are true statics (TS).

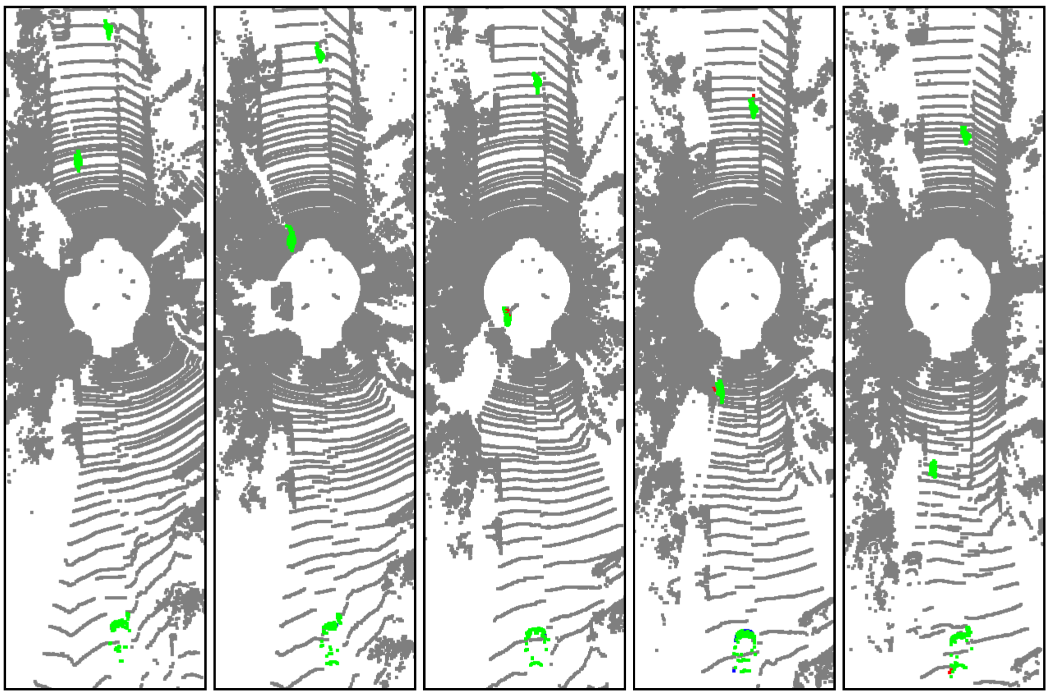


Figure 7.4: Qualitative results on consecutive scans. Our method detects different types of moving objects, i.e., vehicles and cyclists, consistently across a sequence of scans (green=TD, red=FD, blue=FS, gray=TS).

ored in blue). In comparison, our method detects moving objects more accurately.

We further provide qualitative results of our method on consecutive scans in Figure 7.4. As can be seen, our method detects different types of moving objects, i.e., vehicles and cyclists, consistently across a sequence of scans.

7.2.3 MOS Performance Using Auto-Generated Labels

In the second experiment, we re-train LMNet with the automatically generated labels and evaluate the model on the hidden test set of the LiDAR-MOS benchmark, as shown in Table 7.2. We compare the network trained on manual ground truth labels named LMNet+Manual, with the network re-trained on the labels generated by the proposed method named LMNet+Ours. As can be seen, LMNet+Ours achieves a similar IoU on the benchmark as LMNet+Manual. We also show the results of LMNet re-trained with extra automatically generated labels named LMNet+Ours+Extra. In this experiment, we use the road sequences of the KITTI raw dataset [66], generate additional labels using the proposed method, and re-train LMNet with labels automatically generated on both KITTI odometry and road data. Boosted with extra data, we see that LMNet+Ours+Extra outperforms LMNet+Manual trained with manual labels, and is the best performing strategy.

Table 7.2: Online LiDAR-MOS performance on benchmark.

	IoU_{MOS}
SalsaNext [38] (movable classes)	4.4
SceneFlow [105]	4.8
Spsequencenet [159]	43.2
KPConv [177]	60.9
LMNet+Manual	58.3
LMNet+Ours	54.3
LMNet+Ours+Extra	**62.3**

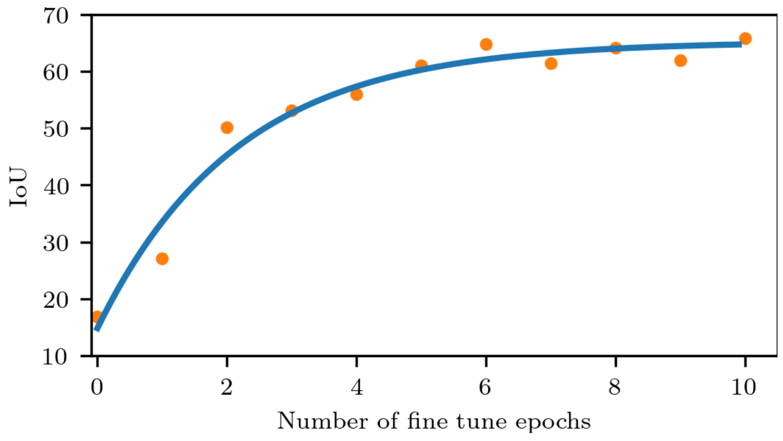
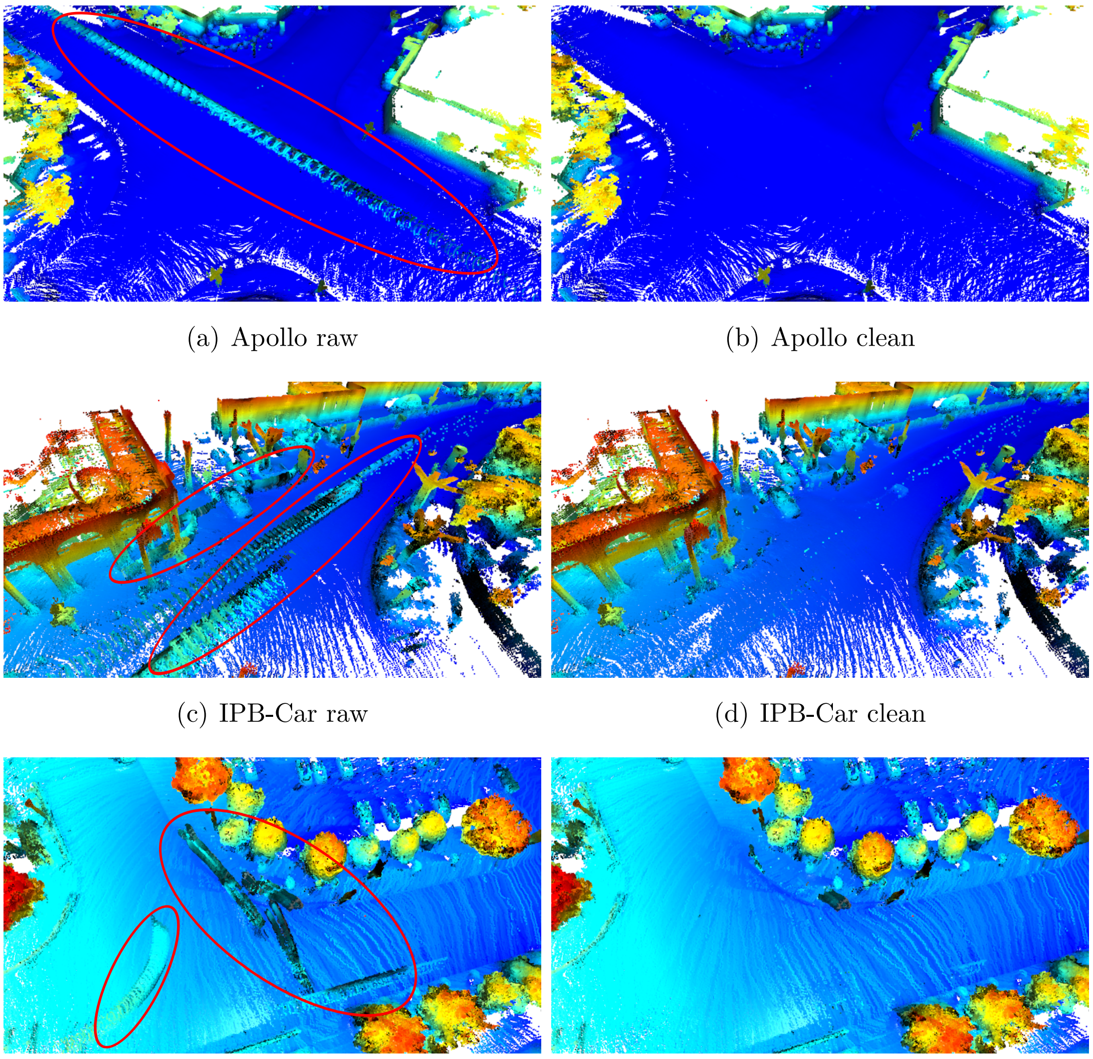


Figure 7.5: MOS performance vs. the number of epochs for fine-tuning on Apollo data.

We additionally compare our method to other online LiDAR-MOS methods that only use the current and past observations as required for real-world self-driving applications. For example, one uses all movable objects as dynamic depending on the semantics from a semantic segmentation network, such as SalsaNext [38], or multi-scan networks to learn to distinguish moving and non-moving object classes, such as Spsequencenet [159] or KPConv [177], or using the flow vectors with a threshold to determine the moving objects, such as SceneFlow [105]. Our method also outperforms all these baselines by a large margin.

7.2.4 Generalization Capabilities

The third experiment investigates the generalization capabilities of our approach, and it supports our last claim that our method can generate LiDAR-MOS labels for different LiDAR data collected from different environments. To this end, we test our method on three more datasets, Apollo [106], MulRan [90], and IPB-Car [30, 32]. To quantitatively evaluate the generalization capabilities of the proposed method, we manually label a small test set on the Apollo-ColumbiaPark-



(a) Apollo raw

(b) Apollo clean

(c) IPB-Car raw

(d) IPB-Car clean

(e) MulRan raw

(f) MulRan clean

Figure 7.6: Map cleaning results on three different datasets. The color from blue to red represents the height from low to high. Artifacts by moving objects are effectively removed by our method, as highlighted by red circles. The mapping results on three different datasets show that our method generalizes well in different environments.

Table 7.3: Online LiDAR-MOS performance on Apollo dataset.

	IoU_{MOS}
LMNet+Manual (trained on KITTI)	16.9
LMNet+Ours	45.7
LMNet+Ours+Fine-Tuned	**65.9**

MapData [106], sequence 2 frame 22300-24300 and sequence 3 frame 3100-3600, which contain many dynamic objects. We use the same LMNet and compare three different setups: the pre-trained model with SemanticKITTI LiDAR-MOS manual labels indicated by LMNet+Manual trained on KITTI, the re-trained model with automatically generated labels on Apollo-ColumbiaPark-MapData sequence 1 frame 6500-7500 and sequence 4 frame 1500-3100 indicated by LMNet+Ours, and the pre-trained model with automatically generated labels on KITTI fine-tuned with automatically generated labels on the Apollo dataset indicated by LMNet+Ours+Tuned.

As shown in Table 7.3, the model pre-trained with SemanticKITTI LiDAR-MOS manual labels cannot generalize very well to different environments. The re-trained model with our approach achieves a better performance, even when using only a small set of training data, in this case only 2600 LiDAR scans. If we fine-tune the pre-trained model with our automatically generated labels, i.e., using automatic generated labels from both KITTI and Apollo, it performs best with only a few fine-tuning epochs, as shown in Table 7.3 and Figure 7.5.

We also test our method on the MulRan and IPB-Car datasets. Since there are no manual labels available, we only show qualitative mapping results of Apollo, IPB-Car, and MulRan datasets in Figure 7.6. As can be seen, using the automatically generated labels, we can effectively remove dynamics during mapping shown in the upper row and obtain comparably clean maps shown in the bottom row.

7.3 Conclusion

In this chapter, we propose an automatic data labeling pipeline for 3D LiDAR data to save the extensive manual labeling effort and to improve the performance of existing learning-based MOS systems by automatically annotation training data. Our proposed approach achieves this by processing the data offline in batches, i.e., it is not designed to run online on a vehicle. It first uses a SLAM method to estimate the poses of a sequence of LiDAR scans. Second, It exploits an occupancy-based dynamic object removal to detect possible dynamic objects coarsely. Then, it extracts segments among the proposals and tracks them using a

Kalman filter. Based on the tracked trajectories, it labels the actually moving objects such as driving cars and pedestrians as moving. In contrast, the non-moving objects, e.g., parked cars, lamps, roads, or buildings, are labeled as static. We show that this approach allows us to label LiDAR data highly effectively and compare our results to those of other label generation methods. We also train a deep neural network with our automatically generated labels and achieve comparable performance to the one trained with manual labels on the same data—and an even better performance when using additional datasets with labels generated by our approach. Furthermore, we evaluate our method on multiple datasets using different sensors, and our experiments indicate that our method can generate labels in different outdoor environments.

Chapter 8

Automatic Labeling for Pole Segmentation

As discussed in Chapter 6, pole-like objects are useful landmarks for global localization. Multiple works have been done using poles for localization in different environments, such as in urban environments by Wilbers [42] and in forests by Pierzchala et al. [136]. This chapter applies the self-supervised learning scheme for pole segmentation, that is used for LiDAR-MOS automatic labeling introduced in the previous chapter illustrating the applicability of our self-supervised method for different binary semantic segmentation tasks. We adopt the automatic labeling-based learning scheme introduced in Chapter 7, but use the geometric-based pole extractor introduced in Chapter 6 to train a pole segmentation network. Instead of segmenting the environment into moving and non-moving parts, the proposed pole segmentation network only distinguishes the pole-like objects from others, which is very useful for localization.

