UNIVERSITÄT BONN

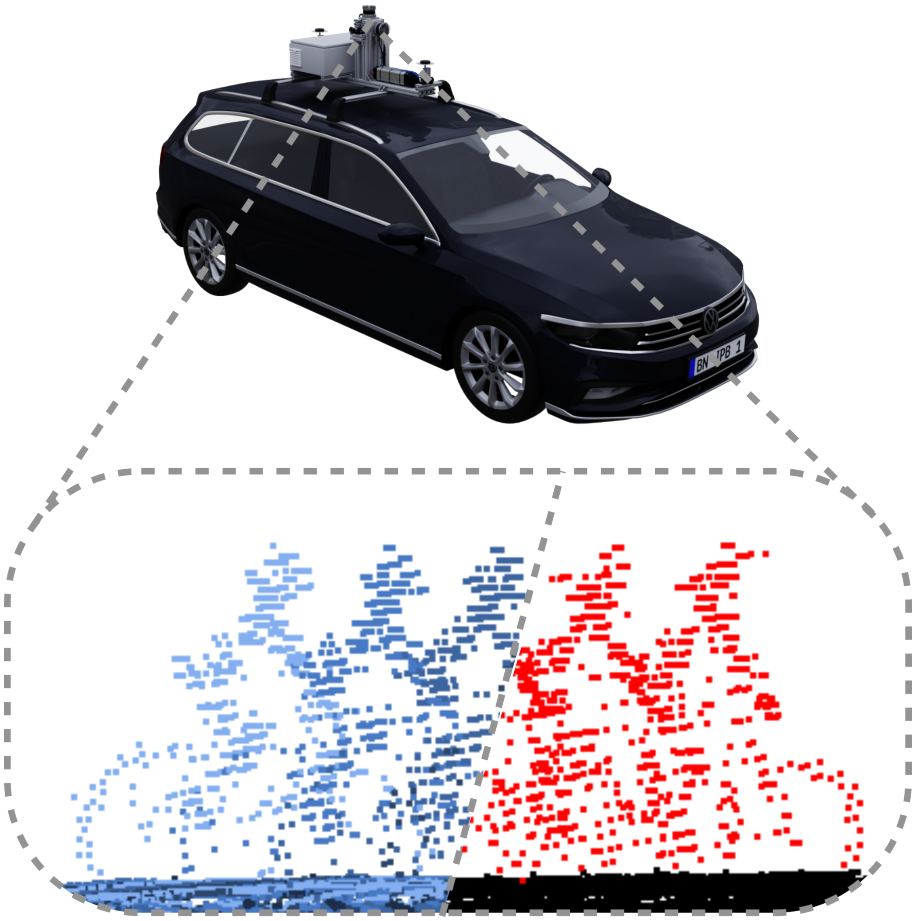
Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
Agrar-, Ernährungs- und Ingenieurwissenschaftliche Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn
Institut für Geodäsie und Geoinformation

# Spatio-Temporal Perception for Mobile Robots in Dynamic Environments

von

Benedikt Mersch

aus
Soest, Deutschland



Bonn 2025

**Referent:**
Prof. Dr. rer. nat. Cyrill Stachniss, Universität Bonn, Deutschland

**1. Korreferent:**
PD Dr. rer. nat. Lasse Klingbeil, Universität Bonn, Deutschland

**2. Korreferent:**
Prof. Dr.-Ing. Marc Hanheide, University of Lincoln, Großbritannien

Tag der mündlichen Prüfung: 10. April 2025

Angefertigt mit Genehmigung der Agrar-, Ernährungs- und Ingenieurwissenschaftlichen Fakultät der Universität Bonn

# Zusammenfassung

Roboter haben das Potenzial, Aufgaben in Umgebungen zu übernehmen, die für Menschen zu gefährlich, komplex oder anspruchsvoll sind. Solche Umgebungen stellen aufgrund ihrer dynamischen und oft unvorhersehbaren Natur besondere Herausforderungen dar. Mobile Systeme müssen daher in der Lage sein, kurz- und langfristige Veränderungen in ihrer Umwelt zu erkennen und entsprechend darauf zu reagieren. Mithilfe intelligenter Wahrnehmungssysteme können Roboter Unfälle verhindern, indem sie beispielsweise andere Verkehrsteilnehmer erkennen und rechtzeitig Ausweichmanöver einleiten. Zudem können sie autonom Personen und Güter transportieren, zeitaufwändige Aufgaben wie die Überwachung von Pflanzenwachstum übernehmen oder gefährliche Missionen im Katastrophenschutz durchführen. Generell erledigen Roboter Aufgaben effizienter und sicherer als Menschen, die durch Müdigkeit, Unaufmerksamkeit oder eingeschränkte Sinneswahrnehmungen beeinträchtigt sein können.

In den genannten Szenarien arbeiten mobile Roboter in der Regel autonom und nehmen kontinuierlich ihre Umgebung wahr, um sowohl ihren internen Zustand als auch die Umwelt zu analysieren. Dazu nutzen sie Sensoren wie globale Navigationssatellitensysteme, Kameras, Radarsensoren, inertiale Messeinheiten oder LiDAR-Scanner. Zu den wichtigsten Aufgaben gehören die Erstellung von Karten, die Lokalisierung in diesen Karten und die Segmentierung von Sensordaten in Klassen wie Autos, Gebäude oder Verkehrsschilder. Um diese Aufgaben effektiv zu lösen, muss ein mobiler Roboter seine komplexe und dynamische Umgebung sowohl räumlich als auch zeitlich wahrnehmen. Dazu analysiert er bewegte Objekte wie Menschen und erkennt gleichzeitig strukturelle Veränderungen der Umwelt. Eine besondere Herausforderung ist, dass weder die Beschaffenheit der Umgebung noch die Eigenschaften der Objekte im Vorfeld bekannt sind. Dies erfordert eine hohe Robustheit der Systeme gegenüber Unsicherheiten sowie die Fähigkeit, über verschiedene Sensorkonfigurationen hinweg zu generalisieren.

Diese Arbeit konzentriert sich auf zwei zentrale Fragen beim Einsatz mobiler Roboter in unbekannten und dynamischen Umgebungen: “Was bewegt sich?” und “Wohin bewegt sich ein Objekt?”. Um diese zu beantworten, verarbeiten

und interpretieren wir räumliche und zeitliche Daten. Erstens müssen wir bewegte Objekte identifizieren, da diese temporäre Hindernisse für die Online-Pfadplanung darstellen. Zweitens ist es notwendig, den zukünftigen Zustand der Umgebung zu schätzen, um beispielsweise einen Pfad zu planen, der das zukünftige Verhalten anderer Verkehrsteilnehmer berücksichtigt.

Im ersten Teil der Arbeit beantworten wir die Frage "Was bewegt sich?". Dabei analysieren wir sequenzielle LiDAR-Punktwolken und segmentieren sie in bewegte und unbewegte Punkte. Anschließend verbessern wir das System, indem wir eine lokale Karte für die Segmentierung nutzen und die dynamische Umgebung in einer volumetrischen Repräsentation modellieren. Um die Generalisierungsfähigkeit der Ansätze zu demonstrieren, testen wir verschiedene Trainingskonfigurationen mit Daten, die von mehreren Sensoren mit unterschiedlichen Scanmustern erfasst wurden. Zudem sind wir in der Lage, langfristige Veränderungen wie geparkte Autos oder wachsende Vegetation zu erkennen und diese Objekte für die Lokalisierung in einer gegebenen Karte herauszufiltern.

Der zweite Teil der Arbeit befasst sich mit der Frage "Wohin bewegt sich ein Objekt?". Dazu stellen wir eine kompakte Darstellung der lokalen Nachbarschaft eines Zielfahrzeugs in Autobahnszenarien vor. Wir entwickeln eine Methode, die diese räumlich-zeitlichen Daten effizient verarbeitet und zukünftige Manöver und Trajektorien der Fahrzeuge vorhersagt. Im letzten Kapitel entwerfen wir ein System, das die gesamte Umgebung anhand einer Sequenz vergangener LiDAR Punktwolken zeitlich vorhersagt. Diese Methode erfordert keine manuell erstellten Annotationen wie Referenztrajektorien und lässt sich daher leicht auf neue Umgebungen übertragen sowie online trainieren und evaluieren.

Mit dieser Arbeit leisten wir wesentliche Fortschritte zur Weiterentwicklung der räumlich-zeitlichen Wahrnehmung in dynamischen Umgebungen. Wir entwickeln mehrere neuartige Methoden, um LiDAR Punktwolken zu segmentieren und zeitliche Vorhersagen der Umgebung zu treffen. Wir zeigen in diversen Experimenten, dass wir die Methoden erfolgreich auf reale Daten in unbekannten Umgebungen mit unterschiedlichen Sensorkonfigurationen übertragen können. Außerdem stellen wir neue annotierte Datensätze und Open-Source Implementierungen unserer Ansätze zur Verfügung und haben alle vorgestellten Beiträge in Zeitschriftenartikeln oder begutachteten Konferenzbeiträgen veröffentlicht.

# Abstract

Robotic systems have the potential to revolutionize operations in environments that are too dangerous, intricate, or demanding for humans. These environments pose notable challenges due to their dynamic and often unpredictable nature, requiring robots to identify and adapt to short- and long-term changes. By leveraging advanced perception capabilities, robots can address critical tasks such as preventing accidents by detecting and reacting to vulnerable road users. They can also autonomously transport humans and goods while adapting to evolving demands, carrying out time-consuming tasks like crop monitoring, or accomplishing dangerous missions like disaster response. Robots can execute tasks more efficiently and safely by overcoming human limitations such as fatigue, inattention, or restricted sensory perception.

In these scenarios, mobile robots typically operate autonomously, continuously perceiving their environment to estimate both their internal state and the state of their surroundings. They usually rely on sensors like global navigation satellite system receivers, cameras, radar sensors, inertial measurement units, or LiDAR scanners. Key tasks include building maps of the environment, localizing in such maps, or segmenting different classes like cars, buildings, or traffic signs. Urban environments are often complex and dynamic, containing moving objects like humans or undergoing structural changes. To solve such tasks, a mobile robot must possess both spatial awareness and spatio-temporal perception – an understanding of how the environment evolves and what is changing in it explicitly. A major challenge these approaches encounter is the unknown nature of the environment beforehand, which requires the system to be highly robust and able to generalize across different sensor configurations and settings.

This thesis focuses on two main questions when deploying mobile robots in unknown and dynamic environments: "What is moving?" and "Where is an object moving to?". We must process and interpret spatial and temporal data to address these. First, knowing which parts of the environment belong to moving objects is an essential spatio-temporal perception task for online path planning. For example, moving objects occupy space only temporarily, meaning we can consider the space again traversable for planning after the object has moved.

Moving objects can also advance into areas previously regarded as free, causing potential collisions with our planned trajectory. Second, we are interested in estimating the future state of the surroundings. This prediction enables us to, for example, properly plan a future path that reflects the future behavior of other traffic participants.

In the first part of the thesis, we answer the question of "What is moving?". We utilize sequential LiDAR scans and segment them into moving and non-moving points to know which parts of the environment are currently moving. We further improve the system by identifying moving objects in a local map and integrating them into a volumetric belief about the dynamic environment. To demonstrate the generalization capabilities of these two approaches, we evaluate multiple training and inference setups using data acquired from various sensors with different sensor patterns. Finally, we also consider long-term changes such as parked cars or growing vegetation as dynamic since they have moved from a prior point in time. We use the previously presented architectures to filter out unstable points for localization in a given map.

The second part of the thesis addresses the question of "Where is an object moving to?". We propose a compact representation of the local neighborhood of a target vehicle in a highway scene and develop an approach for efficiently processing spatio-temporal data to predict future maneuvers and trajectories of the vehicles. Lastly, we design a system that predicts a future representation of the entire environment given a sequence of past LiDAR scans by estimating what the sensor will perceive next. Since this task does not require external supervision, our approach generalizes well to new environments, and we can train and evaluate it online without needing large amounts of labeled data.

This thesis offers multiple contributions to advancing spatio-temporal perception for mobile robots in dynamic environments. We propose several novel methods to process and interpret spatio-temporal data, particularly for LiDAR segmentation and future state prediction. We demonstrate the generalization capabilities of these methods in various experiments on real-world data with different sensor configurations. We also make new labeled datasets and open-source implementations of our approaches available to the research community. We published all presented techniques in peer-reviewed conference papers or journal articles.

# Acknowledgements

The long journey of pursuing a Ph.D. has been both a rigorous challenge and a profoundly rewarding experience. I am truly grateful that I did not walk this path alone but was surrounded by incredible and talented individuals who supported me in various ways to reach this milestone.

First and foremost, I would like to express my profound gratitude to my doctoral advisor, Cyrill Stachniss, for allowing me to pursue my Ph.D. under his guidance. His unwavering support has been invaluable to me from my first day until the completion of this thesis. He created an environment that enabled me to enhance my personal and academic skills, push my boundaries, and constantly strive for excellence. I could not have asked for a better supervisor.

I would also like to thank Lasse Klingbeil and Marc Hanheide for their thoughtful feedback and for dedicating their time to reviewing my thesis.

Looking back, the decision to pursue a Ph.D. was not always obvious to me, but the encouragement and mentorship I received at critical points paved the way for this journey. I owe a special thanks to Heinz Nühse for addressing my doubts about studying engineering and encouraging me to pursue this direction. I am equally grateful to Florian Pillath and Daniel–André Dücker for their guidance on how to work scientifically during my Bachelor's and Master's theses. Lastly, I want to thank Lasse Peters for our invaluable discussions during our time in Berkeley, which inspired me to deepen my knowledge of robotics.

During my Ph.D., I was fortunate to have outstanding friends and colleagues who guided and supported me throughout my research. Sharing an office with two of them was a unique experience that I truly value. I would like to sincerely thank Ignacio Vizzo, who taught me countless technical skills and offered fresh and inspiring perspectives on both academia and life. I am also grateful to Tiziano Guadagnino, who helped me rebuild my confidence in my academic work and provided me with valuable theoretical and practical knowledge about robotics. Both of you played a crucial role in shaping my educational, professional, and personal journey – grazie mille, i miei fratelli.

I want to express my heartfelt gratitude to Jens Behley for his extensive knowledge and support throughout these years. His contributions have resulted in a

continuous flow of ideas and suggestions for new projects and invaluable feedback on my scientific papers. I am also thankful for meeting Xieyuanli (Rhiney) Chen, the first Ph.D. student who genuinely engaged with my work and guided me through the process of completing a scientific paper. Both of you have inspired and motivated me, especially during the early stages of my Ph.D., a time when I was still defining my academic path.

Working in the lab has been a privilege, made even more enjoyable by my supportive colleagues. I deeply value our shared moments, from exciting discussions to casual chats over espresso in the kitchen. I want to thank Perrine Aguiar, Yue (Linn) Chong, Nived Chebrolu, Daniel Casado, Dhagash Desai, Saurabh Gupta, Ibrahim Hroob, Liren Jin, Birgit Klein, Haofei Kuang, Thomas Läbe, Hyungtae Lim, Luca Lobefaro, Meher Malladi, Federico Magistri, Rodrigo Marcuzzi, Elias Marks, Sumanth Nagulavancha, Lucas Nunes, Yue Pan, Thorbjörn Posewsky, Gianmarco Roggiolani, Julius Rückin, Vardeep Singh Sandhu, Matteo Sodano, Niklas Trekel, Jan Weyler, Louis Wiesmann, Matthias Zeller, and Xingguang (Starry) Zhong. Sharing this journey with you has been a joy.

Arriving in a new city during the pandemic presented unique challenges, but I was fortunate to be surrounded by new friends from day one. I am deeply grateful to the flatmates I had during these years, especially Christine Sander, Hannah Dönges, Katharina Blass, Kristin Pöllmann, Lukas Bruch, Marius Stork, and Welf Schröder, for creating a home that felt like family. Living with you provided fresh perspectives on many aspects of life, reminding me there is indeed life beyond work. I also owe a debt of gratitude to all my other friends from the Reuterstraße, whose unwavering support is too vast to fully express here. A special thanks to Mark Standke for our morning runs and endless discussions about new ideas and our respective Ph.D. adventures.

Being away from my hometown during my studies and Ph.D. has often made me feel like I'm living in two places simultaneously. I am deeply grateful to all my friends back home in Lippborg for their unwavering support, the invitations that helped me stay connected, and the visits to Bonn that always made me feel at home. I would like to extend a special thanks to Julius Horstkötter for our phone calls, during which we updated each other on our Ph.D.s progress and about our lives in general. Also, I would like to thank Lukas Möllmann for moving with me to the Rhineland and always having an open ear, no matter the topic.

Family is often considered the cornerstone of one's life, and I wholeheartedly agree. I owe everything to my parents, Claudia and Heiner, who have always supported me unconditionally in all my pursuits. Your belief in me has been a constant source of strength, and this thesis would not have been possible without your involvement. To my brother Jan, I cherish the endless memories of laughter, joy, and our discussions. I also want to thank my cousin Lukas for his constant

presence as my lifelong friend despite any distance between us. I am deeply thankful to my beloved grandparents, Albert, Helga, Josef, and Maria, for their profound influence on my life since childhood. Without you, I would not be the person I am today. Lastly, I want to thank Sabine and Robert for becoming my second family in Cologne over the past few years and showing love and interest in countless ways.

Finally, to Lena, your unwavering support and belief in me have been a constant source of motivation throughout this journey. Words cannot express my gratitude for your encouragement and companionship, which have brought balance and joy to my life during the most demanding times. Sharing this chapter of my life with you is an extraordinary gift for which I am deeply grateful.

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 European Union's HORIZON research and innovation programme under grant agreement No 101070405 (Digiforest), and by the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008 (Harmony).

# Contents

# Acronyms

**CNN** convolutional neural network

**IoU** intersection over union

**LiDAR** light detection and ranging

**LSTM** long short-term memory network

**MOS** moving object segmentation

**RMSE** root mean square error

**RNN** recurrent neural network

**SLAM** simultaneous localization and mapping

# Chapter 1

# Introduction

Mobile robots operate around us to solve tasks that humans do not want to carry out, cannot perform, or which are too dangerous or demanding for humans. These systems must ensure a safe operation by constantly perceiving and analyzing their environment. For example, robots need to know their current location and the state of their environment, including the behavior of other agents.

Nowadays, robotic platforms carry out tasks under specific safety measures in well-defined and well-known environments like manufacturing facilities or logistic warehouses, often at least partially under human supervision. However, to fully realize such systems' potential, they must also be able to operate autonomously in highly complex and dynamic environments and work next to humans or in changing environments. We illustrate the dynamic environments of such promising future directions in Fig. 1.1.

The situation in Fig. 1.1 (a) shows a so-called "ghost bike", which serves as a memorial for a location at which a bicyclist was killed during a traffic accident. Bicyclists are vulnerable road users who usually share the road with vehicles, putting them at high risk of driving mistakes. Advanced perception systems can identify such moving traffic participants and warn the driver or even actively avoid collisions, drastically increasing road safety for all its users. We show another dynamic environment example in Fig. 1.1 (b). Autonomous vehicles can make individual transportation accessible for everyone because human-operated vehicles are often not efficient enough to cover all demands, especially in rural areas. The task of autonomous driving in daily traffic is challenging because the vehicle has to navigate next to human-driven vehicles, bicyclists, and pedestrians, resulting in a highly unknown dynamic environment. Lastly, mobile robots like the platform in Fig. 1.1 (c) can assist us in tedious agricultural tasks like weeding control, growth stage monitoring, or harvesting to increase yield and, at the same time, a more sustainable crop production. Such an environment usually


(a) Road Safety

(b) Transportation

(c) Automation

Figure 1.1: Examples of how advanced perception systems can protect, transport, and support us in different scenarios. In all cases, perception systems must operate in unknown and dynamic environments to allow mobile robots to navigate and perform their intended tasks safely. Images taken by Craig F. Walker (Boston Globe), Hamburger Hochbahn AG, and Lincoln Centre for Autonomous Systems.

undergoes spatio-temporal changes due to, for example, changing vegetation or different seasons, which requires a system that is robust to dynamic environments. Nowadays, we also use mobile platforms to perform dangerous tasks in emergency response, such as extinguishing fires or carrying equipment. In the future, they could autonomously navigate in large buildings to seek the cause of a fire or secure hazardous chemical components. Such systems constantly face dynamic objects like humans and changing environments due to fire. Therefore, they need to recognize and adapt to these spatio-temporal changes to support emergency response efficiently.