Learning-based semantic segmentation networks [38, 101, 120] have achieved strong performance in segmenting and predicting different semantic classes of objects in the environments on LiDAR data. We can use such segmentation networks to online detect pole-like objects that can be used for localization. However, existing supervised networks generalize poorly to sensor data collected by different LiDAR scanners in unseen environments where they have not been specifically trained. To improve the generalization ability and at the same time exploit the good performance obtained by segmentation networks, we propose in this chapter to use our geometric-based pole extractor introduced in Chapter 6 to automatically generate pseudo pole labels from different environments for easily training an accurate and reliable pole segmentation network. We then integrate the pole segmentation network into our pole-based MCL method to improve the localization capabilities.

We evaluate our supervised learning-based method on three different datasets collected with varying types of LiDAR sensors. The evaluation results show that our supervised learning-based approach generalizes well in different environments and outperforms the geometric-based baseline methods in both localization accuracy and runtime.

8.1 Automatic Labeling for Pole Segmentation

We directly use our geometric-based pole extractor, as presented in Chapter 6, to automatically generate pseudo pole labels on LiDAR point clouds. Since our geometric-based pole extractor uses only range information, it generalizes well to different environments and different LiDAR sensors. It can generate a large amount of training data across different datasets. We then use such automatically generated pseudo pole labels to train a pole segmentation network. Trained and boosted with a large amount of such pseudo pole labels from different datasets, our learning-based method can generalize well in different environments and even outperforms the geometric-based method. As for the network architecture, we use the same SalsaNext [38], a range image-based semantic segmentation network, as used in the previous section.

The main contribution of this section is a novel learning-based pole segmentation method on LiDAR range images that can be used for long-term localization of autonomous mobile systems. In line with the previous chapters, we rely on the use of range images for pole extraction. The detected poles in the range image using only geometric information can further be used as pseudo pole labels to train a pole segmentation neural network. After training once with pseudo pole labels generated from different datasets, our learning-based based method can detect poles in different environments and achieve even better localization performance than our geometric-based method.

In sum, we make three key claims in this chapter: Our self-supervised pole extractor is able to (i) extract more reliable poles in the environment compared to the baseline method, as a result, (ii) achieve better online localization performance in different environments, and (iii) faster runtime compared to the geometric method.

8.1.1 Generate Pseudo Labels Using Geometric Pole Extractor

As shown in Chapter 7, geometric information can be used to automatically generate labels for training a LiDAR-based moving object segmentation network and achieve good performance in various environments. Such auto-labeling methods enable network learning in a self-supervised manner, which saves the extensive manual labeling effort and improves the generalization ability of the learning-based method. Similar to the idea of the previous chapter, we also use the poles detected by our geometric-based pole extractor to generate pseudo labels to train an online pole segmentation network.

As presented in Chapter 6, we can extract poles based on the range images exploiting purely geometric information. The intuition behind this pole extraction algorithm is that the range values of the poles are usually smaller than the backgrounds. We therefore first cluster points belonging to stand-alone objects that have different range values with respect to the background points. Then, we use multiple geometric shape-based checks to filter out non-pole objects and obtain pole-like object clusters.

Our geometric method only exploits the range information of the LiDAR scans, without exploiting any learning techniques, it generalizes well to different environments and different LiDAR sensors and can therefore generate a large amount of training data across different datasets, such as the NCLT dataset [28] with Velodyne HDL-32E sensor, SemanticKITTI dataset [13] with Velodyne HDL-64E sensor and MulRan dataset [90] with Ouster OS1-64 sensor. We use automatically generated pole labels from all these datasets to jointly train one segmentation network and apply it to unseen data.

8.1.2 Pole Segmentation Network Trained with Pseudo Labels

In this chapter, we do not design a new network architecture but reuse networks that have been successfully applied to LiDAR-based semantic segmentation. In line with our automatic labeling method for LiDAR-MOS, we rely on SalsaNext [38], an encoder-decoder architecture with a solid performance. After the segmentation, similar filtering steps as used in the geometric method are applied to remove outliers. SalsaNext network is comparably light-weight and can achieve real-time operation, i.e., run faster than the frame rate of the employed LiDAR sensor, which is commonly 10 Hz.

For training the segmentation network, we directly feed them with the range images plus the pseudo pole labels generated from our geometric-based pole extractor. We use the same loss functions as used in the original segmentation methods, while mapping all classes into two per-point classes, poles and non-poles. We retrain the network and evaluate the pole extracting performance with our pole datasets and also localization tasks. Figure 8.1 shows the training pipeline of our proposed learning-based pole segmentation method. Note that, we train the network with pseudo pole labels generated from different datasets, and later use the same model to extract poles in different environments.

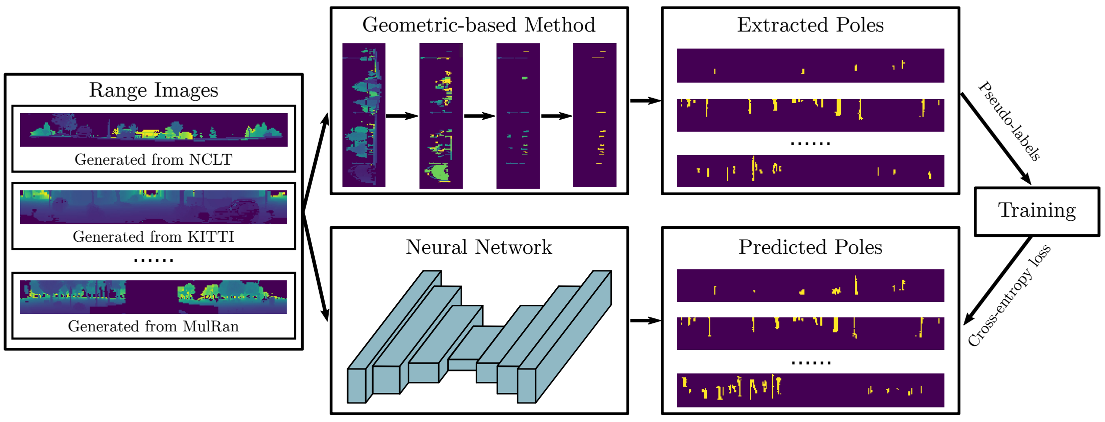


Figure 8.1: Overview of our learning-based pole extraction approach. We first use the geometric method to generate range images and pseudo pole labels on multiple datasets, including NCLT, KITTI, and MulRan. Then, we train a deep neural network to extract pole-like objects directly on the range images. Our learning-based pole extractor generalizes well to all three datasets with a single model.

8.1.3 Pole Segmentation-Based LiDAR Localization

For global localization, we use the same setup as used in Chapter 6. Here, we only present the difference between our learning-based approach and the previous geometric-based approach and refer the full localization pipeline to Chapter 6.

We use the poles extracted from LiDAR range images of mapping sequences to build the pole-based map for both methods. For the learning-based method proposed in this chapter, the same poles used for mapping are also used for training the pole segmentation network. Both mapping and training are conducted offline. We treat the poles stored in the historical maps generated from different datasets as a large training data pool. Therefore, we can exploit a large amount of training data automatically generated by our geometric-based pole extractor to boost the generalization ability and accuracy of our learning-based pole segmentation network.

During the online localization phase, we also stick to the setup as used in Chapter 6. We first use OverlapNet to provide the initial location hypothesis and then sample particles around each location candidate. Different from the previous geometric-based method, in this chapter, we match the poles detected by the online pole segmentation network rather than the geometric pole extractor with those stored in the global map to update the importance weights of the particles. We integrate this pole segmentation-based observation model into MCL to achieve accurate global localization.

8.2 Experimental Evaluation

8.2.1 Evaluation of Auto-Labeling for Pole Segmentation

We present the following experiments to illustrate the capabilities of our automatic label generation method for pole segmentation. The experiments furthermore support our key claims that our auto-labeling method is able to (i) extract more reliable poles in the environment compared to the baseline method, as a result, (ii) achieve better online localization performance in different environments, and (iii) faster runtime compared to the geometric method.

8.2.2 Datasets for Pole Extraction and LiDAR Localization

Following Chapter 6, we use our manually labeled poles in session 2012-01-08 of the NCLT dataset to evaluate different pole extraction methods. Besides our own labeled data, we also reorganize the SemanticKITTI [13] dataset sequence 00-10 by extracting the pole-like objects like traffic signs, poles, and trunk, and then clustering the point clouds to generate the ground truth pole instances.

To assess the localization reliability and accuracy of our method, we use the NCLT dataset [28] and MulRan dataset [90]. These two datasets are collected in different environments (U.S., Korea) with different LiDAR sensors (Velodyne HDL-32E, Ouster OS1-64). In these two datasets, the robot passes through the same place multiple times with month-level temporal gaps, hence ideal to test the long-term localization performance. We compare our methods to both a pole-based method proposed by Schaefer et al. [153] and our range image-based method [36], RangeMCL. For the SemanticKITTI dataset, there is no overlap area between different sequences for evaluating long-term localization. Therefore, we only used the extracted pole labels from the SemanticKITTI dataset to train our network. Note that, the SemanticKITTI dataset was collected in Germany with a Velodyne HDL-64E LiDAR scanner.

We stick to the same setup as used in Chapter 6 for all experiments to ensure fair comparisons.

8.2.3 Pole Extractor Performance

The first experiment evaluates the pole extraction performance of our approach and supports the claim that our range image-based method outperforms the baseline method in pole extraction.

We evaluate both our geometric-based pole extractor introduced in Chapter 6, named Ours-G, and our learning-based pole segmentation method proposed in this chapter, named Ours-L. For training the pole segmentation network, we use data from multiple datasets, including the session 2012-01-08 in the NCLT dataset, sequence KAIST 02 in the MulRan dataset and sequence 00-02, 05-09 in the SemanticKITTI dataset. For validation, we use sequences 03 and 04 in the SemanticKITTI dataset and sequence 10 for testing. We train the network for 150 epochs using stochastic gradient descent with an initial learning rate of 0.01 which is decayed by 0.01 after each epoch. The batch size is 12 and the spatial dropout probability is 0.2. The size of the range image is 32×256 and the valid range values are normalized between 0 and 1. To prevent overfitting, we augmented the data by applying a random rotation or translation, flipping randomly around the y-axis with a probability of 0.5. During the matching phase, we find the matches via nearest-neighbor search using a k-d tree with 1 m distance bounds.

Table 8.1 summarizes the precision, recall, and F1 score of our method and Schaefer et al. [153] with respect to the ground truth pole map on both the NCLT dataset and SemanticKITTI dataset. As can be seen, our methods achieve better performance and extract more poles in both environments compared to the baseline method. Compared to our geometric-based pole extractor, our learning-based method finds more poles while introducing more false positives, which decreases precision. This can also be seen in Figure 8.2, which shows pole extraction examples of our geometric and learning-based pole extractor.

Note that we trained our pole segmentation network only once with pseudo pole labels generated from different datasets and evaluated it on multiple different datasets. As can be seen in Figure 8.2, the environments of different datasets vary a lot, while our learning-based method can still extract poles well without fine-tuning, which shows a good generalization ability of our method. The possible reason for that is the range values of the poles are usually significantly different than the backgrounds, which makes poles distinctive and easy to be detected on range images. Compared to multiclass segmentation, it is easier for the neural network to learn a more general model to detect poles based on the range images.

8.2.4 Localization on the NCLT Dataset

The second experiment is presented to support the claim that our approach achieves higher accuracy on localization in different environments. For all the experiments, we use the same setup as used in the baselines and report their results from the original work.

We use the same setup as used in Chapter 6 on the NCLT dataset for localization. To briefly summarize, we first build the map using the data from the first session and also the scans of later sessions from the unseen places. We use our OverlapNet to provide the top 20 similar place hypotheses and sample 200

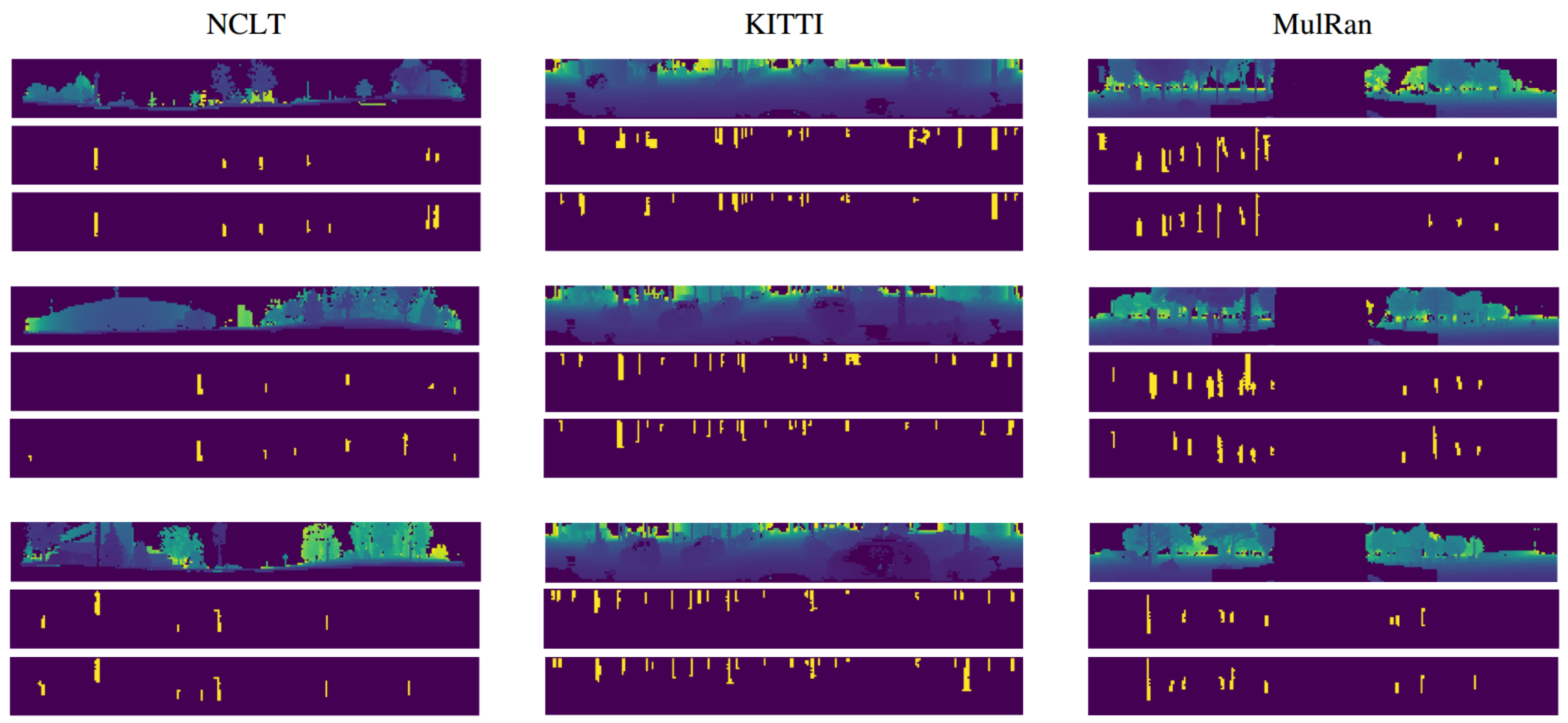


Figure 8.2: Pole extraction examples using our geometric and learning-based pole extractor. For each sample, the first figure is the range image, the second one is the pole extraction results by our geometric method, and the last one shows the pole extraction results by our learning-based method. Note that, for the learning-based method we get all the results on three datasets with a single trained model.