To solve such tasks, we equip robots with proprioceptive and exteroceptive sensors. Accelerometers, gyroscopes, or wheel encoders are proprioceptive sensors that measure the robot's internal state. On the other hand, exteroceptive sensors like global navigation satellite systems, cameras, radars, and light detection and ranging (LiDAR) sensors acquire information from the robot's environment, and we can use them for both estimating the robot's state and the state of the en-

vironment. This thesis will primarily focus on perception tasks using sequential point clouds obtained from a LiDAR sensor. A LiDAR sensor is an active sensor that shoots laser beams into the environment while measuring the time until the beam returns to the sensor. From this time-of-flight information, we can compute the beam range. Given the azimuth and elevation angle at which the beam was directed, we obtain an Euclidean coordinate in the sensor's local frame. LiDAR sensors have shown a promising potential due to their precise 3D measurements compared to cameras and radars and their usability without passive light sources, for example, at night.

## 1.1 Main Research Questions

Given the challenges mentioned above, we can state two main questions these robotic systems should be able to answer: "What is moving?" and "Where is an object moving to?", which we visualize in Fig. 1.2.

### "What is moving?"

First, a mobile robot needs to know which objects in its environment are moving because these require further analysis to, for example, avoid collisions with moving objects as shown in Fig. 1.2. We phrase this in the question "What is moving?" and answer it in the first part of the thesis.

One way to infer moving objects in LiDAR data is to process multiple scans and assign a semantic class to each point, for example, buildings, pedestrians, or moving vehicles. Afterward, we can remap the predicted labels into moving and non-moving classes. We call this task multi-scan semantic segmentation, which has been studied under various names in the literature [100, 108, 161, 164, 176, 192, 204]. Another recent research topic is panoptic tracking or 4D panoptic segmentation, in which we not only segment the points into classes but also assign them to instances and track these over time [11, 95, 118, 119, 204, 217]. The methods above have the upside of additionally providing the semantic class of the object, which can be further used for traversability analysis in path planning [135]. However, the downside is that these approaches are often complex and cannot be directly transferred to new, unknown environments because of prior assumptions about the sensor or environment. Most approaches infer the output based on a parametric model, whose parameters are trained from labeled data. This procedure requires the availability of large datasets with labeled point clouds, which is a time-consuming and tedious task if done for many classes. Another challenge arises when deploying such systems in new, unseen environments, with potentially new and unknown object classes present

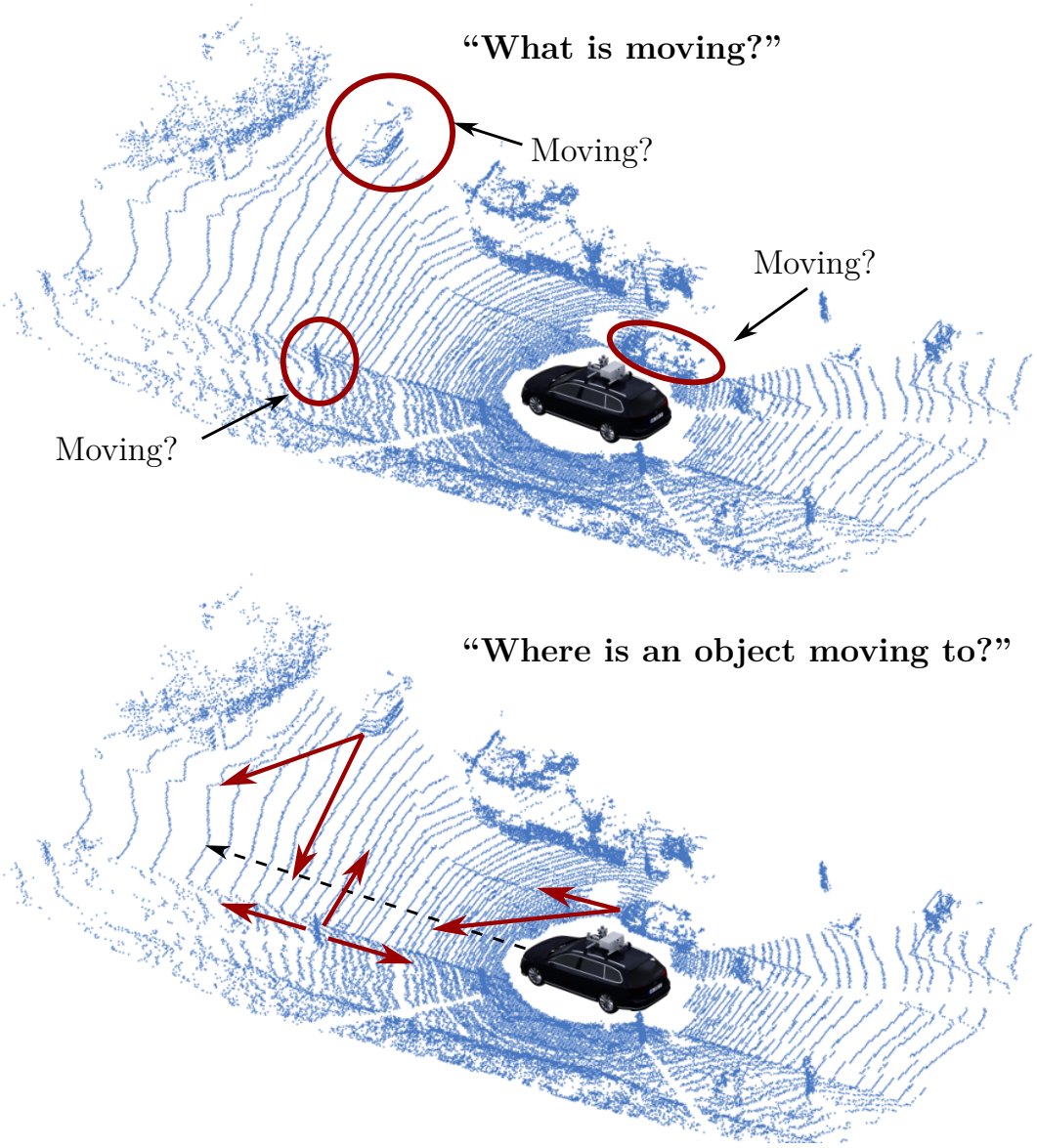



Figure 1.2: Two main questions robot systems must answer when operating in dynamic environments. We visualize a LiDAR point cloud in blue, measured from the black car. *Top:* The perception system must identify moving objects around the car independent of their semantic class. *Bottom:* For moving objects, estimating where they are moving to in the future is important to avoid collisions.

in the scene and a changed sensor setup. Lastly, it needs to be clarified if we need the full semantic information of objects for downstream tasks. For example, it is sufficient to know which LiDAR measurements belong to moving objects to obtain a static map of the environment for planning or obstacle avoidance.

In this thesis, we focus on the task of moving object segmentation (MOS) [28]. In MOS, we decide for each LiDAR point if it belongs to a moving or a non-moving object. We argue that for most downstream tasks like collision avoidance, knowing which semantic class the moving object is associated with is not critical.

This paradigm shift circumvents existing problems in semantic segmentation, like the underlying long-tailed distribution of the semantic classes, meaning that there are many classes with less frequent occurrence such that they are either underrepresented or not represented at all in the training data. For example, suppose that objects like a baby stroller or a bicycle with a trailer are not part of the training data. In that case, a semantic segmentation approach might not recognize them as one of the known moving classes and will not detect them correctly. In contrary, a MOS method can still identify them as moving points. Training models for MOS also requires labeled data of moving and non-moving points, but we can drastically reduce the complexity and the required number of labeled examples. We will also show in this thesis that these models can generalize well to new environments and sensor setups.

### "Where is an object moving to?"

After identifying moving objects, the robot can predict what will happen next and plan accordingly. Such estimation helps to decide if a planned path interferes with other objects' trajectories in the future. Therefore, the question we aim to answer in the second part of the thesis is "Where is an object moving to?".

In general, we have to look at sequential sensor data over time to predict the future state of the environment because more than a single snapshot of the environment is needed to reason about its temporal change. Over the past years, learning-based approaches have shown to be effective for natural language processing, in which a sequence of text is processed to, for example, predict the next most likely word token [141]. This research field inspired other researchers to predict completely different future states, such as the future trajectories of traffic participants [5, 43, 67]. Most researchers explored architectures like recurrent neural network (RNN) [48, 83], long short-term memory network (LSTM) [72], or transformers [184] to process spatio-temporal data. These systems can become rather complex in their design and training. One reason is that the spatial and temporal information is often processed sequentially. Instead, this thesis focuses on convolution-based architectures for prediction because of their simplicity and effectiveness for robotic applications. For example, one advantage is the joint processing of spatial and temporal information, which makes the training and testing of our architectures faster.

We must address several challenges to successfully deploy a prediction model in a new and unseen environment. Similar to the perception tasks mentioned above, these models are often trained based on labeled data, and the training data can differ from the measurements in a new environment. Consequently, the prediction models must generalize well to new and unseen environments. Besides that, some prediction models take the trajectories of neighboring objects

as inputs, which requires previous detection and tracking of objects. Therefore, these preceding perception steps must also work reliably in the new environment and deal with missed detections. Another challenge of supervised prediction is that there is no possibility of evaluating the performance of a model during runtime because the ground truth trajectories are not available. Consequently, it is unclear if we can trust our prediction model in a new environment, making it hard to integrate its output into downstream planning tasks safely. One way to tackle these challenges is to directly predict what the sensor will perceive next instead of an intermediate representation like tracked trajectories. Since this information will always be available, we can train these approaches online in new environments. At the same time, this allows us to evaluate the performance of our prediction model because we can compare its previously predicted output with the actual new measurements. Self-supervised prediction is a promising new research direction that we will explore in the last chapter of this thesis.

## 1.2 Main Contributions

In this thesis, we will tackle fundamental questions that must be answered for a robot to operate in dynamic environments: "What is moving?" and "Where is an object moving to?". This section summarizes the main contributions of the thesis to the spatio-temporal perception of mobile robots.

The first contribution is an approach to segment moving objects in a sequence of LiDAR scans, which we present in Chap. 3 and call 4DMOS. Our underlying assumption is that we can infer the motion of an object from a limited number of consecutive scans. We propose to rely solely on the information about which parts of the environment are occupied at a particular time and represent the sequence of LiDAR measurements as a sparse 4D tensor. We use sparse convolutions to jointly extract spatio-temporal features and predict a moving object confidence score for each point representing the likelihood of belonging to a moving object. To further increase the robustness towards false positive and negative predictions, we propose a receding horizon strategy that allows us to fuse multiple predictions for the same point in a static state binary Bayes filter. We experimentally demonstrate the effectiveness of our approach and show that it generalizes well to new, unseen environments because, in contrast to existing methods, we do not exploit any environment- or sensor-specific information.

The second contribution is a map-based approach to segment moving objects, which we refer to as MapMOS and describe in Chap. 4. Integrating all measurements into a local map and using the map for the segmentation has advantages compared to the sequence-based approach in 4DMOS, for example, when an object is partially occluded within the sequence. Also, some sensors like scanners by

Livox have an irregular sensing pattern, which breaks the assumption of 4DMOS that the moving object is always visible within the sequence. In addition, we propose maintaining a volumetric belief about which parts of the environment can contain moving objects. We show how we can use the belief to improve the segmentation performance and obtain a static map of the environment, which we can exploit in downstream tasks like planning.

We demonstrate in both, Chap. 3 and Chap. 4, that generalization to new sensors and environments is a crucial capability of modern spatio-temporal perception systems and that a lot of state-of-the-art approaches fail on this. To further test and evaluate these generalization capabilities properly, we propose a public moving object segmentation benchmark with heterogeneous sensors in Chap. 5. The dataset contains moving object labels for four different LiDAR sensors with varying fields of view and sensor patterns that capture the scene simultaneously for the training, validation, and testing of MOS approaches. We demonstrate the generalization capabilities of our proposed methods and test how different training setups affect the performance when evaluating the estimation from data obtained with another sensor type.

To close the first part, which answers the "What is moving?" question, we show how we can use our architecture to segment stable points in Chap. 6. This task aims at segmenting not only currently moving points but generally unstable points for localization in a pre-built map. We consider points unstable if, for example, they belong to vegetation or parked cars that have changed between the initial mapping and the later localization session. Our fourth contribution is a system that filters out unstable points and can improve localization in changing environments, such as an orchard or a parking lot.

Part II of this thesis answers the "Where is an object moving to?" question, which is essential for path planning in dynamic environments to, for example, avoid collisions. The fifth contribution of this thesis is an approach that predicts the maneuver and trajectory of a vehicle in structured environments like highways. We propose to encode the past information of neighboring vehicles over time in an efficient, dense tensor and use spatio-temporal convolutions to classify the maneuver and then regress the corresponding trajectory of a vehicle. We train the approach in a supervised fashion and successfully predict trajectories close to the recorded ground truth.

Lastly, the sixth contribution in Chap. 8 is a method that can predict a future state of the environment solely from a stream of measured point clouds by predicting a sequence of future point clouds. The main advantage is that no preceding perception steps like detection and tracking are required. The approach can be trained and evaluated online without any manually provided labels by comparing how close the measured scan is to the previously predicted point cloud.

Overall, this thesis offers six contributions to spatio-temporal perception in dynamic environments. We published all presented approaches in peer-reviewed conference papers and journal articles and experimentally tested them on real-world data. All methods can operate at the sensor frame rate given adequate hardware.

## 1.3 Publications

All parts of this thesis have been published in peer-reviewed conference papers and journal articles:

- B. Mersch*, T. Höllen*, K. Zhao, C. Stachniss, and R. Roscher. Maneuver-based Trajectory Prediction for Self-driving Cars Using Spatio-temporal Convolutional Networks. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021. DOI: 10.1109/IROS51168.2021.9636875. (* authors contributed equally)

- 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. DOI: 10.48550/arXiv.2110.04076

- 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. DOI: 10.1109/LRA.2022.3183245

- B. Mersch, T. Guadagnino, X. Chen, Tiziano, I. Vizzo, J. Behley, and C. Stachniss. Building Volumetric Beliefs for Dynamic Environments Exploiting Map-Based Moving Object Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 8(8):5180–5187, 2023. DOI: 10.1109/LRA.2023.3292583

- I. Hroob*, B. Mersch*, C. Stachniss, and M. Hanheide. Generalizable Stable Points Segmentation for 3D LiDAR Scan-to-Map Long-Term Localization. *IEEE Robotics and Automation Letters (RA-L)*, 9(4):3546–3553, 2024. DOI: 10.1109/LRA.2024.3368236. (* authors contributed equally)

- H. Lim, S. Jang, B. Mersch, J. Behley, H. Myung, and C. Stachniss. HeLiMOS: A Dataset for Moving Object Segmentation in 3D Point Clouds From Heterogeneous LiDAR Sensors. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2024. DOI: 10.1109/IROS58592.2024.10801938

## 1.4 Further Scientific Contributions

During my doctorate, I was also involved as a co-author in the following peer-reviewed conference and journal publications which are not part of the thesis:

- 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(4):6529–6536, 2021. DOI: 10.1109/LRA.2021.3093567

- 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. DOI: 10.1109/LRA.2022.3166544

- I. Vizzo, B. Mersch, R. Marcuzzi, L. Wiesmann, J. Behley, and C. Stachniss. Make It Dense: Self-Supervised Geometric Scan Completion of Sparse 3D Lidar Scans in Large Outdoor Environments. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):8534–8541, 2022. DOI: 10.1109/LRA.2022.3187255

- I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023. DOI: 10.1109/LRA.2023.3236571

- M. Zeller, V. Sandhu, B. Mersch, J. Behley, M. Heidingsfeld, and C. Stachniss. Radar Velocity Transformer: Single-scan Moving Object Segmentation in Noisy Radar Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023. DOI: 10.1109/ICRA48891.2023.10161152

- H. Lim, L. Nunes, B. Mersch, X. Chen, J. Behley, H. Myung, and C. Stachniss. ERASOR2: Instance-Aware Robust 3D Mapping of the Static World in Dynamic Scenes. In *Proc. of Robotics: Science and Systems (RSS)*, 2023. DOI:10.15607/RSS.2023.XIX.067

- I. Vizzo, B. Mersch, L. Nunes, L. Wiesmann, T. Guadagnino, and C. Stachniss. Toward Reproducible Version-Controlled Perception Platforms: Embracing Simplicity in Autonomous Vehicle Dataset Acquisition. In *Worshop on Building Reliable Ratasets for Autonomous*

*Vehicles, IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, 2023. DOI: 10.1109/ITSC57777.2023.10421988

- M. Zeller, V. Sandhu, B. Mersch, J. Behley, M. Heidingsfeld, and C. Stachniss. Radar Velocity Transformer: Single-scan Moving Object Segmentation in Noisy Radar Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023. DOI: 10.1109/ICRA48891.2023.10161152

- S. Gupta, T. Guadagnino, B. Mersch, I. Vizzo, and C. Stachniss. Effectively Detecting Loop Closures using Point Cloud Density Maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024. DOI: 10.1109/ICRA57147.2024.10610962

- L. Nunes, R. Marcuzzi, B. Mersch, J. Behley, and C. Stachniss. Scaling Diffusion Models to Real-World 3D LiDAR Scene Completion. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. DOI: 10.1109/CVPR52733.2024.01399

- T. Guadagnino*, B. Mersch*, I. Vizzo*, S. Gupta, M. Malladi, L. Lobefaro, G. Doisy, and C. Stachniss. Kinematic-ICP: Enhancing LiDAR Odometry with Kinematic Constraints for Wheeled Mobile Robots Moving on Planar Surfaces. Accepted for *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2025. DOI: 10.48550/arXiv.2410.10277. (* authors contributed equally).

## 1.5 Open Source Contributions

During my doctorate, I published multiple open-source implementations of the presented approaches and contributed to the following open-source projects:

- **Point-cloud-prediction**: Self-supervised point cloud prediction
  `https://github.com/PRBonn/point-cloud-prediction`

- **4DMOS**: Sequence-based moving object segmentation
  `https://github.com/PRBonn/4DMOS`

- **KISS-ICP**: LiDAR odometry pipeline for simple, accurate, and robust registration
  `https://github.com/PRBonn/kiss-icp`

- **MapMOS**: Map-based moving object segmentation
  `https://github.com/PRBonn/4DMOS`

- **Meta-workspace**: Workspace for reproducible version-controlled perception platforms
  `https://github.com/ipb-car/meta-workspace`
- **Kinematic-ICP**: LiDAR odometry pipeline for wheeled mobile robots
  `https://github.com/ipb-car/kinematic-icp`
- **KISS-SLAM**: LiDAR SLAM pipeline
  `https://github.com/ipb-car/kiss-slam`

# Chapter 2

# Basic Techniques

In this chapter, we review basic techniques that we use throughout the remaining chapters and which are essential to understanding the main contributions of this thesis. Spatio-temporal perception generally refers to estimating a state given measured data over time. This data can be, for example, a sequence of point clouds a mobile robot is collecting using its LiDAR scanner. Often, such spatio-temporal data is extensive and requires advanced processing, which we will present in this chapter.

We can use this data to identify moving objects or predict where they go. Due to the complexity of the problems presented, solving these purely based on pre-defined rules and heuristics is often challenging. For example, LiDAR scans can be noisy, incomplete, or corrupted by reflections, which makes it hard to extract meaningful features reliably. Therefore, a common approach is to define a parametric model that maps input data to the desired output and to learn the mapping parameters from the data. Classic machine learning algorithms still rely on hand-designed features. In contrast, deep learning algorithms operate on raw data without manual steps beforehand and usually consist of multiple hidden layers that extract highly abstract features. We refer the reader to the books by Goodfellow et al. [61] or Prince [147] for a detailed overview of deep learning.

The most simple and well-known deep learning architecture is a multi-layer perceptron, which consists of an input, output, and multiple intermediate hidden layers. Each layer describes a fixed non-linear mapping from input to output features. We optimize the parameters of this mapping by comparing the model's output with a known reference. For example, this can be the cross-entropy loss of a predicted likelihood for the ground truth class in classification. We usually solve the optimization problem using stochastic gradient descent or a more sophisticated variant and backpropagate the gradients with respect to the parameters. We will not go into the details of deep neural networks and how to train them because these techniques are well-presented in the existing literature [61].

In the case of spatio-temporal data, the main challenge lies in processing data of variable size while extracting both spatial and temporal relationships. Multi-layer perceptrons operate on fixed-size input vectors and cannot distinguish higher-dimensional data. Instead, we will focus in Sec. 2.1 on a different type of neural network, namely convolutional neural networks (CNNs). These build the foundation for all architectures used in this thesis. Initially developed for 2D images, CNNs have been further extended to work with higher dimensional data such as 3D tensors, see Sec. 2.2. To process a sequence of 3D LiDAR scans for segmentation, we will consider in Sec. 2.3 multiple 3D tensors over time, resulting in a spatio-temporal 4D convolution. Since LiDAR point clouds are naturally sparse compared to, for example, a dense camera image, we will show in Sec. 2.4 how sparse convolutions reduce the memory footprint and speed up the runtime of the convolutions for point cloud data.