Table 8.1: Pole extraction accuracy on NCLT and KITTI datasets. Ours-G refers to our previous geometric-based method, and Ours-L refers to our proposed learning-based pole extractor.

Dataset	Method	Precision	Recall	F1 Score
NCLT	Schaefer [153]	0.69	0.39	0.50
	Ours-G	**0.77**	0.66	**0.71**
	Ours-L	0.68	**0.67**	0.67
Semantic KITTI	Schaefer [153]	0.62	0.38	0.46
	Ours-G	**0.69**	0.44	0.52
	Ours-L	0.61	**0.58**	**0.59**

particles around each place hypothesis within a 2.5 m circle. The orientations are uniformly sampled from −5 to 5 deg. We resample particles if the number of effective particles is less than 0.5. To get the pose estimation, we use the average poses of the best 10% of the particles.

Table 8.2 shows the position and orientation errors for every session. We run the localization 10 times and compute the average means and RMSEs to the ground truth trajectory. The results show that both our geometric and learning-based methods surpass Schaefer et al. [153] in almost all sessions with an average error of 0.17 m and 0.16 m respectively. Besides, in session 2013-02-23, the baseline method fails to localize resulting in an error of 2.47 m, while our method never loses track of the robot position (Figure 6.5). This is because our pole extractor can robustly extract poles even in an environment where there are fewer poles. Schaefer et al. [153] analyze their localization failure in session 2013-02-23 for the reason that the barrels in a construction area are moved a few meters to the right in the later session. As these barrels are detected as poles by their approach, they are built in the map and cause the wrong pole matching to the map in this area. In our pole extraction algorithms, we discard those poles with too large radiuses. Thus, the barrels are not a part of our map and our localization will not be influenced by the movement of these barrels. Interestingly, our learning-based method improves localization results more than our geometric-based method in most sessions. It may be caused by a more general pole segmentation model trained with pseudo labels generated from different environments.

8.2.5 Localization on the MulRan Dataset

To further show the generalization ability of our method, we test both our geometric and learning-based methods on the MulRan dataset, which was collected from a different type of LiDAR sensor in a different environment. We use the

Session	Δ_{pos} [m]	Date	RMSE_{pos} [m]		Δ_{ang} [deg]		RMSE_{ang} [deg]	
	Ours-G	Ours-L	Ours-G	Ours-L	Ours-G	Ours-L	Ours-G	Ours-L
2012-01-08	**0.12**	**0.12**	0.15	**0.14**	**0.63**	**0.63**	0.81	**0.80**
2012-01-15	**0.15**	**0.15**	**0.20**	**0.20**	**0.75**	**0.75**	**0.98**	0.99
2012-01-22	**0.15**	**0.15**	0.19	**0.18**	**0.91**	**0.91**	**1.24**	**1.24**
2012-02-02	0.14	**0.13**	0.17	**0.16**	**0.70**	**0.70**	0.92	**0.91**
2012-02-04	0.14	**0.13**	0.17	**0.16**	**0.67**	**0.67**	0.88	**0.86**
2012-02-05	**0.14**	0.15	**0.21**	0.25	**0.70**	**0.70**	**0.94**	0.95
2012-02-12	0.25	**0.24**	**1.00**	**1.00**	0.79	**0.78**	1.02	**0.99**
2012-02-18	**0.13**	**0.13**	0.18	**0.16**	**0.68**	**0.68**	**0.91**	**0.91**
2012-02-19	0.14	**0.13**	0.18	**0.17**	**0.69**	0.71	**0.92**	0.94
2012-03-17	0.14	**0.13**	0.17	**0.16**	**0.80**	**0.80**	1.03	**1.02**
2012-03-25	0.18	**0.17**	0.24	**0.22**	1.38	**1.37**	1.79	**1.77**
2012-03-31	0.14	**0.13**	0.18	**0.16**	**0.73**	**0.73**	0.94	**0.93**
2012-04-29	**0.15**	**0.15**	**0.22**	0.23	**0.82**	**0.82**	**1.07**	**1.07**
2012-05-11	**0.13**	**0.13**	0.16	**0.15**	**0.75**	0.77	**0.97**	0.99
2012-05-26	**0.14**	**0.14**	0.18	**0.16**	**0.67**	0.68	**0.87**	**0.87**
2012-06-15	**0.15**	**0.15**	0.19	**0.18**	0.65	**0.63**	0.87	**0.84**
2012-08-04	0.17	**0.16**	0.23	**0.19**	**0.84**	0.85	**1.09**	**1.09**
2012-08-20	0.16	**0.15**	0.21	**0.18**	**0.69**	**0.69**	**0.91**	**0.91**
2012-09-28	0.17	**0.15**	0.24	**0.19**	0.73	**0.71**	0.95	**0.93**
2012-10-28	0.19	**0.17**	0.28	**0.23**	0.68	**0.68**	0.91	**0.90**
2012-11-04	0.21	**0.18**	0.32	**0.23**	0.72	**0.70**	0.97	**0.93**
2012-11-16	0.30	**0.25**	0.44	**0.37**	1.40	**1.38**	1.92	**1.90**
2012-11-17	0.20	**0.17**	0.32	**0.22**	0.69	**0.68**	0.95	**0.91**
2012-12-01	0.23	**0.21**	**0.43**	0.44	0.67	**0.65**	0.89	**0.85**
2013-01-10	0.19	**0.16**	0.23	**0.19**	**0.63**	0.64	**0.81**	0.82
2013-02-23	0.24	**0.21**	0.57	**0.49**	**0.59**	**0.59**	**0.85**	**0.85**
2013-04-05	0.30	**0.27**	0.87	**0.82**	**0.64**	**0.64**	1.04	**1.02**
Average	0.17	**0.16**	0.29	**0.27**	**0.76**	**0.76**	1.02	**1.01**

Table 8.2: Results of our experiments with the NCLT dataset compared to Schaefer [153], averaged over ten localization runs per session. The variables Δ_{pos} and Δ_{ang} denote the mean absolute errors in position and heading, respectively, RMSE_{pos} and RMSE_{ang} represent the corresponding root mean squared errors.

Table 8.3: Localization results on MulRan dataset.

	Schaefer [153]	RangeMCL [36]	Ours-G	Ours-L
RMSE_{pos} [m]	1.82	0.83	**0.48**	0.49
RMSE_{ang} [deg]	0.56	3.14	**0.27**	0.28

Table 8.4: Pole extracion runtime results.

	NCLT	KITTI	MulRan
Ours-G	12 Hz	2 Hz	4 Hz
Ours-L	17 Hz	16 Hz	16 Hz

MulRan dataset KAIST 02 sequence (collected on 2019-08-23) to build the global map and use KAIST 01 sequence (collected on 2019-06-20) for localization. Table 8.3 shows the location and yaw angle RMSE errors on MulRan Dataset. As can be seen, our geometric and learning-based methods consistently achieve a better performance than both baseline methods [36, 153]. Note that, we train our pole segmentation only once, and there is no fine-tuning when applying it to a new environment.

8.2.6 Runtime

This experiment has been conducted to support the claim that our approach runs online at the sensor frame rate. As shown in Table 8.4, we compare our method to the baseline method proposed by Schaefer et al. [153] on three different datasets, including NCLT (session 2012-01-08), KITTI (sequence 09), and MulRan (KAIST 02) datasets. As reported in their paper, on the NCLT dataset the baseline method takes an average of 1.33 s for pole extraction on a PC using a GPU. We tested our geometric method without using a GPU and our method only needs 0.09 s for pole extraction and all MCL steps take less than 0.1 s yielding a run time faster than the LiDAR frame rate of 10 Hz.

The performance of geometric-based pole extractors, both Schaefer's and ours, is influenced by the size of the input data, and it is a trade-off between localization accuracy and speed. To achieve good localization results for the geometric method, we use the range image size of 32×256 for NCLT and 64×500 for KITTI and MulRan, which leads to a decrease in the runtime performance. However, our learning-based method is not influenced by the size of input data. In our case, we fix the size of network input as 32×256, and our network always works online with good localization performance with a single GPU, which shows a significant advantage of our learning-based method.

8.3 Conclusion

In this chapter, we present a novel self-supervised pole segmentation approach on LiDAR data that runs online and can be used in different environments. We use the extracted poles as pseudo labels to train a deep neural network for online range image-based pole segmentation. After boosted with pseudo pole labels extracted from multiple datasets collected in different environments, our learning-based method can run across different datasets and achieve even better localization results compared to our geometric-based method. We further test our learning-based pole extraction methods for localization on different datasets with different LiDAR scanners, routes, and seasonal changes. The experimental results show that our methods outperform other baseline approaches.

In the final part of this thesis, we propose an automatic data labeling pipeline for 3D LiDAR data to save the extensive manual labeling effort. Our method exploits only geometric information together with the temporal-spatial dependence of the dataset to automatically generate labels for training segmentation networks. We test our auto-labeling pipeline for both LiDAR-based moving object segmentation and pole-like object segmentation and improve the performance of existing learning-based MOS and pole segmentation systems. Our proposed approach achieves this by processing the data offline in batches and generating labels on different datasets collected from different environments. We show that our auto-labeling method allows us to label LiDAR data highly effectively and compare our results to those of other label generation methods. We also train deep neural networks with our auto-generated labels and achieve similar performance compared to the one trained with manual labels on the same data—and an even better performance when using additional datasets with labels generated by our approach. Furthermore, we evaluate our automatic labeling methods for both LiDAR-MOS and pole segmentation on multiple datasets using different sensors and our experiments indicate that our methods can generate labels in diverse environments, consequently training networks with strong generalization ability.

Chapter 9

Related Work

A large number of research activities have been conducted in the field of robotics and autonomous vehicles. Perception tasks, such as SLAM and localization, are among the longest-running research areas, as they are critical components for most autonomous mobile systems. Furthermore, many textbooks [8, 163, 178] and surveys [6, 7, 24] focusing on SLAM and localization have been published over the past decades. Recently, with the advent of deep learning and neural networks, especially in the computer vision context, the fast growth of exploiting semantics in autonomous vehicles has been driven. Multiple surveys [64, 85, 92] review the possible ways of using semantic information to improve the capabilities of autonomous mobile systems. Especially in the perception area, semantics have attracted increasing attention for tasks such as SLAM and localization [56, 83, 167].

In this chapter, we discuss LiDAR-based methods for perception tasks of autonomous vehicles and the advantages of our approach compared to existing methods related to the developments in this thesis. We review both classical geometric-based approaches and newly semantic-based methods that exploit a high-level understanding of the environment. Instead of presenting how to obtain such semantic information, e.g., semantic segmentation, object detection, etc., in this chapter, we focus more on the approaches about how to use such semantics for perception tasks of autonomous vehicles, e.g., SLAM and localization. We divide this chapter into four sections in which we deal with individual aspects of the structure of this thesis. In Section 9.1, we present different works that achieve LiDAR-based SLAM, including traditional geometric-based methods and recent semantic-based methods. In Section 9.2, we discuss LiDAR-based loop closing works used to eliminate the drift of pose estimation during SLAM. Section 9.3 discusses approaches that localize robots in given maps. Finally, Section 9.4 deals with related works that automatically generate training data for LiDAR perception tasks, thus reducing the effort of the laborious labeling of the data.

9.1 LiDAR-Based SLAM

LiDAR-based odometry estimation and simultaneous localization and mapping or SLAM are classical topics in robotics and autonomous vehicles with a large body of scientific work. Traditional LiDAR-based odometry and mapping systems often reduce the 3D point cloud data by relying on features [211], subsampled clouds [124, 183], or voxels [211] as well as normal distributions transform (NDT) based map representations [43, 150, 166]. For example, the LiDAR odometry and mapping (LOAM) by Zhang and Singh [211, 212] 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 and frame-to-model registration. As a follow-up, LeGO-LOAM by Shan et al. [157] conducts ground segmentation and isotropic edge points extraction from the projected range image. Then a two-step non-linear optimization is executed on ground and edge correspondences to solve two sets of transformation coefficients successively.