## 2.1 2D Convolutional Neural Networks

In computer vision, convolutions are an operation for applying a hand-designed filter to an image. Such filters usually operate locally on a neighborhood of pixels and are used, for example, for de-noising, smoothing, or segmentation. The filter is represented by a kernel function that takes neighboring pixel values as input and outputs a new value. The convolution defines how this filter slides along the image axis to obtain a new image of feature values. The kernel function usually has fixed parameters that are chosen based on heuristics. For example, the Sobel operator or Sobel filter approximates the gradient of an image's intensity function and is commonly used to detect edges in images. We refer the reader to the book by Förstner and Wrobel [55] or Szeliski et al. [174] for more examples. Instead of finding fixed filters for different tasks, researchers focused on learning the filter parameters from data. This allowed researchers to find representations more suitable for such tasks, for example, identifying curves in handwritten digit recognition [2, 64].

With the advancement of deep learning for computer vision tasks like digit recognition [2] or object detection [59], CNNs are an efficient and well-performing framework to extract information from images. Like the abovementioned image convolutions, CNNs apply a mathematical operation on a local neighborhood. These operations usually consist of a linear mapping of the pixel-wise features with the kernel parameters and then passing it through a non-linear activation function. We illustrate two convolutional steps in Fig. 2.1. In this simple example, our image has a single channel resulting in matrix $\boldsymbol{I}$, which we can index with discrete pixel coordinates $u$ and $v$ for its value $\boldsymbol{I}_{u,v}$. The kernel is a matrix of size $3 \times 3$ with learnable parameters $w_{i,j} \in \mathbb{R}$. Suppose the indexed pixel exceeds

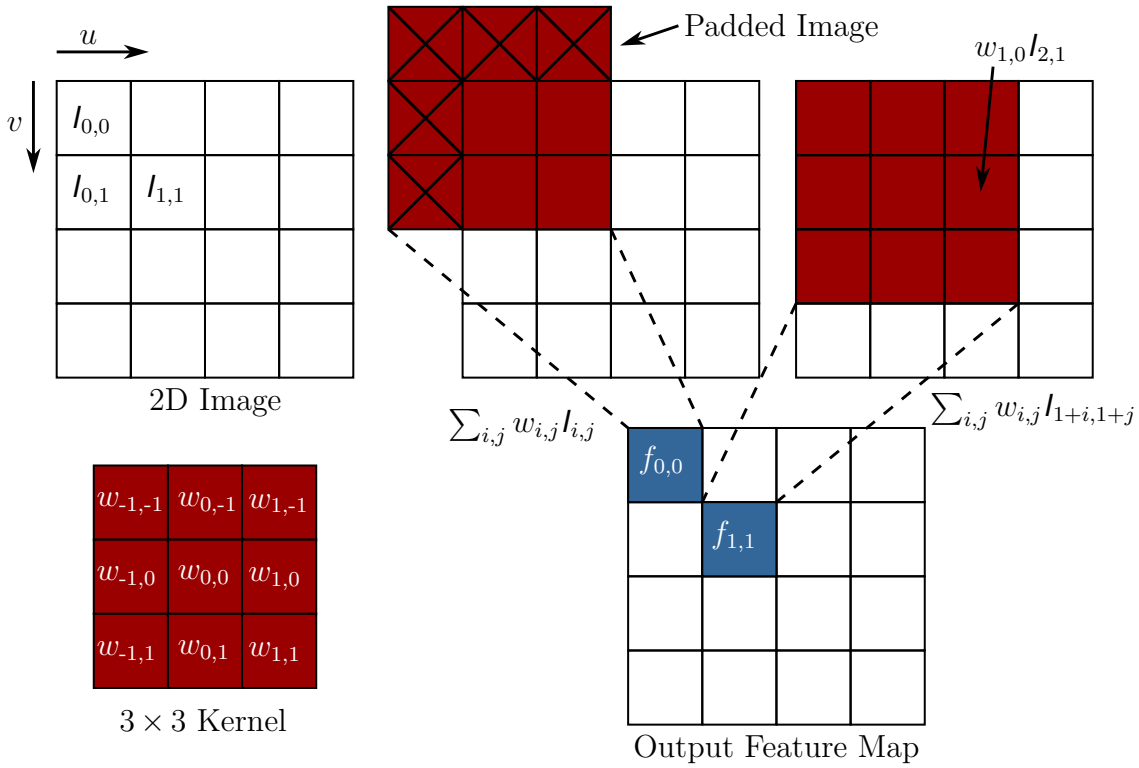



Figure 2.1: Two examples for the feature computation in a 2D convolution.

the image coordinates and no channel information is available. In that case, using a padding strategy like zero-padding is common, which sets the channel value to zero. If we do not pad the image, we cannot compute the output feature and, therefore, reduce the size of the output feature map by one along each dimension. For a location $[u, v]^\mathsf{T} \in \mathbb{Z}^2$ of the output feature image, we can compute its value $f_{u,v} \in \mathbb{R}$ by

$$f_{u,v} = \sum_{\left[\begin{smallmatrix} i \\ j \end{smallmatrix}\right] \in \mathcal{V}_K^2} w_{i,j} I_{u+i,v+j}, \tag{2.1}$$

where $\mathcal{V}_K^2$ denotes the set of offsets which represent the neighborhood of the location $[u, v]^\mathsf{T}$ given by a kernel of size $K$. In this example with a kernel of size 3, this results in 9 offsets of

$$\mathcal{V}_3^2 = \left\{ \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}. \tag{2.2}$$

Passing the resulting feature map through non-linear activation functions and normalization layers is standard to ensure a zero-mean and unit-variance Gaussian distribution of the resulting non-linear features. Also, the feature maps are often downsampled or upsampled to decrease or increase their resolution to manage the memory footprint and the level of detail the features describe. We can achieve downsampling by applying strided convolutions. Usually, we apply the

convolutional filter with a stride of one across the entire image, meaning that we move the filter at most by one pixel across each dimension of the feature map. A stride of two moves the filter by two pixels, effectively skipping half of the convolutions. The larger stride leads to a faster computation of the convolution and an output that has a smaller size compared to the input. An alternative to strided convolutions are pooling layers, which reduce the size of the feature map using different operations like max pooling, average pooling, or global pooling.

By downsampling, we also increase the receptive field of the output features, meaning that we compute each feature from a larger part of the input feature map. This property is essential to processing large objects in an image by ensuring that the output feature "sees" enough of the object in the input image.

It is common to use transpose convolutions to upsample a feature map. These increase the size of the output map by applying the kernel to each input feature individually, effectively resulting in multiple sub-maps with the kernel size that we can combine to create a larger output map.

We call this collection of layers a convolutional block. Many variations and strategies exist for layering such architectures in practice, but most consist of similar building blocks. Finally, we obtain a CNN by stacking multiple convolutional blocks, resulting in a deep architecture with an often large number of trainable parameters.

Using local image operators for feature extraction has two main advantages compared to the traditional multi-layer perceptrons: First, they ensure local connectivity of the output features because they are not connected to all input features but just the local neighborhood. Second, due to the translation equivariance of the local operator, the parameters of a kernel are shared, meaning that we use the same kernel across the whole image, effectively reducing the total number of trainable parameters.

## 2.2 Higher Dimensional Convolutions

We can generalize the convolution in Eq. (2.1) to $D$-dimensional spaces. To compute an output feature $\boldsymbol{f}_{\boldsymbol{u}}^{\text{out}} \in \mathbb{R}^{N_{\text{out}}}$ at a discrete coordinate $\boldsymbol{u} \in \mathbb{Z}^D$, we apply a convolution kernel $\boldsymbol{W} \in \mathbb{R}^{K^D \times N_{\text{out}} \times N_{\text{in}}}$ on the input features of size $N_{\text{in}}$ in the local neighborhood defined by the kernel with kernel size $K$. More specifically, we can split the kernel into $K^D$ matrices, resulting in a matrix $\boldsymbol{W}_{\boldsymbol{i}} \in \mathbb{R}^{N_{\text{out}} \times N_{\text{in}}}$ for each offset $\boldsymbol{i} \in \mathbb{Z}^D$ given by the kernel. Doing so, we express the convolution by summing over the products of each kernel matrix $\boldsymbol{W}_{\boldsymbol{i}}$ with the corresponding input feature $\boldsymbol{f}_{\boldsymbol{u}+\boldsymbol{i}}^{\text{in}} \in \mathbb{R}^{N_{\text{in}}}$ at location $\boldsymbol{u}+\boldsymbol{i}$. This results in the following higher-

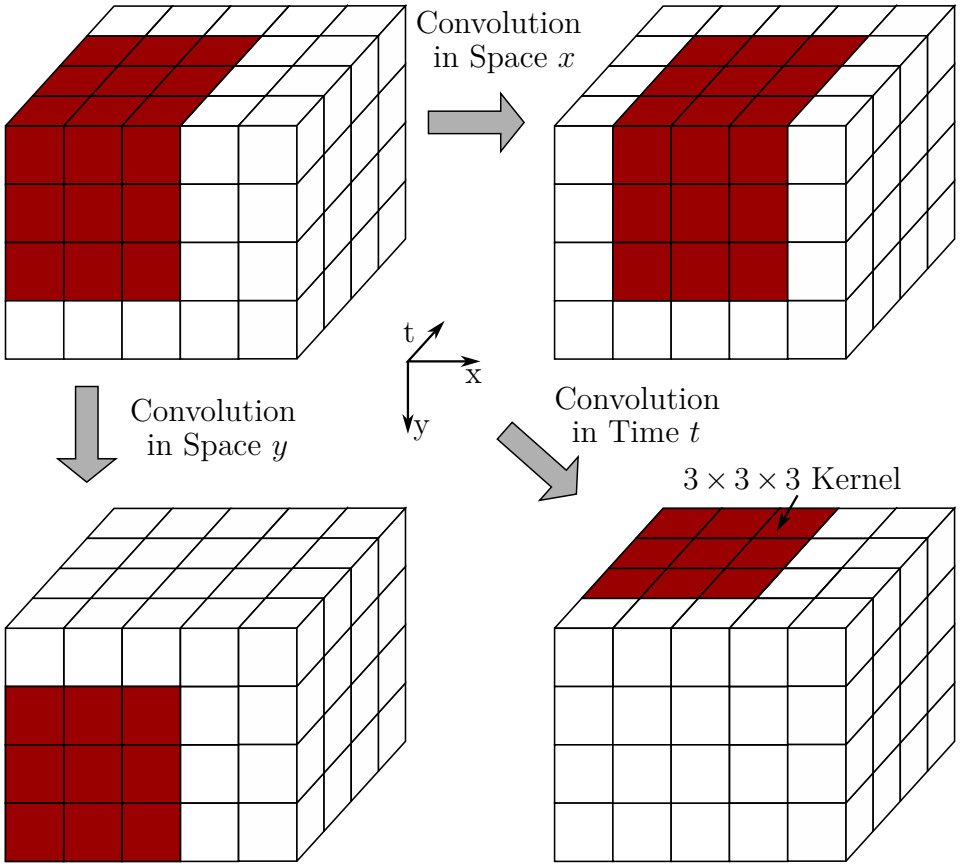



Figure 2.2: Example of how we apply a 3D convolutional filter along the spatial axes $x$ and $y$ and a temporal axis $t$.

dimensional convolution for the output feature $\boldsymbol{f}_{\boldsymbol{u}}^{\text{out}} \in \mathbb{R}^{N_{\text{out}}}$ at location $\boldsymbol{u} \in \mathbb{Z}^D$

$$\boldsymbol{f}_{\boldsymbol{u}}^{\text{out}} = \sum_{\boldsymbol{i} \in \mathcal{V}_K^D} W_{\boldsymbol{i}} \boldsymbol{f}_{\boldsymbol{u}+\boldsymbol{i}}^{\text{in}}, \tag{2.3}$$

in which $\mathcal{V}_K^D$ denotes the set of $D$-dimensional offsets the kernel of size $K$ is operating on with $\left|\mathcal{V}_K^D\right| = K^D$. Note that the result of Eq. (2.3) is usually followed by a non-linear activation function, which we omit for better readability.

The $D$-dimensional convolutions allow us to extract features not solely from images but also from higher dimensional data. We will exploit this in Chap. 7 and Chap. 8 to apply 2D and 3D convolutions over spatial dimensions but also across time. We visualize such a 3D convolution in Fig. 2.2 in which we shift the kernel along all three coordinate axes. These higher dimensional convolutions allow researchers to extract spatio-temporal features within a single architecture and use them to do various spatio-temporal perception tasks. Temporal convolutions across time are intuitive because the common assumptions about spatial convolutions still hold: First, an output feature at a given time index should mainly depend on the inputs that are close in time. Second, the feature computation should be equivariant to the absolute time since we apply the same

convolutional kernel along the time axis. In the next section, we give an example of extending higher dimensional convolutions to sequences of 3D point clouds.

## 2.3 Spatio-Temporal 4D Convolutions

Spatio-temporal perception often involves 3D data from LiDAR scanners. Today, sensors commonly output up to 2.6 M points per second, processing a large amount of data at the sensor frame rate. In the first part of this thesis, we will extract spatio-temporal features from such a stream of data to segment the points into, for example, moving and non-moving points.

Formally, we process a point cloud sequence $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_M\}$ of $M$ LiDAR scans $\mathcal{P}_t = \{\boldsymbol{p}_{t,1}, \boldsymbol{p}_{t,2}, \dots, \boldsymbol{p}_{t,N_t}\}$ with $N_t$ points $\boldsymbol{p}_{t,i} \in \mathbb{R}^3$. We can consider the collection of points as a 4D point cloud by extending the point coordinates with the scan's timestamp, resulting in $\hat{\boldsymbol{p}}_{t,i} = [x_i, y_i, z_i, t]^\top \in \mathbb{R}^4$. For simplicity, we can omit the subscript $t$ because we can infer the point cloud index from the time coordinate of the point. To apply our convolutional framework, we must represent the data in a regular grid to obtain a notion of the neighborhood and define the convolutions. To do this, we voxelize the 4D point cloud using a given spatial voxel size $\Delta s \in \mathbb{R}$ and temporal resolution $\Delta t \in \mathbb{R}$ between scans, resulting in a discrete coordinate $\boldsymbol{u} \in \mathbb{Z}^4$

$$\boldsymbol{u} = \left[ \left\lfloor \frac{x}{\Delta s} \right\rfloor, \left\lfloor \frac{y}{\Delta s} \right\rfloor, \left\lfloor \frac{z}{\Delta s} \right\rfloor, \left\lfloor \frac{t}{\Delta t} \right\rfloor \right]^\top, \tag{2.4}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation. A voxel can encompass multiple points that fall into it, but we always keep track of the original points. We additionally store features in the voxel that contain information about the measurement, like intensity, or use a constant value to represent the occupancy of the grid cell.

We can now define 4D convolutions on the resulting voxel grid based on the higher dimensional convolution introduced in Sec. 2.2 by choosing an input dimension of $D = 4$. We visualize such a spatio-temporal 4D convolution in Fig. 2.3. Note that in this simplified example, we only show a single kernel of size $3 \times 3 \times 3 \times 3$ and without padding, which results in a smaller output tensor. One can see that the feature of a voxel at time $t$ considers inputs from all neighboring voxels at the same points in time but also one timestamp before and after. We will offset the kernel in both the spatial and temporal directions for the following computations.

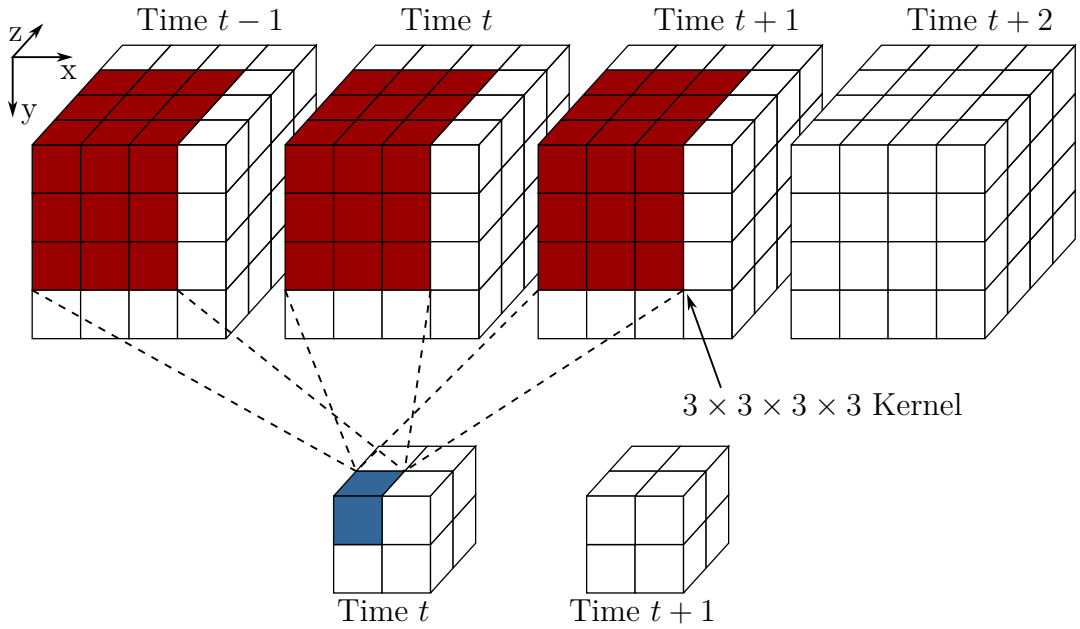



Figure 2.3: Schema of a Spatio-Temporal 4D convolution. Note that the cubes represent the same spatial voxel grid at different points in time.

## 2.4 Sparse Convolutions

The spatio-temporal 4D convolutions presented in Sec. 2.3 do not scale well for large environments because the size of the input volume grows cubically with the spatial and linear with the temporal dimension. Assuming a voxel size of 10 cm, a sequence of 10 scans with a radius of 100 m, this can result in the worst case in an input tensor with $\frac{200}{0.1}^3 \cdot 10 = 8 \cdot 10^{10}$ voxels over which the convolutional kernels will slide to extract spatio-temporal features. However, LiDAR scans are sparse by nature, meaning that the free space causes a lot of empty voxels, and the result of the convolutions will be zero most of the time. We will exploit this by representing the voxel grid as a sparse tensor and using a sparse convolutional network that can operate on sparse grids.

In existing literature, sparse convolutional networks are not uniquely defined. Liu et al. [107] proposed to prune parameters in a CNN to achieve a sparse architecture. However, the convolutions are still defined on the dense inputs. The definition of convolutions operating directly on sparse data has been further investigated by Graham et al. [63] and Choy et al. [32]. In this thesis, we follow the terminology from Choy et al. [32], who also provide an implementation, the MinkowskiEngine. This implementation allows the computation of the output features only for non-empty regions by applying the convolutional kernels to the non-empty inputs.

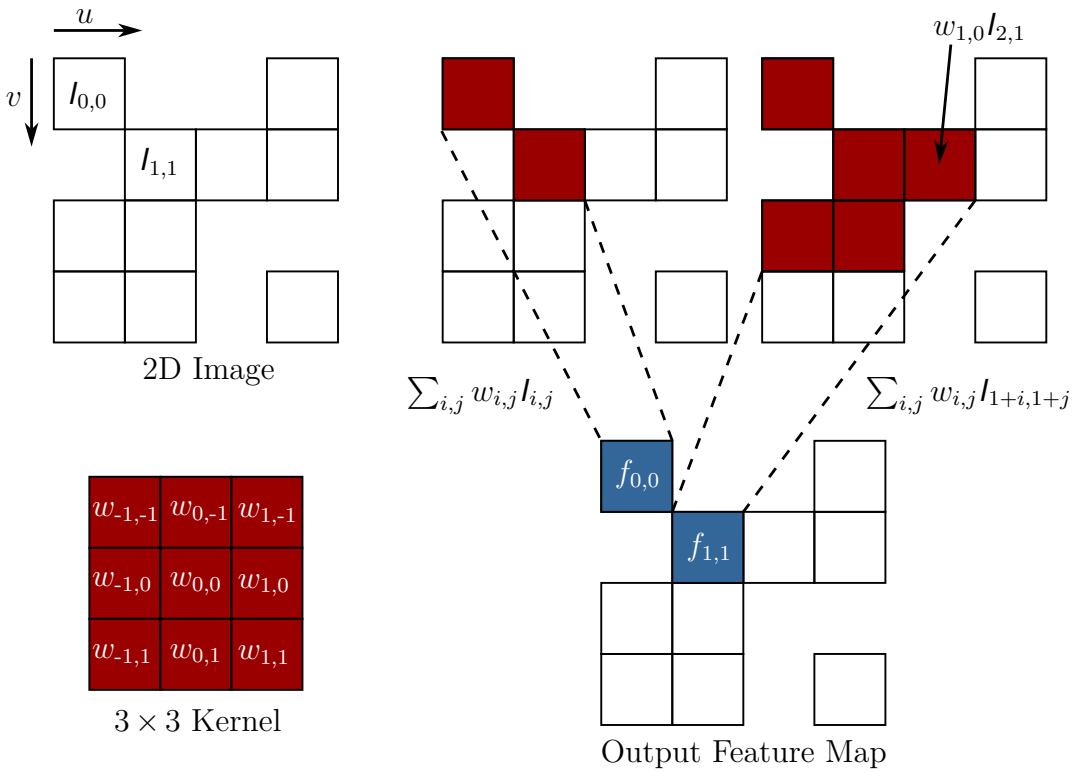



Figure 2.4: Schema of a sparse convolution. In contrast to the dense convolution, we only compute output features for non-empty voxels and only from non-empty inputs.