Unlike the above-mentioned methods using only key points, the approach by Moosmann and Stiller [124] directly uses range images generated from a LiDAR scanner 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. In contrast to the grid-based map representation, Park et al. [131] proposed a surfel-based mapping approach based on ElasticFusion by Whelan et al. [195] that allows optimization of continuous-time trajectories for a rotating 2D laser scanner. SuMa by Behley and Stachniss [15] also exploits surfels and operates on all LiDAR points and performs registration to a surfel map at every step of the algorithm. Instead of relying on nearest neighbor search, SuMa uses projective data association to find correspondences between the current scan and rendered model views, which is computationally efficient. Unlike other approaches, it performs frame-to-model tracking for every incoming scan and updates the map representation at the same time. We build our semantic LiDAR-SLAM presented in Chapter 3 based on SuMa due to its high efficiency and accuracy.

Recently, motivated by the advances of deep learning and convolutional neural network for scene understanding, many semantic SLAM techniques have been proposed exploiting semantic information from different types of sensors, including cameras [3, 22, 74, 97, 111, 115, 175], cameras + IMU data [21], stereo cameras [63, 100, 103, 184, 202], or RGB-D sensors [18, 112, 113, 147, 152, 169, 207]. Most of these approaches were applied indoors or in specific scenarios such as fields and forests using either an object detector or semantic segmentation on the camera images. For example, Mercier et al. [115] propose a deep network to localize target object instances in 2D RGB images. They train the network with synthetic data and directly apply it to real-world images demonstrating good applicability for indoor robotics and augmented reality applications. McCool et al. [111] propose a lightweight neural network to improve the speed and accuracy of agriculture robots in detecting weeds and crops in the fields. Based on the same agricultural robotics platform, Hall et al. [74] introduce a clustering method on such network features to achieve an unsupervised weed scouting for later weed localization and destruction. Unlike localizing weeds in fields, Robert et al. [143] focus on identifying tree bark that can be used for robot navigation and re-localization in forest environments. They propose a deep learning-based tree bark identification network that outperforms the hand-crafted descriptors for bark re-identification.

To achieve outdoor large-scale semantic SLAM, one can combine 3D LiDAR sensors with RGB cameras. For example, Yan et al. [201] combine 2D images and 3D points to detect cars and pedestrians. They exploit heuristics to first coarsely segment and track objects in sequential LiDAR data. By projecting LiDAR segments into images, they apply shape and color priors to distinguish different objects. Finally, they remove the detected object to generate clean maps of the environments. Wang and Kim [188] use images and 3D LiDAR point clouds from the KITTI dataset [67] collected by a car to jointly estimate road layout and segment urban scenes semantically. They use such semantic information to provide location priors for navigation. Jeong et al. [81, 82] also propose a multi-modal sensor-based semantic 3D mapping system which improves the segmentation results as well as the mapping results. To achieve reliable SLAM results in forests, Pierzchała et al. [136] use a 3D LiDAR together with a stereo camera and an IMU to detect the tree stems and use such information to improve robot localization and mapping results. The work by Parkison et al. [132] develops a point cloud registration algorithm by directly incorporating image-based semantic information into the estimation of the relative transformation between two point clouds. By adding the semantic constraints, their method improves the LiDAR poes estimates. A subsequent work by Zaganidis et al. [208] combines LiDAR points and images to achieve semantic 3D point cloud registration. It uses semantic information to improve pose estimation but cannot be used for online operation because of the long processing time. All these approaches focus on combining 3D LiDAR with other sensors, especially cameras, to improve object detection or semantic segmentation for localization or mapping.

In contrast to the approaches mentioned above, our proposed semantic LiDAR SLAM only uses LiDAR data. There are not many LiDAR-only semantic SLAM methods published yet. The most similar approaches to our proposed method presented in Chapter 3 are by Sun et al. [173] and Dubé et al. [53], which also realize semantic SLAM using only a single LiDAR sensor. Sun et al. [173] present a semantic mapping approach, which is formulated as a sequence-to-sequence encoding-decoding problem. They represent and maintain a 3D map as a Recurrent-OctoMap, where each cell is modeled as a recurrent neural network. Although the proposed Recurrent-OctoMap achieves good semantic mapping results, especially for long-term consistently mapping, it is time-consuming and can not operate online for autonomous driving. On the contrary, Dubé et al. [53, 54] propose an approach called SegMap, which builds a coarse semantic map and localizes the car within the semantic map online. SegMap extracts segments from the point cloud and assigns semantic labels to them. It exploits learning-based descriptors of segments for extracting semantic information that can be later used for the global retrieval and multi-robot collaborative SLAM. Despite achieving good localization results, SegMap only generates coarse semantic maps with segments of cars, buildings, and others.

Different from Recurrent-OctoMap and SegMap, our proposed semantic LiDAR SLAM, SuMa++ presented in Chapter 3, operates online and builds dense semantic maps with an abundance of semantic classes. Our approach uses LiDAR data and exploits information from a semantic segmentation operating on range images generated from LiDAR scans. Using these semantics, SuMa++ filters outliers caused by dynamic objects, like moving vehicles and humans, to improve both mapping and odometry accuracy.

Although our semantic LiDAR SLAM successfully exploits semantic information to achieve more accurate pose estimation and improve the mapping results, it relies on multiclass semantic segmentation [38, 101, 120]. However, the performance of these multiclass semantic segmentation networks strongly depends on the diversity and amount of labeled training data that may be costly to obtain. To alleviate this, in Chapter 5, we propose LiDAR-based moving object segmentation (LiDAR-MOS) [33]. Instead of detecting multiple semantic classes such as vehicles, humans, buildings, etc., we aim at separating the *actually moving* objects such as driving cars from static or non-moving objects such as buildings, parked cars, etc. Such binary labels are easier to obtain than multiclass semantics. For example, we use automatic labeling techniques as described in Chapter 7 to automatically generate training data for LiDAR-MOS without human annotation effort. With simplified binary labels, we retrain the same semantic segmentation networks [38, 101, 120]. During online operation, we use the MOS predictions estimated by the retrained network to mask out the dynamic objects in the input LiDAR data of the same geometric LiDAR SLAM system, SuMa [15]. As the experimental results presented in Chapter 5, by applying our MOS predictions as a preprocessing mask, the odometry results are even better than our multiclass semantic-enhanced SLAM, SuMa++ [35].

9.2 Loop Closing for LiDAR 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. A correct loop closing helps the SLAM system to reduce the accumulated drift of the pose estimation due to robot motion and enhances the mapping accuracy.

Loop closing with 2D LiDAR scanners has been well studied in the indoor environment [76, 162]. Stachniss et al. [162] reduce the localization error and obtain more accurate maps of a 2D LiDAR-SLAM by actively seeking loop closures. Hess et al. [76] build submaps and then align scans to submaps to generate closed loops by a branch-and-bound approach to efficiently develop more consistent global maps. For loop closing in the large-scale outdoor environment is still challenging due to the dynamic objects and the changing appearance of the environments. In this thesis, we focus more on approaches that can be applied in outdoor environments for autonomous driving.

Multiple geometric-based 3D LiDAR loop closure detection methods have been proposed. For example, Steder et al. [164] propose a place recognition system operating on range images generated from 3D LiDAR data. They first exploit a combination of bag-of-words and NARF features [165] to find the loop closure candidates, and then use the matched features for robots' relative poses estimation. Instead of using specific features, Röhling et al. [144] 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. Once finding a loop, they use ICP to calculate the relative transformation between loop candidates and integrate the loop constraints into an optimization-based SLAM system. The work by He et al. [75] presents M2DP, which projects a LiDAR scan into multiple reference planes to generate a descriptor using a density signature of points in each plane. In the end, M2DP uses the first left and right singular vectors of these 2D signatures as the descriptor to find the loop closure candidates robustly. Kim et al. [87] generate scan context images based on point clouds and use them for LiDAR-based place recognition. They use scan context images as global descriptors to encode each point cloud and find LiDAR pair matches based on such descriptors. Besides using pure geometric information, there are also methods [37, 73] exploiting the intensity/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. These works show that the LiDAR intensity information can be good supplements for finding loop closures when using LiDAR data only. Vysotska et al. [186] link LiDAR observations with background knowledge about the environment, such as building structures to create global constraints, similar to loop closures, to reduce the pose estimation uncertainty of SLAM systems. All the methods mentioned above are based on hand-designed features and geometric information, which may not be robust enough for outdoor changing environments.

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, Dube et al. [55] 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. Based on such segments, Cramariuc et al. [39] train a CNN to extract descriptors from segments and use them to retrieve nearby place candidates. Their experimental results show that the learning-based descriptors outperform hand-crafted ones in finding loop closures in real-world urban driving scenarios. Instead of using segments, Angelina et al. [181] 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. [204] develop LocNet, which uses semi-handcrafted features learning based on a siamese network to solve LiDAR-based place recognition. To recognize places and estimate the yaw angle offset of pairs of candidates at the same time, Schaupp et al. [155] propose a system called OREOS. They use a convolutional neural network to extract compact descriptors from LiDAR scans and use the features to retrieve nearby place candidates from a map and to estimate the yaw discrepancy. Most recently, there are also several works that exploit semantic information to improve loop closure detection accuracy. For example, Zaganidis et al. [209] proposed a normal distribution histogram-based loop closure detection method, which is assisted by semantic information. Kong et al. [91] 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. Li et al. [99] enhance the scan context descriptors by exploiting semantic information to improve the descriptiveness of the descriptors. A two-step global semantic ICP is then applied to align the point clouds for loop closing.

Contrary to the above-mentioned methods, our method presented in Chapter 4 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 [120]. 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 to provide an estimate of the matching quality in terms of the overlap.

9.3 LiDAR-Based Localization

Besides loop closing, our method introduced in Chapter 4 can be further used for LiDAR-based localization. Localization is a classical topic in robotics and autonomous vehicles [44, 178]. 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 initial 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. A popular traditional framework for robot localization, that relies on probabilistic state estimation techniques, is Monte-Carlo localization (MCL) [44, 62], which uses a particle filter to estimate the pose of the robot. MCL has been well-applied in 2D indoor environments with 2D laser scanners, as presented in the work by Dellaert et al. [44]. However, it is still challenging to use MCL with 3D LiDAR in outdoor environments due to the extensive computation in outdoor areas and big changes in environments caused by the dynamic objects.

High-definition (HD) maps are frequently used for localization in outdoor environments in the context of autonomous vehicles. HD maps usually include map elements such as road shape, road marking, traffic signs, etc., which are often constructed using multiple sensors, such as LiDAR, radar, digital camera, and GPS. There are many approaches proposed for building and using high-definition maps for localization in outdoor environments. For example, Levinson et al. [98] 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 et al. [198] 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. [10] use a fully convolutional neural network to perform online-to-map matching to improve the robustness of dynamic objects and eliminate the need for LiDAR intensity calibration. Their approach requires a good GPS prior to achieve good performance. Based on this approach, Wei et al. [191] 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 et al. [116] 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. [197] 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.

Despite good localization results that can be achieved using HD maps, such high-definition maps are expensive to obtain and not always available. Instead of using HD maps, we proposed several possible solutions for outdoor localization in our previous work using only LiDAR data. Our work led by Wiesmann [196] proposes a deep network to compress and decompress the point cloud maps which can be later used for online localization. Recently, our work led by Vizzo [185] also proposes a lightweight mesh mapping algorithm using Poisson surface reconstruction for LiDAR scans. Based on such mesh maps, we propose a novel LiDAR range image-based observation model for MCL to achieve LiDAR-based global localization [36]. There are also other approaches aiming at performing LiDAR-based place recognition to initialize localization in outdoor environments to relieve the computational burden. For example, Kim et al. [86, 89] 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. After finding the matched pairs of scans, they then use ICP to compute the final localization results. Yin et al. [204] 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. [37] 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 [60], and then refine the estimated transformation using ICP. In Chapter 6, we also use such a two-step scheme to achieve global localization. We first use place recognition to find initial location hypotheses and then use MCL to achieve metric localization. Different from the existing methods, we use a pole-like object-based observation model to update the importance weights of particles, which is more robust to changing environments and achieves good long-term localization performance.

Recently, multiple approaches exploiting deep neural networks and semantic information for 3D LiDAR localization have been proposed. Zhang et al. [210] 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. [109] combine semantic information such as lanes and traffic signs in a Bayesian filtering framework to achieve accurate and robust localization results. Furthermore, their approach does not require detailed knowledge about the appearance of the world, thus requiring orders of magnitude less map storage than traditional geometric maps, which is important for autonomous driving in large environments. Similarly, Yan et al. [200] exploit buildings and intersections information from a LiDAR-based semantic segmentation system to localize on OpenStreetMap data using 4-bit semantic descriptors, which is fast and lightweight. Tinchev et al. [179] propose a learning-based method to match segments of trees and localize the robot in both urban and natural environments. Their approach also learns a light feature space representation that can be deployed using only a CPU. Sun et al. [172] use a deep-probabilistic model to accelerate the initialization of the MCL 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. These learning-based semantic localization methods typically rely on sparse but distinctive semantic landmarks to achieve fast runtime, lightweight map representation, and accurate localization results for autonomous driving in large-scale environments.

Different from the above-discussed methods, our method presented in Chapter 4 and Chapter 6 only exploits LiDAR information to achieve global localization without using any other prior information. Our approach relies on convolutional neural networks to either directly predict the overlap between LiDAR scans and their yaw angle offset as described in Chapter 4 or detect pole-like objects as described in Chapter 6. We exploit neural networks and such task-specific semantic information to build new observation models for MCL which is robust and accurate.