### 2.4.1 Sparse Tensors

A sparse tensor is defined as the non-empty coordinates of a voxel grid and its corresponding features. For our 4D example from Sec. 2.3, we can express the resulting 4D tensor as a sparse tensor with $N$ coordinates $\boldsymbol{C}$ and features $\boldsymbol{F}$ using the notation introduced by Choy et al. [32] reading

$$\boldsymbol{C} = \begin{bmatrix} x_1 & y_1 & z_1 & t_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & z_N & t_N \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \boldsymbol{f}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{f}_N^\mathsf{T} \end{bmatrix}, \tag{2.5}$$

where $x_i, y_i, z_i, t_i$ are the discretized coordinates from Eq. (2.4). In contrast to the dense convolutions in Sec. 2.2, sparse convolutions are defined by not only the kernel but also the input and output coordinates because not all neighbors are present in the case of a sparse voxel grid. The so-called generalized convolution [32] encompasses this and is given by

$$\boldsymbol{f}_{\boldsymbol{u}}^{\text{out}} = \sum_{\boldsymbol{i} \in \left\{ \mathcal{V}_K^D \,\middle|\, \boldsymbol{u}+\boldsymbol{i} \in \boldsymbol{C}^{\text{in}} \right\}} \boldsymbol{W}_{\boldsymbol{i}} \boldsymbol{f}_{\boldsymbol{u}+\boldsymbol{i}}^{\text{in}} \quad \text{for } \boldsymbol{u} \in \boldsymbol{C}^{\text{out}}, \tag{2.6}$$

where $\mathcal{V}_K^D$ again defines the offsets given by the kernel, but this time, we apply this set to the coordinate $\boldsymbol{u}$ and intersect it with the set of non-empty input

coordinates $C^{\text{in}}$. In doing so, we only consider non-empty voxels for the feature computation and only compute output features for non-empty voxels defined by coordinates in $C^{\text{out}}$. We show a sparse example of the previously visualized 2D convolution in Fig. 2.4.

### 2.4.2 MinkUNet

In the remainder of the thesis, we will use the MinkowskiEngine for sparse convolutions [32]. Specifically, we base our sparse 4D architecture on the MinkowskiUNet models developed by Choy et al. [32]. This architecture implements widely used concepts like residual blocks [70] and a U-shaped architecture with skip connections [152] to achieve sufficient compression and de-compression while maintaining a high level of details during the feature extraction. We build our architecture based on the MinkUNet32, which we visualize in Fig. 2.5.

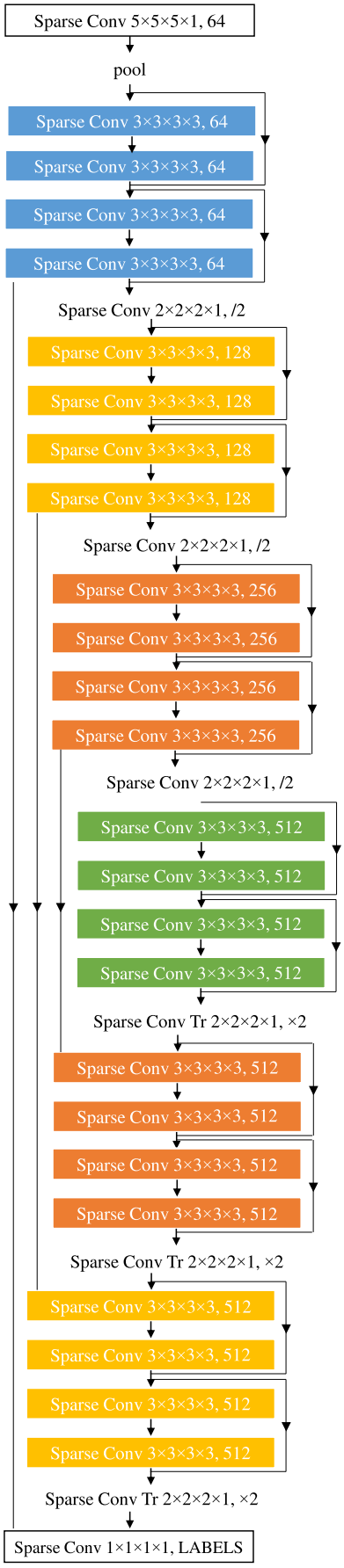



Figure 2.5: Architecture of a MinkowskiUNet32. Originally taken from [32].

# Part I

# Spatio-Temporal Segmentation of LiDAR Data

# Chapter 3

# Sequence-Based Moving Object Segmentation

A key challenge for autonomous vehicles is to navigate in unseen dynamic environments, which usually raises the question of "What is moving?". Distinguishing moving from static objects in 3D LiDAR data is a crucial task for autonomous systems and essential to, for example, plan collision-free trajectories [177], or understand the future behavior of other traffic participants [182, 198]. In LiDAR MOS, we aim to identify which points in a sequence of LiDAR scans belong to moving objects. Afterward, one can use the knowledge about moving objects to further improve localization [28, 30], mapping [28], or scene flow estimation [14, 60, 181].

Various approaches exist for integrating the knowledge of moving objects into state estimation problems. For visual simultaneous localization and mapping (SLAM), some feature-based approaches remove dynamic objects to avoid wrong correspondences [167, 208, 210]. On the contrary, Henein et al. [71] use an instance-level object segmentation to identify potentially moving points and explicitly track their motion as rigid bodies in a factor graph optimization framework. Recently, the Khronos [158] framework was proposed to maintain a spatio-temporal representation of the environment while integrating short-term dynamics and long-term structural changes.

For LiDAR SLAM, Pfreundschuh et al. [142] as well as Chen et al. [28, 30] demonstrate the effectiveness of performing MOS in LiDAR data for improving the quality of data associations. Moving objects also affect mapping for localization and long-term planning because they remain as so-called "ghost artifacts" in the map. Static mapping approaches [9, 10, 90, 104] alleviate this by first building a map and then cleaning it from traces of dynamic objects in a post-processing step. Thus, the addressed estimation problem has multiple relevant applications in robotics.

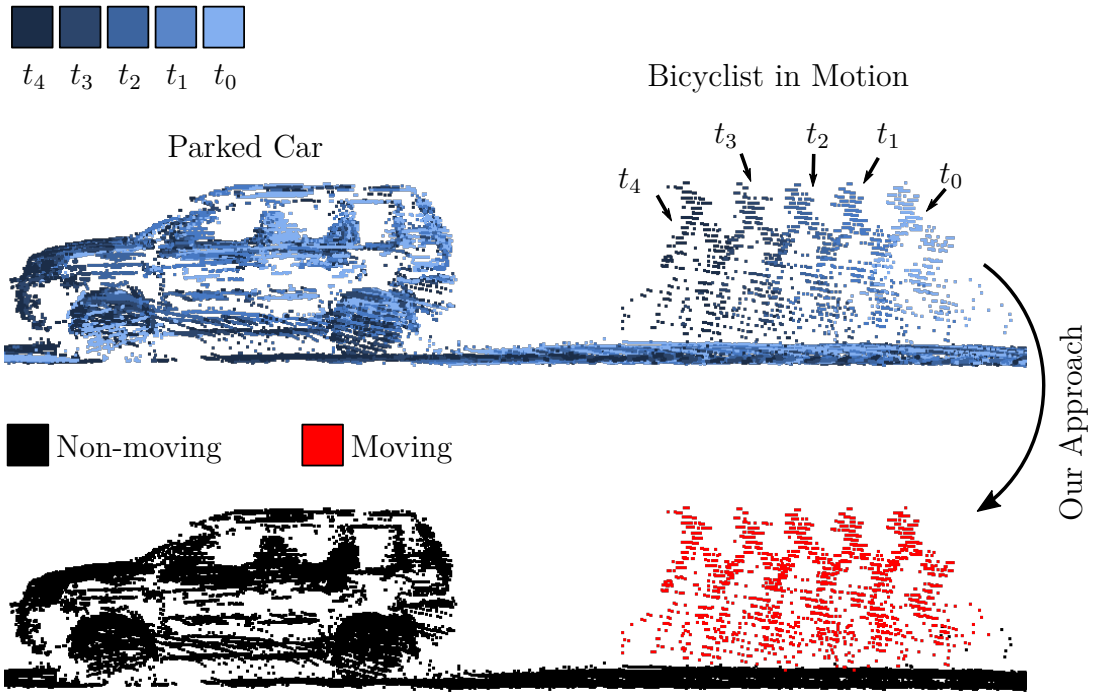



Figure 3.1: Given a sequence of point clouds, our approach identifies that the points belonging to the bicyclist move in space over time. *Top*: Input sequence with points colorized (blue) with respect to the timestamp. The darker the blue color, the newer the scan. *Bottom*: Our method successfully predicts the bicyclist as "moving" (red) and the parked car as "non-moving" (black).

There exist mapping approaches that identify if observed points are *potentially moving* or *have moved* throughout the mapping process [9, 30, 75, 144]. In this chapter, we identify objects that are *moving* within a short time horizon because these are the most interesting for online navigation. In contrast to the semantic segmentation task, MOS in 3D LiDAR data does not require a complex notion of semantic classes with extensive labeling to supervise learning-based methods or evaluate their performance. Instead, the goal is to predict whether points move or remain static throughout space and time. We visualize an exemplary of a parked car and a moving bicyclist scenario in Fig. 3.1. In general, the task requires extracting temporal information from the LiDAR sequence to decide which points are moving and which are not. Previous works tackled this problem by extracting temporal information from residual range images [28] or bird's eye view images [126], typically using a 2D CNN. The back-projection from these 2D representations to the 3D space often requires post-processing like k-nearest neighbor clustering [28, 35, 50, 125] to avoid labels bleeding into points that are close in the image space but distant in 3D. Other approaches can identify objects that have moved in 3D space directly during mapping [9] or with a clustering and tracking approach [29]. Nevertheless, these offline methods often rely on accessing all LiDAR observations in the sequence.