More specifically, in Chapter 4, we exploit a siamese network architecture [23] and proposed a new network named OverlapNet [30, 31], which uses 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, normal information, intensity/remission, and probabilities of semantic classes generated by a semantic segmentation system [120]. We use the overlap and relative yaw angle estimates predicted by our OverlapNet to build a learning-based observation model for MCL as detailed in Chapter 4.

The learning-based methods perform well in the trained environments, while they usually cannot generalize well in different environments or with different LiDAR sensors. Instead of using dense multiclass semantic information estimated by neural networks [101, 120], in Chapter 6, we propose a rather lightweight solution for long-term localization, which extracts only pole landmarks from point clouds. During the mapping phase, we first extract poles in range images generated from LiDAR point clouds. After obtaining the position of poles in the range image, we use the ground truth poses of the robot to reproject them into the global coordinate system to build a global map. During online localization, we utilize MCL for updating the importance weights of the particles by matching the poles detected from online sensor data with the poles in the global map to achieve robust and reliable localization.

9.4 Task-Specific Semantics for LiDAR Perception

As discussed in Section 9.1 and Section 9.3, multiclass semantic information provided by neural networks can be used to improve the robustness and accuracy of LiDAR perception tasks, such as SLAM and localization. However, the performance of these multiclass semantic segmentation networks strongly relies on the diversity and amount of labeled training data, which may be costly to obtain.

To alleviate the strong dependence on human manual labelings, we propose using more specific semantics for different LiDAR perception tasks, which are also easier to obtain than multiclass semantics, such as using automatic labeling techniques. In this section, we present and discuss the work related to moving object segmentation and pole-like object extraction that can be used for specific tasks like SLAM and localization. We also discuss here the possible ways to automatically generate training data for each task.

9.4.1 Moving Object Segmentation for SLAM

While there has been a large interest in vision-based [9, 114, 133], radar-based [2] and vision and LiDAR combined [139, 201] moving object segmentation approaches, only using a LiDAR sensor for MOS is challenging due to the distance-dependent sparsity and uneven distribution of the range measurements. We concentrate here on approaches using only LiDAR sensors.

There are both geometric model-based and deep neural network-based methods in the literature to address the problem of online LiDAR-MOS. For example, Yoon et al. [206] detect dynamic objects in LiDAR scans by exploiting geometric heuristics, e.g., the residual between LiDAR scans, free space checking, and region growing to find moving objects. Dewan et al. [47] propose a LiDAR-based scene flow method that estimates motion vectors for rigid bodies. They formulate the problem as an energy minimization problem, where they assume local geometric constancy and incorporate regularization for smooth motion fields. Besides geometry-based approaches, there are also deep network-based methods [11, 68, 105], which use generic end-to-end trainable models to learn local and global statistical relationships directly from data and estimate the flow vector for each point in the LiDAR point clouds. Such scene flow methods usually estimate motion vectors between two consecutive scans, which may not differentiate between slowly moving objects and sensor noise.

Semantic segmentation can be seen as a relevant step toward moving object segmentation, since it can provide information about potentially moving objects, such as vehicles and pedestrians. As presented in Section 2.4, LiDAR-based semantic segmentation methods operating only on the sensor data have recently achieved great success [38, 101, 120]. Based on such online semantic segmentation networks, our method presented in Chapter 3, SuMa++ [35], exploits semantics to detect and filter out dynamic objects to improve the LiDAR SLAM performance. Based on the geometric model-based approach [47], Dewan et al. later also develop both semantic classification and segmentation methods [46, 48], which exploit the temporally consistent information from the sequential LiDAR scans to distinguish moving and non-moving objects. There are also several 3D point cloud-based semantic segmentation approaches [159, 177], which exploit sequential point clouds and predict moving objects. However, these methods require a large number of semantic labels, which are not always available and make it difficult to train and generalize to unseen data.

Different from the multiclass semantic-based methods, Wang et al. [187] 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 [146] also propose a learning-based method to predict the probabilities of potentially movable objects directly. The output from their network is fed to the mapping module for building a 3D grid map containing only static parts of the environment. Bogoslavskyi and Stachniss [20] propose a class-agnostic segmentation method for 3D LiDAR scans that exploits range images to enable online operation and leads to more coherent segments. Based on such segments, one can later distinguish between moving and non-moving objects by tracking the segments and calculating the speed [34].

Most existing semantic segmentation methods 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. There are also multiple 3D point cloud-based semantic segmentation approaches [159, 174, 177], which 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.

The most similar work to ours is the one by Yoon et al. [206], which also detects dynamic objects in LiDAR scans. It exploits heuristics, e.g., the residual between LiDAR scans, free space checking, and region growing to find moving objects. Instead of using only geometric heuristics, our method presented in Chapter 5 is based on neural networks and we investigate the usage of three recent range projection-based semantic segmentation methods proposed by Milioto et al. [120], Cortinhal et al. [38], and Li et al. [101] 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 current and past observations. 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.

Despite good LiDAR-MOS performance achieved by our approach using neural networks trained with simplified binary labels, it still relies on manual labels. Due to the sparsity of the LiDAR sensor data and the lack of publicly available datasets which contain point-wise moving and static labels, it is still challenging for the LiDAR-MOS to generalize to different environments with varying appearances. To further eliminate dependence on manual labeling, in Chapter 7, we propose an approach that can generate LiDAR-MOS labels automatically. Next, we discuss more possible ways of automatically generating training data for LiDAR-MOS, which have been proposed in the literature.

Unlike online LiDAR-MOS, label generation can be done offline, which enables better performance in perception tasks using sequential future and past observations. Multiple works have already been proposed to clean the point cloud map, which removes dynamic objects during the mapping procedure, resulting in a clean static map [4, 65, 88, 104, 129, 138, 154, 199]. Early methods by Gehrung et al. [65] and Schauer et al. [154] use time-consuming voxel ray casting to remove non-stable points which require accurately aligned poses to clean the dense terrestrial laser scans. To alleviate the computational burden, visibility-based methods have been proposed [138, 199]. This type of method associates a query point cloud to a map point within a narrow field of view, e.g., cone-shaped used by Pomerleau et al. [138] and Xiao et al. [199]. Kim et al. [88] propose a range image-based method, which exploits the consistency check between the query scan and the pre-built map to remove dynamic points. The authors use a multi-resolution false prediction reverting algorithm to refine the map. Instead of using range images, Pagad et al. [129] 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. Recently, Lim et al. [104] also propose a method to first remove dynamic objects by checking the occupancy of each sector of LiDAR scans and then relabel the points on the ground plane as static using a region growth. In contrast, Arora et al. [4] first segment out the ground plane and then remove the "ghost effect" caused by the moving object during mapping.

Even though such map cleaning methods can distinguish moving objects from the static map, there are many false positive detections possibly caused by noisy points or inaccuracies in the estimated poses. Different from that, MOS training data generation aims to accurately separate actual moving objects from static or non-moving objects. Hoermann et al. [77] propose a fully automatic labeling method to predict dynamic occupancy grids for urban autonomous driving. It exploits the occupancy grid map and Bayesian filtering to distinguish static and dynamic grids and uses them as training data to a convolutional neural network for online dynamic grid detection. Instead of using 2D occupancy grid maps, Thomas et al. [176] present a self-supervised learning approach for the semantic segmentation of 3D LiDAR scans in the indoor environments. Their method automatically labels 3D point clouds and provides movable probabilities for each point using ray-tracing. The method by Pfreundschuh et al. [135] also generates labels automatically to train a network for moving object-aware SLAM working in small areas, e.g., a hall or a train station. Different from the works mentioned above, our approach presented in Chapter 7 does not require ground truth poses and exploits tracking to distinguish between moving and non-moving objects. Furthermore, our method can be used in large-scale outdoor scenes.

9.4.2 Pole-Like Object Extraction for Localization

Using pole landmarks for localization and mapping has attracted a lot of interest in the literature with different types of sensors, such as cameras [94, 161], RGB-D sensors [84, 168], and cameras combined with LiDAR scanners [136, 151]. It has been well studied with 2D laser scanners [122, 123]. For example, FastSLAM by Montemerlo [122, 123] uses the tree trunks detected by a laser range finder as landmarks and localizes the car within the Victoria Park [72]. In this section, we focus on pole-based localization using only 3D LiDAR data, which is more robust to outdoor changing environments.

For pole-based localization with LiDAR data, there are both geometric-based methods [29, 51, 153, 156, 160, 192, 193] and learning-based methods [50, 137, 190]. For example, the method by Sefati et al. [156] first removes the ground plane from the 3D point cloud and projects the remaining points on a horizontal grid. After that, they cluster the grid cells and fit a cylinder for each cluster to find the pole-like landmarks. In the end, they use a particle filter with nearest-neighbor data association to estimate the robot's poses. Weng et al. [192] and Schaefer et al. [153] use similar particle filter-based methods to estimate the pose of the robot but use different geometric pole extractors. Weng et al. [192] exploit a heuristic that the point cloud densities on the poles are larger than their surroundings. Therefore, they first discretize the space and then extract poles based on the density of points in each voxel grid. Schaefer et al. [153] propose pole detector considering both the laser ray endpoints and the free space in between the laser sensor and the endpoints, which demonstrates good localization performance. Shi et al. [160] extract pole-like objects from the point cloud by spatial independence analysis and cylindrical or linear feature detection. Besides, they also classify the pole-like objects into street lamps, traffic signs, and utility poles by 3D shape matching.

Instead of using only geometric information, semantic information has been also widely exploited in extracting pole-like landmarks for global localization. Many works have been done using image data [61, 94, 136, 143, 161]. For example, Robert et al. [143] propose learning-based descriptors for tree bark re-identification, which can be used to detect tree landmarks in robot navigation robustly in forests. In similar forest environments, Fortin et al. [61] recently propose a wood log instance segmentation dataset for training networks to detect tree log instances that can be used for robotic navigation systems. Instead of using only camera images, Pierzchala et al. [136] combine a camera together with LiDAR, IMU, and GPS to detect tree trunks as landmarks and integrate them into a graph-SLAM. Their methods build tree trunk-aware 3D maps of forests which can be used to localize robots accurately and reliably. There are also works only using LiDAR data to detect pole-like objects for robot localization. For example, Weng et al. [193] exploit the reflective intensity information provided by LiDAR scanners to extract traffic signs which are always painted with highly reflective materials. Chen et al. [29] fuse poles and curbs information into a non-linear optimization problem to obtain the vehicle location. They propose a branch-and-bound-based global optimization method to tackle the data association problem of poles. Recently, Plachetka et al. [137] use a deep neural network for pole extraction by learning encodings of the point cloud input. They propose a respective object anchor design with an accompanying strategy for matching ground truth objects to object anchors during network training. In this way, their network can detect pole-like objects with different shapes and heights. Wang et al. [190] also exploit a neural network to infer the semantics of LiDAR point clouds and cluster pole-like objects based on semantics. The localization is then achieved by matching the pole clusters of the current local map with those of the global map.

In contrast to the existing pole-based localization methods, our method presented in Chapter 6 uses a range image-based method and avoids the comparable costly processing of 3D point cloud data [51]. Thus, our implementation is fast and tracks the pose of the robot online. Our pole extractor uses geometric information only and generalizes well to different environments and different LiDAR sensors. In Chapter 8, we also propose a neural network-based pole segmentation approach for LiDAR localization [50]. Similar to our LiDAR-MOS auto-labeling method, after simplifying the semantic information needed for localization from multiple classes into only pole-like objects, we can generate the training data automatically without human supervision. In Chapter 8, we further use the poles extracted by our geometric-based method presented in Chapter 6 as pseudo labels to train a pole segmentation network. Trained with a large number of pseudo pole labels automatically generated by our geometric-based pole extractor from different datasets, our learning-based method can generalize well in different environments and outperforms other state-of-the-art geometric methods. When integrating our pole extractors into the MCL system, both our geometric and learning-based methods achieve good localization performance. When trained with a large number of pseudo pole labels automatically generated by our geometric-based pole extractor from different datasets, our learning-based method can generalize well in different environments and even outperforms the geometric method.

Chapter 10

Conclusion

High-level semantic understanding of the surroundings is the key for autonomous vehicles to achieve full autonomy in dynamic and complex real-world environments. The main contributions of this thesis are novel approaches that exploit semantic information to improve the performance of LiDAR perception tasks such as SLAM and localization for autonomous vehicles. We divided this thesis into three parts to answer three main questions: (i) how to use existing semantic information provided by semantic segmentation methods for perception tasks, (ii) which types of semantics are more useful for specific perception tasks, and (iii) how to generate semantic labels for training networks to learn such semantics.

In Part I, we provided examples that exploit the semantic information provided by the existing semantic segmentation networks to improve the performance of LiDAR-based perception tasks. We presented in Chapter 3 a novel semantic-based LiDAR SLAM approach named SuMa++. The main contribution of our SuMa++ is that it integrates the semantic information provided by a segmentation network on LiDAR scans into a SLAM system to build dense semantic maps and provide accurate pose estimates. It explicitly checks the semantic consistencies between scans and the map to filter out dynamic objects and provide high-level constraints during the pose estimation process. We evaluated our approach on both the KITTI road and KITTI odometry datasets [66, 67]. Compared to the geometric methods, our method generates more consistent maps with semantic information even in situations with many moving objects. The evaluation results on the KITTI odometry benchmark also show that our proposed semantic-based approach outperforms geometric approaches in terms of pose estimation.

Instead of explicitly using the semantic class consistencies, in Chapter 4, we proposed a novel neural network called OverlapNet. The main contribution of our OverlapNet is to implicitly exploit different types of information generated from LiDAR scans and estimate similarities between pairs of 3D LiDAR scans.

OverlapNet uses both geometric and semantic information of pairs of LiDAR scans as input and provides similarity estimates between them in an end-to-end fashion. We integrated our OverlapNet into downstream tasks like loop closure detection and global localization and improved the performance of such tasks. We evaluated our approach on multiple datasets collected using different LiDAR scanners in various environments. The experimental results show that our OverlapNet can effectively detect loop closures surpassing the detection performance of state-of-the-art methods at that point in time, and 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. Our OverlapNet was the first network to exploit multiple cues generated by LiDAR scans and estimate both the similarity and yaw angle offset between pairs of scans. We released the implementation of our OverlapNet. It has been followed by many later works [107, 108, 142, 215], which push the state-of-the-art LiDAR-based place recognition and loop closing forward.

Instead of exploiting multiclass semantic information from existing segmentation networks, in Part II, we answered the question of which types of semantics are more useful for different specific tasks. We presented an example of using moving/non-moving semantics for SLAM task in Chapter 5. Instead of detecting all potentially movable objects such as vehicles or humans, the proposed method aims at separating the actually moving objects such as driving cars from static or non-moving objects such as buildings, parked cars, etc. It exploits sequential and temporal information to achieve an effective moving object segmentation, which is then used to improve the pose estimation and mapping results of SLAM. We compared our approach to several other state-of-the-art methods showing superior moving object segmentation quality in urban environments. Additionally, we created a new benchmark for LiDAR-based moving object segmentation based on SemanticKITTI [12, 13]. We published it together with the code of our method to allow other researchers to compare their approaches transparently. It was the first public available LiDAR-MOS benchmark, which provides dense point-wise MOS labels and a hidden test set for evaluating LiDAR-MOS approaches.

Unlike SLAM estimating the vehicle's poses with respect to the on-the-fly map, localization aims to localize the robot within a given map. Thus, for localization, pole-like objects, such as traffic signs, poles, lamps, etc., are frequently used landmarks in urban environments due to their local distinctiveness and long-term stability. We presented a novel, accurate, and fast pole extraction approach in Chapter 6 that operates online on LiDAR range images. We then used pole-like objects as landmarks and proposed a novel pole-based observation model for MCL and achieved good localization results. We tested the proposed pole extraction and localization approach on different datasets including KITTI [67], SemanticKITTI [13], NCLT [28], and MulRan [90], which were created with different LiDAR scanners, routes, and seasonal changes. The experimental results show that our approach outperforms other state-of-the-art approaches at that point in time while running online for pose tracking. In addition, we released the implementation of our approach and our pole dataset to the public for evaluating the performance of other pole extractors.

One of the bottlenecks in supervised learning approaches is the necessary amount of labeled data. Labeling training data for such approaches is a laborious task and thus expensive. In Part III, we presented two examples to automatically generate labels for training segmentation networks. By specifying and simplifying the categories of semantics for specific tasks, we turned the challenging multi-class semantic segmentation problem into easier binary classification tasks, which makes automatic label generation feasible. We provided two examples of our automatic labeling methods for LiDAR-MOS in Chapter 7 and pole segmentation in Chapter 8. For automatic labeling of LiDAR-MOS, we provided comparisons to other existing techniques that can be used for generating LiDAR-MOS labels. The experimental results suggest that our method achieves a solid performance on LiDAR-MOS label generation and substantially boosts the online performance by additional automatically generating labels. We evaluated our method on four different datasets, including our own LiDAR-MOS benchmark [33], IPB-Car [32], MulRan [90], and Apollo [106], showing strong generalization capabilities for our approach. For automatic labeling of pole segmentation, we used the extracted poles from our geometric pole extractor as pseudo labels to train a deep neural network for online range image-based pole segmentation. Boosted with pseudo pole labels extracted from multiple datasets collected in different environments, our learning-based method can run across different datasets and achieve even better localization results compared to geometric methods. We further tested our learning-based pole extraction methods for localization on different datasets, including NCLT [28] and MulRan [90]. The experimental results show that our methods outperform other baseline approaches.

Overall, this thesis presented novel approaches to exploit semantic information to improve performances of LiDAR perception tasks such as SLAM and localization for autonomous vehicles. It provided examples of how to use semantics, which types of semantics to use, and how to automatically generate such semantics, for SLAM and localization. We evaluated and tested all our proposed methods using public available datasets recorded by autonomous vehicles in different outdoor environments and showed that our proposed semantic-based methods outperformed the state-of-the-art baseline methods at that point time. We additionally released all the implementation of our methods as well as the new LiDAR-MOS and pole segmentation benchmarks.

10.1 Open Source Contributions

All our proposed algorithms presented in this thesis are open-source to facilitate further research. The links to the repositories of our implementations are as follows:

- Chapter 3 presented our proposed efficient semantic LiDAR SLAM method, called SuMa++. This implementation is available online at: `https://github.com/PRBonn/semantic_suma`.
- Chapter 4 presented our proposed OverlapNet, which is a novel neural network for LiDAR loop closing and global localization. The network itself is available at: `http://github.com/PRBonn/OverlapNet`. The implementation of using OverlapNet for global localization is available at: `https://github.com/PRBonn/overlap_localization`.
- Chapter 5 presented our proposed LiDAR-based moving object segmentation. The implementation is available at: `http://github.com/PRBonn/lidar-mos` and the benchmarks is at: `http://bit.ly/mos-benchmark`.
- Chapter 6 presented our pole-based localization method and Chapter 8 used it for training a pole segmentation network. Both implementations of our pole extractor and pole segmentation are available online at: `http://github.com/PRBonn/https://github.com/PRBonn/pole-localization`.
- Chapter 7 presented our automatic label generation pipeline for LiDAR-MOS. The implementation is available at: `http://github.com/PRBonn/auto-mos`.

10.2 Future Work

In this thesis, we provided examples that exploit semantic information to improve the performance of LiDAR perception tasks such as SLAM and localization for autonomous vehicles. We proposed approaches either directly using the multiclass semantic information from existing semantic segmentation networks or exploiting task-specific semantics for SLAM and localization individually. Despite promising results that have been achieved using our proposed approaches, multiple novel techniques developed while writing this thesis as well as some promising future works can further improve the performance of the presented perception tasks for autonomous vehicles.

First of all, different levels of semantics can be further exploited for perception tasks. The approaches presented in Part I of this thesis exploited point-wise

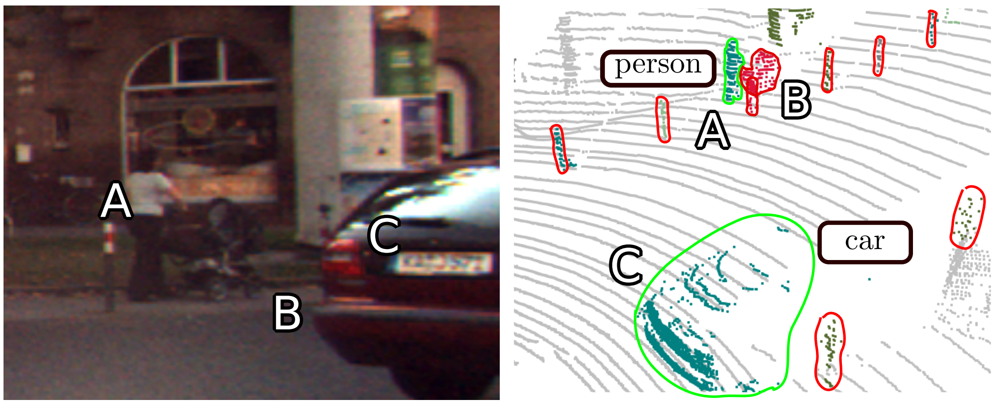


Figure 10.1: Open-world instance segmentation. The green segments outline correspond to *known* classes, i.e., person *A* and car *C*, and red outlines correspond to things that are not part of the training data. While *A* and *C* are *known* classes, *B* corresponds to a baby stroller—an *unknown* class, which obviously should be detected even though not annotated in the training set.

semantics at the measurement level. A higher object-level semantics, such as moving objects and pole-like objects presented in Part II, further improved SLAM and localization performance by providing more specific semantic information. Most recently, we also proposed instance-level semantics [14, 119, 127, 128], which not only distinguish different classes of objects, i.e., car, pedestrians, etc., but also identify and separate individual instances. For example, the joint work led by Lucas Nunes [127] exploits class-agnostic instance segments and contrastive loss to learn descriptive representations of the objects in the scene unsupervised and boosts the semantic segmentation results. Based on such unsupervised learned instance representations, we also tackle the open-world instance segmentation task [128]. As shown in Figure 10.1, one of the bottlenecks of the existing semantic segmentation is that they are bound by the labeled classes, ignoring long-tailed classes not annotated in the training data due to the scarcity of examples. However, those unknown long-tailed classes, such as a baby stroller or unseen animals, can be crucial when interpreting the vehicle surroundings for safe interaction. The cooperation led by Lucas Nunes [128] was under review during the writing of this thesis and published in the IEEE Robotics and Automation Letters (RA-L) journal before the final editing of this thesis. We believe such open-world instance segmentation is essential to achieving a scalable deployment of autonomous vehicles in the real world.

Beyond a specific level, semantic information could be also represented as a hybrid and hierarchical or even more complex way. In the same example shown in Figure 10.1, a person pushing a baby stroller could be detected as an unknown object and, at the same time, one moving object. Multiple hybrid semantics could be assigned to the same objects in favor of different tasks. Another example for mapping and localization could be hierarchically constructing or exploiting one map with multiple layers. The same map could store sensor measurements for accurate pose estimation, global descriptors for place recognition, and appearance information for environmental categorization. Such a hybrid and hierarchical understanding of the environment will provide richer information, and multiple interpolations of the same data enable us to better exploit them within different contexts at the same time.

Secondly, novel network architectures can provide a high-level understanding in both spatial and temporal dimensions for perception tasks of autonomous vehicles. The approaches presented in this thesis mostly use 2D convolutional neural networks, which are lightweight and work well with LiDAR range images. However, they may not be the best way to exploit 4D spatial-temporal information. Recently, the joint work led by Benedikt Mersch [117] uses 3D CNN architecture with consecutive LiDAR range images to better exploit spatial-temporal information for future point prediction. Predicting the surroundings state in the future can be very useful for autonomous vehicles making plans and avoiding collisions. In another work led by Benedikt Mersch [118], we use a sparse 4D CNN directly on the point clouds to improve LiDAR moving object segmentation results. Our proposed 4D CNN can directly process 3D point cloud videos using high-dimensional convolutions to fuse 3D space and an extra time dimension. Our 4D CNN outperforms our 2D CNN methods [33] in LiDAR MOS. Another cooperation led by Jiadai Sun [170] proposes a novel coarse-to-fine network, which first exploits sequential LiDAR range images to obtain temporal information and then a small 3D CNN to exploit 3D spatial information and refine the moving object segmentation results on LiDAR data. This method combines both range image and point cloud representations of LiDAR data and provides more accurate segmentation results. The work led by Junyi Ma [108] proposes OverlapTransformer, which adapts the state-of-the-art transformer network architecture [182] in natural language processing for the LiDAR-based place recognition task. Benefiting from the attention mechanism and the devised rotation-invariant architecture, the proposed OverlapTransformer recognizes the same places, even when the vehicle driving in opposite directions as shown in Figure 10.2. This enables our method to achieve better performance than our 2D CNN-based network [30] in finding loop closure candidates and recognizing places. These cooperated works [108, 118, 170] were under review during the writing of this thesis. [108, 118] are now published in the IEEE Robotics and Automation Letters (RA-L) journal, and [170] is published in the proceeding of IROS 2022.

Besides undergoing work, there are other possibilities for exploiting temporal and spatial information. For example, the current networks still exploit data within a short period. How to design a network exploiting long-term data and

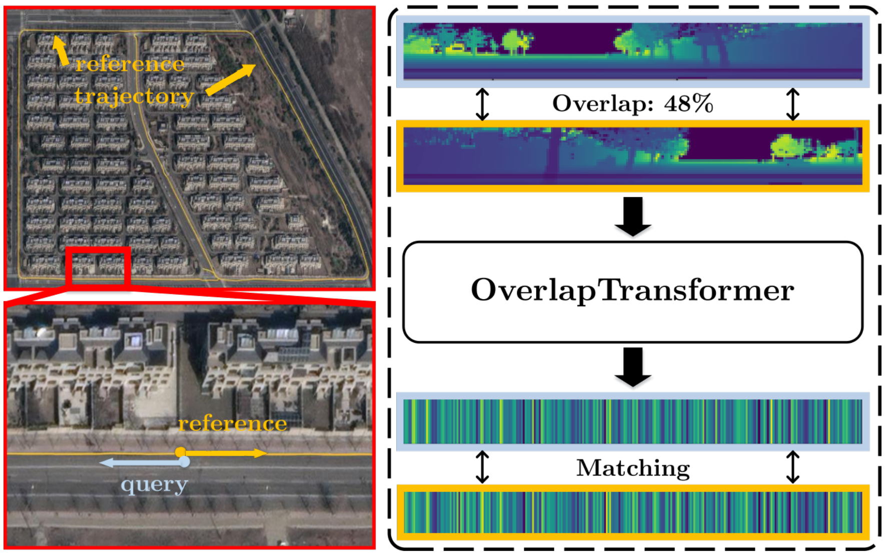


Figure 10.2: Transformer network-based LiDAR place recognition. Query scan (blue) and reference scan (orange) with adjacent locations but opposite viewpoints in our challenging real-world dataset. Our proposed OverlapTransformer is able to generate rotation-invariant global descriptors with only range images, which is robust to view-point changing for place recognition.

achieve continual learning for lifelong localization and mapping is still an open question. One possible way is to use graph or hierarchical network architectures to handle short-term adaptation and long-term memory retention separately with different network nodes or branches. Another possible way could be exploiting the probabilistic network to learn the motion probability of different objects from long-term data. One could then use objects with low motion probability for long-term mapping while high motion probability objects for local obstacle avoidance or path planning. We believe that using novel network architectures can provide a high-level understanding of both spatial and temporal dimensions and further boost the performance of the presented perception tasks in future research.

Last but not least, multi-modality sensing and information fusion can provide richer information for autonomous vehicles to better understand the world. Furthermore, the redundancy from different sensing modalities can improve the robustness and reliability of an autonomous vehicle working in real-world scenarios. Our previous work led by Wenbang Deng [45] proposes to use RGB and depth data together to achieve good semantic mapping results in urban search and rescuing environments. Based on that, we won the Best-in-Class Exploration and Mapping in Rescue Robot League at RoboCup competition 2021. Figure 10.3 shows the rescuing scenario used in RoboCup competition and the corresponding semantic map generated by our approach. As can be seen, our method works well in the complex rescuing environment and builds a consistent semantic map

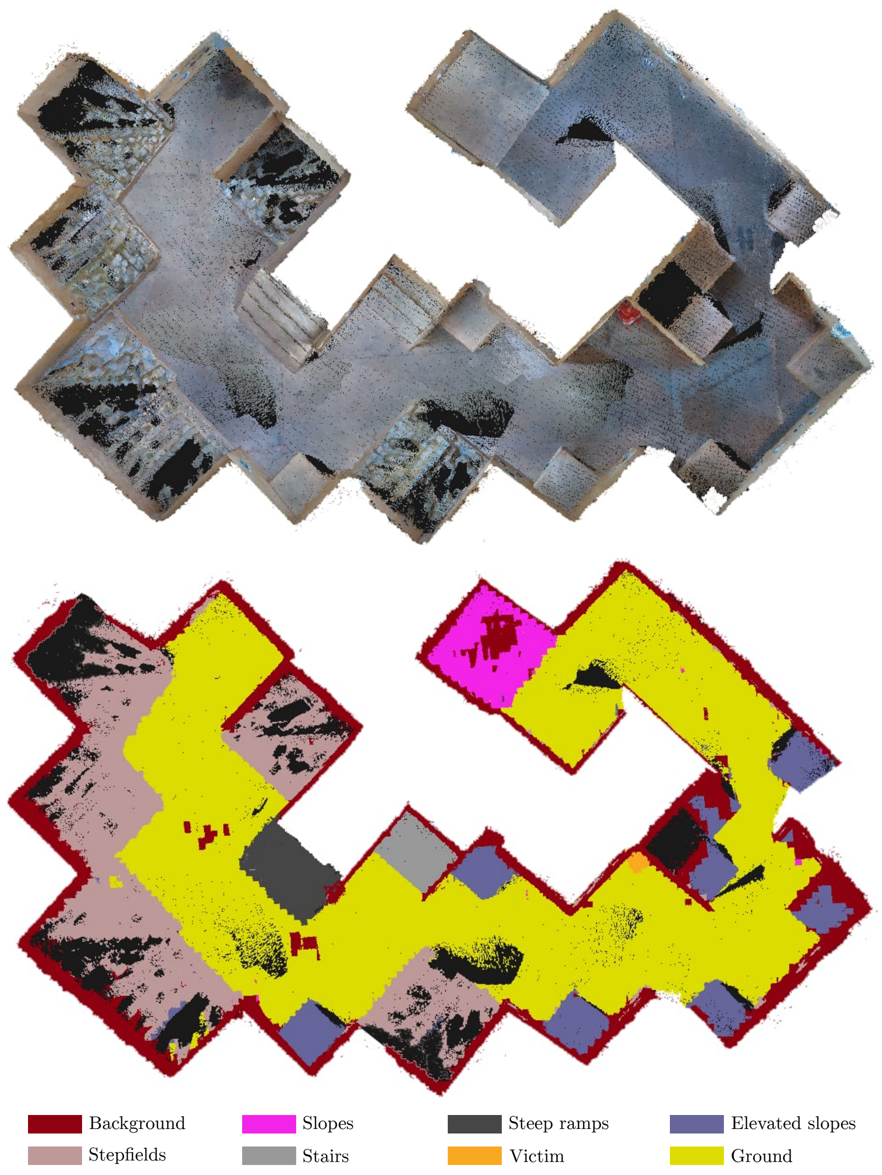


Figure 10.3: Dense semantic map generated by our approach in the RoboCup Rescue League competition field. The upper image shows the colored point-cloud map of the RRL test field (view from the top). The lower image shows the corresponding semantic map, including eight different terrains and objects.

used for the follow-up rescuing. The work led by Andrzej Reinke [141] uses a 3D LiDAR scanner and a monocular camera to provide reliable state estimation for autonomous vehicles. Our method exploits the advantages of different existing ego-motion estimating approaches and is resilient to failure cases. In the work done in cooperation with Mengjie Zhou [213], we describe an efficient cross-modality localization method. It integrates a novel observation model that matches ground-level images to 2D cartographic maps such as OpenStreetMap into the Monte Carlo localization system, and achieves convincing global localization results. This enables autonomous vehicles to localize on publicly available maps using the data from onboard cameras in urban environments. All these works [45, 141, 213] show good examples of using different sensing modalities for autonomous vehicles to boost the performance of the perception tasks and provide redundant information for the purpose of safety.

Besides the above-mentioned sensors, i.e., camera, LiDAR, and RGB-D sensors, multiple other sensors can be further used for autonomous driving, such as radars, sonars, IMUs, GPS, and even microphones. There are still many challenges to fuse different multimodal sensor data. For example, we need to first spatially and temporally align different sensor measurements with different resolutions, i.e., correctly calibrating and synchronizing multimodal sensor data. How to achieve automatically online calibration without using artificial markers is still challenging. One possible solution could be using deep neural networks together with sequential data to online predict the relative transformations between the sensors. Using the sequential data and the geometry constraints, such networks could be trained in a self-supervised way. Another challenge is to fuse multimodal sensor data from totally different domains, such as image and sound. It is very natural and relatively simple for human drivers to make driving actions according to these two biometric cues. One possible way to fuse them for autonomous driving is to mimic the human brain and use neural networks, which map such different modalities into embedding space, find correspondences and then combine them for better awareness of the surrounding environment. We believe that a better way to fuse more sensor information for future research of autonomous vehicles will bring better understanding of the environments and enable more robust and reliable autonomous mobile systems.

Bibliography

[1] E. E. Aksoy, S. Baci, and S. Cavdar. Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2020.

[2] 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 (ICRA)*, 2019.

[3] B. Arain, C. McCool, P. Rigby, D. Cagara, and M. Dunbabin. Improving underwater obstacle detection using semantic image segmentation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[4] M. Arora, L. Wiesmann, X. Chen, and C. Stachniss. Mapping the Static Parts of Dynamic Scenes from 3D LiDAR Point Clouds Exploiting Ground Segmentation. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2021.

[5] M. Arora, L. Wiesmann, X. Chen, and C. Stachniss. Static Map Generation from 3D LiDAR Point Clouds Exploiting Ground Segmentation. *Journal on Robotics and Autonomous Systems (RAS)*, 2022.

[6] T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part I. *IEEE Robotics and Automation Magazine (RAM)*, 13(2):99–110, 2006.

[7] T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine (RAM)*, 13(3):108 –117, 2006.

[8] T. D. Barfoot. *State estimation for robotics.* Cambridge University Press, 2017.

[9] 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 (ICRA)*, 2018.

[10] I. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun. Learning to Localize Using a LiDAR Intensity Map. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2018.

[11] S. A. Baur, D. J. Emmerichs, F. Moosmann, P. Pinggera, B. Ommer, and A. Geiger. SLIM: Self-Supervised LiDAR Scene Flow and Motion Segmentation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] J. Behley, V. Steinhage, and A. Cremers. Efficient Radius Neighbor Search in Three-dimensional Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[18] B. Bescos, J. Fácil, 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.

[19] P. Besl and N. McKay. A Method for Registration of 3D Shapes. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992.

[20] I. Bogoslavskyi and C. 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.

[21] S. Bowman, N. Atanasov, K. Daniilidis, and G. Pappas. Probabilistic Data Association for Semantic SLAM. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[22] 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)*, 2018.

[23] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature Verification using a “Siamese” Time Delayed Neural Network. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 07(04):669–688, 1993.

[24] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and 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.

[25] 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 *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proc. of Pacific-Asia. Conf. on Knowledge Discovery and Data Mining*, 2013.

[27] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. on Knowledge Discovery from Data (TKDD)*, 10(1):1–51, 2015.

[28] N. Carlevaris-Bianco, A. Ushani, and R. Eustice. University of Michigan North Campus long-term vision and lidar dataset. *Intl. Journal of Robotics Research (IJRR)*, 35(9):1023–1035, 2016.

[29] G. Chen, F. Lu, Z. Li, Y. Liu, J. Dong, J. Zhao, J. Yu, and A. Knoll. Pole-Curb Fusion Based Robust and Efficient Autonomous Vehicle Localization System With Branch-and-Bound Global Optimization and Local Grid Map Method. *IEEE Trans. on Vehicular Technology*, 70(11):11283–11294, 2021.

[30] 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*, 46:61–81, 2021.

[31] 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.

[32] 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.

[33] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss. Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data. *IEEE Robotics and Automation Letters (RA-L)*, 6:6529–6536, 2021.

[34] X. Chen, B. Mersch, L. Nunes, R. Marcuzzi, I. Vizzo, J. Behley, and C. Stachniss. Automatic Labeling to Generate Training Data for Online LiDAR-Based Moving Object Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):6107–6114, 2022.

[35] 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.

[36] 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.

[37] K. Cop, P. 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.

[38] 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.

[39] A. Cramariuc, R. Dube, H. Sommer, R. Siegwart, and I. Gilitschenski. Learning 3D Segment Descriptors for Place Recognition. In *Proc. of the IROS Workshop on Learning for Localization and Mapping*, 2017.

[40] I. Cvisic, J. Cesic, I. Markovic, and I. Petrovic. SOFT-SLAM: Computationally Efficient Stereo Visual SLAM for Autonomous UAVs. *Journal of Field Robotics (JFR)*, 35(4):578–595, 2017.

[41] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. on Graphics (TOG)*, 36(4):1, 2017.

[42] Daniel Wilbers. *Graph-based Sliding Window Localization and Map Refinement for Automated Vehicles.* PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2021.

[43] A. Das, J. Servos, and S. 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.

[44] 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.

[45] W. Deng, K. Huang, X. Chen, Z. Zhou, C. Shi, R. Guo, and H. Zhang. Semantic RGB-D SLAM for Rescue Robot Navigation. *IEEE Access*, 8:221320–221329, 2020.

[46] A. Dewan and W. Burgard. DeepTemporalSeg: Temporally Consistent Semantic Segmentation of 3D LiDAR Scans. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[47] 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 (IROS)*, 2016.

[48] 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 (IROS)*, 2017.

[49] R. Dobrushin. Prescribing a system of random variables by conditional distributions. *Theory of Probability & Its Applications*, 15(3):458–486, 1970.

[50] H. Dong*, X. Chen*, S. Särkkä, and C. Stachniss. Online Pole Segmentation on Range Images for Long-term LiDAR Localization in Urban Environments. *Journal on Robotics and Autonomous Systems (RAS)*, 2022. (*authors contributed equally.).

[51] H. Dong*, X. Chen*, and C. Stachniss. Online Range Image-based Pole Extractor for Long-term LiDAR Localization in Urban Environments. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2021. (*authors contributed equally.).

[52] D. Droeschel and S. Behnke. Efficient continuous-time slam for 3d lidar-based online mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[53] 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.

[54] R. Dube, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena. Segmap: Segment-based mapping and localization using data-driven descriptors. *Intl. Journal of Robotics Research (IJRR)*, 39(2-3):339–355, 2020.

[55] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Lerma. SegMatch: Segment Based Place Recognition in 3D Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[56] M. Elhousni and X. Huang. A survey on 3d lidar localization for autonomous vehicles. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2020.

[57] 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.

[58] 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.

[59] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[60] 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.

[61] J.-M. Fortin, O. Gamache, V. Grondin, F. Pomerleau, and P. Giguère. Instance segmentation for autonomous log grasping in forestry operations. *arXiv preprint, arXiv:2203.01902*, 2022.

[62] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, 1999.

[63] P. Ganti and S. L. Waslander. Network uncertainty informed semantic feature selection for visual slam. In *Proc. of the Conf. on Computer and Robot Vision (CRV)*, 2019.

[64] S. Garg, N. Sünderhauf, F. Dayoub, D. Morrison, A. Cosgun, G. Carneiro, Q. Wu, T.-J. Chin, I. Reid, S. Gould, P. Corke, and M. Milford. Semantics for robotic mapping, perception and interaction: A survey. *Foundations and Trends® in Robotics*, 8(1–2):1–224, 2020.

[65] 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.

[66] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The KITTI Dataset. *Intl. Journal of Robotics Research (IJRR)*, 32(11), 2013.

[67] 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.

[68] 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.

[69] W. S. Grant, R. C. Voorhies, and L. Itti. Efficient velodyne slam with point and plane features. *Autonomous Robots*, 43(5):1207–1224, 2019.

[70] 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.

[71] T. Guadagnino, X. Chen, M. Sodano, J. Behley, G. Grisetti, and C. Stachniss. Fast Sparse LiDAR Odometry Using Self-Supervised Feature Selection on Intensity Images. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7597–7604, 2022.

[72] J. E. Guivant and E. M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Trans. on Robotics and Automation*, 17(3):242–257, 2001.

[73] J. Guo, P. Borges, C. Park, and A. Gawel. Local descriptor for robust place recognition using LiDAR intensity. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1470–1477, 2019.

[74] D. Hall, F. Dayoub, J. Kulk, and C. McCool. Towards Unsupervised Weed Scouting for Agricultural Robotics. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[75] 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.

[76] 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.

[77] S. Hoermann, M. Bach, and K. Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[78] P. J. Huber. *Robust Statistics.* Wiley, 1981.

[79] 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. 2004.

[80] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction using a Moving Depth Camera. In *ACM Symposium on User Interface Software and Technology*, pages 559–568, 2011.

[81] J. Jeong, T. Yoon, and J. Park. Towards a Meaningful 3D Map Using a 3D Lidar and a Camera. *IEEE Sensors Journal*, 18(8), 2018.

[82] 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.

[83] N. Jonnavithula, Y. Lyu, and Z. Zhang. Lidar odometry methodologies for autonomous driving: A survey. *arXiv preprint, arXiv:2109.06120*, 2021.

[84] T. Kanji. Active cross-domain self-localization using pole-like landmarks. In *the Intl. Conf. on Mechatronics and Automation (ICMA)*, 2021.

[85] Ç. Kaymak and A. Uçar. A brief survey and an application of semantic image segmentation for autonomous driving. In *Handbook of Deep Learning Applications*, pages 161–200. 2019.

[86] G. Kim, S. Choi, and A. Kim. Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments. *IEEE Trans. on Robotics (TRO)*, 8:1–19, 2021.

[87] G. Kim and A. Kim. Scan Context: Egocentric Spatial Descriptor for Place Recognition within 3D Point Cloud Map. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[88] 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 (IROS)*, 2020.

[89] 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.

[90] 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.

[91] X. Kong, X. Yang, G. Zhai, X. Zhao, X. Zeng, M. Wang, Y. Liu, W. Li, and F. Wen. Semantic Graph based Place Recognition for Point Clouds. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[92] I. Kostavelis and A. Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Journal on Robotics and Autonomous Systems (RAS)*, 66:86–103, 2015.

[93] H. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[94] J. Kümmerle, M. Sons, F. Poggenhans, T. Kühner, M. Lauer, and C. Stiller. Accurate and efficient self-localization on roads using basic geometric primitives. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[95] 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.

[96] 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.

[97] C. Lehnert, D. Tsai, A. Eriksson, and C. McCool. 3d move to see: Multi-perspective visual servoing towards the next best view within unstructured and occluded environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2019.

[98] J. Levinson, M. Montemerlo, and S. Thrun. Map-Based Precision Vehicle Localization in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[99] L. Li, X. Kong, X. Zhao, T. Huang, W. Li, F. Wen, H. Zhang, 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.

[100] 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.

[101] 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. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):738–745, 2022.

[102] X. Li, Z. Liu, P. Luo, 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.

[103] K. Lianos, J. Schonberger, M. Pollefeys, and T. Sattler. VSO: Visual Semantic Odometry. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2018.

[104] H. Lim, S. Hwang, and H. Myung. ERASOR: Egocentric Ratio of Pseudo Occupancy-Based Dynamic Object Removal for Static 3D Point Cloud Map Building. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):2272–2279, 2021.

[105] 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 (CVPR)*, 2019.

[106] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song. L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[107] J. Ma, G. Xiong, J. Xu, J. Song, and D. Sun. Mutual pose recognition based on multiple cues and uncertainty capture in multi-robot systems. In *Proc. of the IEEE Intl. Conf. on Unmanned Systems (ICUS)*, 2021.

[108] J. Ma, J. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen. OverlapTransformer: An Efficient and Yaw-Angle-Invariant Transformer Network for LiDAR-Based Place Recognition. *IEEE Robotics and Automation Letters*, 7(3):6958–6965, 2022.

[109] 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.

[110] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 year, 1000 km: The oxford robotcar dataset. *Intl. Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[111] C. McCool, T. Perez, and B. Upcroft. Mixtures of Lightweight Deep Convolutional Neural Networks: Applied to Agricultural Robotics. *IEEE Robotics and Automation Letters (RA-L)*, 2017.

[112] 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)*, 2018.

[113] J. McCormac, A. Handa, A. 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.

[114] 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 (ICRA)*, 2013.

[115] J.-P. Mercier, M. Garon, P. Giguere, and J.-F. Lalonde. Deep template-based object instance detection. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2021.

[116] C. 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.

[117] B. Mersch, X. Chen, J. Behley, and C. Stachniss. Self-supervised Point Cloud Prediction Using 3D Spatio-temporal Convolutional Networks. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2021.

[118] B. Mersch, X. Chen, I. Vizzo, L. Nunes, J. Behley, and C. Stachniss. Receding Moving Object Segmentation in 3D LiDAR Data Using Sparse 4D Convolutions. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7503–7510, 2022.

[119] 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.

[120] 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.

[121] S. Mohapatra, M. Hodaei, S. Yogamani, S. Milz, P. Maeder, H. Gotzig, M. Simon, and H. Rashed. LiMoSeg: Real-time Bird's Eye View based LiDAR Motion Segmentation. In *Proc. of the International Conf. on Computer Vision Theory and Applications*, 2022.

[122] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, 2002.

[123] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Intl. Conf. on Artificial Intelligence (IJCAI)*, 2003.

[124] F. Moosmann and C. Stiller. Velodyne SLAM. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2011.

[125] R. Mur-Artal, J. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. on Robotics (TRO)*, 31(5):1147–1163, 2015.

[126] S. Nagashima, K. Ito, T. Aoki, H. Ishii, and K. Kobayashi. A high-accuracy rotation estimation algorithm based on 1D phase-only correlation. In *Proc. of the Intl. Conf. on Image Analysis and Recognition*, 2007.

[127] L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2116–2123, 2022.

[128] L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2116–2123, 2022.

[129] 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 (ICRA)*, 2020.

[130] G. Pandey, J. McBride, and R. Eustice. Ford campus vision and lidar data set. *Intl. Journal of Robotics Research (IJRR)*, 30(13):1543–1552, 2011.

[131] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan. Elastic lidar fusion: Dense map-centric continuous-time slam. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[132] S. Parkison, L. Gan, M. Jadidi, and R. Eustice. Semantic Iterative Closest Point through Expectation-Maximization. In *Proc. of British Machine Vision Conference (BMVC)*, 2018.

[133] P. Patil, K. 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 (CVPR)*, 2020.

[134] L. Peters, D. Fridovich-Keil, V. Rubies-Royo, C. J. Tomlin, and C. Stachniss. Inferring objectives in continuous dynamic games from noise-corrupted partial state observations. In *Proc. of Robotics: Science and Systems (RSS)*, 2021.

[135] P. Pfreundschuh, H. F. C. Hendrikx, V. Reijgwart, R. Dubé, R. Siegwart, and A. Cramariuc. Dynamic Object Aware LiDAR SLAM based on Automatic Generation of Training Data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[136] M. Pierzchała, P. Giguère, and R. Astrup. Mapping forests using an unmanned ground vehicle with 3d lidar and graph-slam. *Computers and Electronics in Agriculture*, 145:217–225, 2018.

[137] C. Plachetka, J. Fricke, M. Klingner, and T. Fingscheidt. Dnn-based recognition of pole-like objects in lidar point clouds. In *Proc. of the IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, 2021.

[138] 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.

[139] 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 (IROS)*, 2016.

[140] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint, arXiv:1804.02767v1*, 2018.

[141] A. Reinke, X. Chen, and C. Stachniss. Simple But Effective Redundant Odometry for Autonomous Vehicles. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[142] Y. Ren, B. Liu, R. Cheng, and C. Agia. Lightweight semantic-aided localization with spinning lidar sensor. *IEEE Trans. on Intelligent Vehicles*, 2021.

[143] M. Robert, P. Dallaire, and P. Giguère. Tree bark re-identification using a deep-learning feature descriptor. In *Proc. of the Conf. on Computer and Robot Vision (CRV)*, 2020.

[144] 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.

[145] O. Ronneberger, P.Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015.

[146] 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 (ICRA)*, 2018.

[147] 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)*, 2018.

[148] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. of Int. Conf. on 3-D Digital Imaging and Modeling*, 2001.

[149] R. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[150] J. Saarinen, T. Stoyanov, H. Andreasson, and A. 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.

[151] S. Sabatini, M. Corno, S. Fiorenti, and S. M. Savaresi. Vision-based pole-like obstacle detection and localization for urban mobile robots. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2018.

[152] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. 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)*, 2013.

[153] 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)*, 2019.

[154] 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 (RA-L)*, 3(3):1679–1686, 2018.

[155] L. Schaupp, M. Bürki, R. Dubé, R. Siegwart, and C. 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.

[156] M. Sefati, M. Daum, B. Sondermann, K. D. Kreisköther, and A. Kampker. Improving vehicle localization using semantic and pole-like landmarks. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2017.

[157] T. Shan and B. Englot. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[158] C. Shi, X. Chen, K. Huang, J. Xiao, H. Lu, and C. Stachniss. Keypoint Matching for Point Cloud Registration using Multiplex Dynamic Graph Attention Networks. *IEEE Robotics and Automation Letters (RA-L)*, 6:8221–8228, 2021.

[159] 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 (CVPR)*, 2020.

[160] Z. Shi, Z. Kang, Y. Lin, Y. Liu, and W. Chen. Automatic recognition of pole-like objects from mobile laser scanning point clouds. *Remote Sensing*, 10(12), 2018.

[161] R. Spangenberg, D. Goehring, and R. Rojas. Pole-Based Localization for Autonomous Vehicles in Urban Scenarios. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[162] C. Stachniss, D. Hähnel, and W. Burgard. Exploration with Active Loop-Closing for FastSLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2004.

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

[164] 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.

[165] B. Steder, R. Rusu, K. Konolige, and W. 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.

[166] 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.

[167] M. Sualeh and G.-W. Kim. Simultaneous localization and mapping in the epoch of semantics: a survey. *Intl. Journal of Control, Automation and Systems*, 17(3):729–742, 2019.

[168] M. N. Sudin, S. Abdullah, and M. Nasudin. Humanoid localization on robocup field using corner intersection and geometric distance estimation. *Int. Journal of Interactive Multimedia & Artificial Intelligence*, 5(7), 2019.

[169] N. Suenderhauf, 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)*, 2017.

[170] J. Sun, Y. Dai, X. Zhang, J. Xu, R. Ai, W. Gu, C. Stachniss, and X. Chen. Efficient Spatial-Temporal Information Fusion for LiDAR-Based 3D Moving Object Segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[171] J. Sun, Y. Wang, M. Feng, D. Wang, J. Zhao, C. Stachniss, and X. Chen. ICK-Track: A Category-Level 6-DoF Pose Tracker Using Inter-Frame Consistent Keypoints for Aerial Manipulation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[172] 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.

[173] 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.

[174] 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.

[175] 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)*, 2017.

[176] H. Thomas, B. Agro, M. Gridseth, J. Zhang, and T. D. Barfoot. Self-supervised learning of lidar segmentation for autonomous indoor navigation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[177] 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.

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

[179] 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.

[180] C. Toft, W. Maddern, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, T. Pajdla, F. Kahl, and T. Sattler. Long-term visual localization revisited. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 2020.

[181] A. Uy and G. 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)*, 2018.

[182] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[183] 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.

[184] 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.

[185] 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.

[186] O. Vysotska and C. Stachniss. Improving slam by exploiting building information from publicly available maps and localization priors. *Photogrammetrie – Fernerkundung – Geoinformation (PFG)*, 85(1):53–65, 2017.

[187] D. 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 (ICRA)*, 2012.

[188] 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.

[189] 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 (TPAMI), 41(11):2740–2755, 2018.

[190] Z. Wang, S. Li, M. Cao, H. Chen, and Y. Liu. Pole-like objects mapping and long-term robot localization in dynamic urban scenarios. In *Proc. of the IEEE Conf. on Robotics and Biomimetics (ROBIO)*, 2021.

[191] 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)*, 2019.

[192] L. Weng, M. Yang, L. Guo, B. Wang, and C. Wang. Pole-based real-time localization for autonomous driving in congested urban scenarios. In *Proc. of the Intl. Conf. on Real-time Computing and Robotics (RCAR)*, 2018.

[193] S. Weng, J. Li, Y. Chen, and C. Wang. Road traffic sign detection and classification from mobile LiDAR point clouds. In *Proc. of the Intl. Conf. on Computer Vision in Remote Sensing (CVRS)*, 2016.

[194] 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.

[195] 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.

[196] 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)*, 6:2060–2067, 2021.

[197] 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.

[198] R. Wolcott and R. Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[199] 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 (JPRS)*, 107:38–49, 2015.

[200] 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.

[201] 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.

[202] 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.

[203] S. Yang, L. Zheng, X. Chen, L. Zabawa, M. Zhang, and M. Wang. Transfer Learning from Synthetic In-vitro Soybean Pods Dataset for In-situ Segmentation of On-branch Soybean Pod. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*, 2022.

[204] 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.

[205] T. Yin, X. Zhou, and P. Krahenbuhl. Center-Based 3D Object Detection and Tracking. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[206] 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 (CRV)*, 2019.

[207] 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)*, 2018.

[208] 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.

[209] A. Zaganidis, A. Zerntev, T. Duckett, and G. 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.

[210] 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)*, 2018.

[211] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*, 2014.

[212] J. Zhang and S. Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41:401–416, 2017.

[213] M. Zhou, X. Chen, N. Samano, C. Stachniss, and A. Calway. Efficient localisation using images and openstreetmaps. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[214] Q. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv preprint, arXiv:1801.09847*, 2018.

[215] Y. Zhou, Y. Wang, F. Poiesi, Q. Qin, and Y. Wan. Loop closure detection using local 3d deep descriptors. *IEEE Robotics and Automation Letters (RA-L)*, 5:18–25, 2022.

List of Figures

List of Tables

List of Algorithms