In this chapter, we focus on segmenting currently moving objects online and directly in 3D space from a limited time horizon of observations. Given a sequence of 3D LiDAR scans, we predict for each point if it belongs to a moving object, for example, bicyclists or driving cars, or a static, i.e., non-moving one, like parked cars, buildings, or trees. Our approach turns the sequence of observed LiDAR scans into a voxelized sparse 4D point cloud as explained in Sec. 2.4. We apply computationally efficient sparse 4D convolutions to jointly extract spatial and temporal features and predict moving object confidence scores for all points in the sequence. We develop a receding horizon strategy that allows us to predict moving objects online and refine predictions based on new observations. We use a static state binary Bayes filter to recursively integrate new scan predictions, resulting in a more robust estimation. We evaluate our approach on the SemanticKITTI MOS benchmark [15, 28] and show more accurate predictions than existing methods. Since our approach only operates on the geometric information of point clouds over time, it generalizes well to new environments, which we evaluate on a dataset not seen during training.

## 3.1 4DMOS – Moving Object Segmentation With 4D Convolutions

The main contribution of this chapter is 4DMOS, a novel approach that predicts moving objects online for a short sequence of LiDAR scans. We exploit sparse 4D convolutions to jointly extract spatio-temporal features from the input point cloud sequence. Such higher dimensional convolutions introduced in Sec. 2.4 allow us to efficiently process the data directly in the voxelized 4D space without the need of projecting to lower dimensional spaces like range images as done in previous works [29]. Such projections usually lead to information loss, and the back-projection and clustering to retrieve per-point predictions introduce artifacts like label bleeding [125]. The outputs of our network are moving object confidence scores for the points in each input scan.

Our method operates in a sliding window fashion and appends a new observed scan to the input sequence while discarding the oldest one. By doing so, our method can include new observations in the estimation as they arrive. We implemented a static state binary Bayes filter to fuse these predictions and, in this way, increase the robustness to false predictions. Since our method uses only the spatial point information over time, it is class agnostic and generalizes well on unseen data.

In sum, we make three key claims: Our approach (i) segments moving objects in LiDAR data more accurately compared to existing methods,

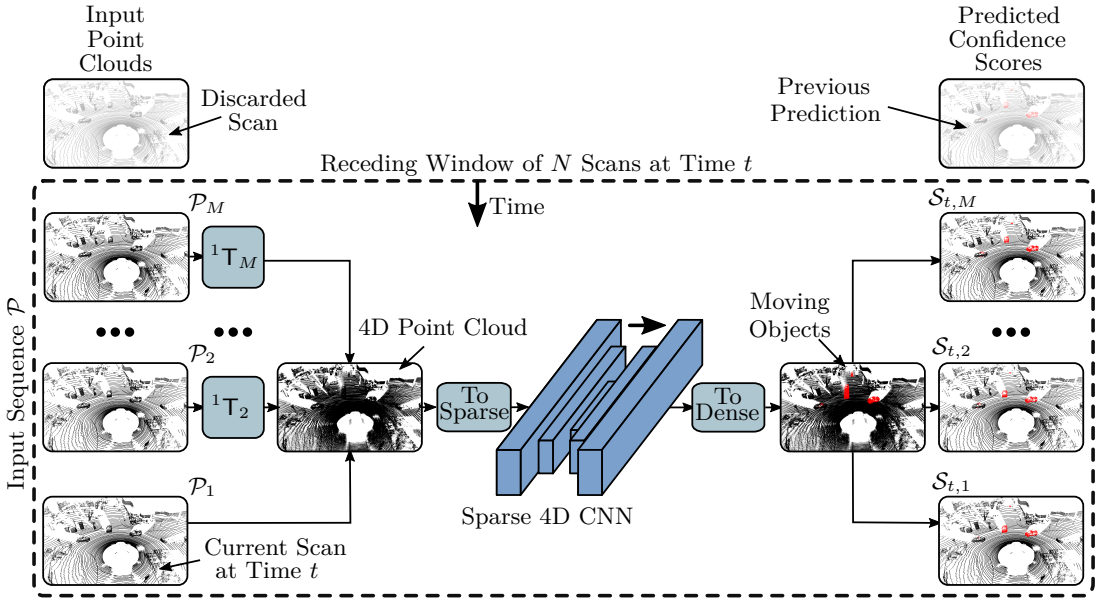



Figure 3.2: Overview of our approach operating with a receding horizon strategy. We transform all scans of the considered receding window to the current viewpoint. Next, we aggregate all points and create a sparse 4D point cloud. We apply sparse 4D convolutions to extract spatio-temporal features jointly. Our final layer predicts moving object confidence scores for all points in the input sequence.

(ii) generalizes well to unseen environments without additional transfer learning techniques, and (iii) improves the results by integrating new observations online. We explicitly back up these three claims by our experimental evaluation in Sec. 3.2. The code for this chapter and all pre-trained models are available at `https://github.com/PRBonn/4DMOS`.

### 3.1.1 Overview

Given a point cloud sequence $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_M\}$ of $M$ LiDAR scans $\mathcal{P}_t = \{\boldsymbol{p}_{t,1}, \boldsymbol{p}_{t,2}, \ldots, \boldsymbol{p}_{t,N_t}\}$ with $N_t$ points $\boldsymbol{p}_{t,i} \in \mathbb{R}^3$ taken at time $t$, we represent the points as homogeneous coordinates, i.e., $\mathbf{p}_i = [x_i, y_i, z_i, 1]^\top$. Our approach aims to predict which points are actually moving in the input sequence $\mathcal{P}$. As shown on the left in Fig. 3.2, we first express the points in a common reference frame to better distinguish between ego motion and the motion of objects. For simplicity, we transform the past point clouds $\{\mathcal{P}_2, \ldots, \mathcal{P}_M\}$ to the viewpoint of the current scan $\mathcal{P}_1$ and create a sparse 4D tensor, see Sec. 3.1.2.

We extract spatio-temporal features with a sparse convolutional architecture and predict confidence scores of being actually moving for each point in the sequence, see Sec. 3.1.3. As soon as we obtain a new LiDAR scan, we shift the prediction window as explained in Sec. 3.1.4. The receding horizon strategy allows

recursively updating the estimation by fusing later predictions for the same scan in the sequence in a static state binary Bayes filter, see Sec. 3.1.5.

### 3.1.2 Sequential Input Representation

The first step is to locally align all past point clouds $\{\mathcal{P}_2, \dots, \mathcal{P}_M\}$ in the sequence $\mathcal{P}$ to the viewpoint of the current LiDAR scan $\mathcal{P}_1$. In this chapter, we assume to have access to *estimated* relative pose transformations ${}^{j-1}\mathsf{T}_j$ from the frame of scan $\mathcal{P}_j$ to the frame of scan $\mathcal{P}_{j-1}$. Odometry estimation is a standard task for autonomous vehicles. We can efficiently solve it on-board with an online odometry system like KISS-ICP [188] or SLAM approaches like SuMa [17] or CT-ICP [41], further improved by integrating information from an inertial measurement unit [163] or by using wheel encoders. Our approach is agnostic to the source of odometry information, and a local consistency is sufficient. We provide an experiment on how the odometry affects the results in Sec. 3.2.7. In Chap. 4, we will show how to integrate the odometry estimation into the pipeline.

We represent the relative transformations between scans $\left\{{}^{1}\mathsf{T}_2, \dots, {}^{M-1}\mathsf{T}_M\right\}$ as homogeneous transformation matrices, i.e., ${}^{j-1}\mathsf{T}_j \in \mathbb{SE}(3)$. Further, we denote the $j^{\text{th}}$ scan transformed to the current viewpoint at index one by

$$\mathcal{P}^{j\to 1} = \left\{{}^{1}\mathsf{T}_j \mathbf{p}_i\right\}_{\mathbf{p}_i \in \mathcal{P}_j} \quad \text{with } {}^{1}\mathsf{T}_j = \prod_{k=0}^{j-2} {}^{j-k-1}\mathsf{T}_{j-k}. \tag{3.1}$$

The motivation behind locally aligning the scans in the sequence is to eliminate the ego motion of the vehicle, such that our CNN can focus on local point patterns that move in space over time relative to the ego motion. We also provide experiments on the effect of the pose alignment in Sec. 3.2.5 and Sec. 3.2.7. After applying the transformations, we aggregate the aligned scans into a 4D point cloud by converting from homogeneous coordinates to cartesian coordinates and by adding the time as an additional dimension, resulting in coordinates $[x_i, y_i, z_i, t_i]^\top$ for point $\boldsymbol{p}_i$.

Since outdoor point clouds obtained from a LiDAR sensor are naturally sparse, we quantize the 4D point cloud into a sparse voxel grid with a fixed resolution in time $\Delta t \in \mathbb{R}$ and the same $\Delta s \in \mathbb{R}$ for all three spatial axes. As explained in Sec. 2.4, we use a sparse tensor to represent the voxel grid and only store non-empty voxel indices and associated features. This representation efficiently encodes the sparse nature of the point cloud, comprising both the point coordinates $\mathsf{C}$ and the corresponding features $\mathsf{F}$. We formulate the sparse tensor in the following manner, where each row corresponds to a voxel:

$$\mathsf{C} = \begin{bmatrix} b_1 & x_1 & y_1 & z_1 & t_1 \\ & & \vdots & & \\ b_N & x_N & y_N & z_N & t_N \end{bmatrix}, \qquad \mathsf{F} = \begin{bmatrix} \boldsymbol{f}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{f}_N^\mathsf{T} \end{bmatrix}, \tag{3.2}$$

where $b_i$ is the batch index, $t_i$ is the time index of the 4D point and $\boldsymbol{f}_i$ is the feature vector associated to the $i$-th coordinate voxel. Sparse tensors are more memory efficient than dense voxel grids since they only store information about the voxels occupied by points instead of all voxels. The sparse representation allows us to efficiently use spatio-temporal CNNs since common dense 4D convolutions become intractable on large scenes.

### 3.1.3 Feature Extraction With Sparse 4D Convolutions

Several network architectures were proposed to work directly on point cloud data, such as PointNet [148], PointNet++ [149], KPConv [176], and PointNetLK [8]. However, most architectures are computationally expensive and cannot generalize well for high-dimensional spaces [32].

Using the sparse input representation discussed in Sec. 3.1.2, we can apply time- and memory-efficient sparse 4D convolutions to jointly extract spatio-temporal features from the sparse 4D occupancy grid and predict a moving object confidence score for each point. To this end, we use the MinkowskiEngine [32] for sparse convolutions, see Sec. 2.4.2. Sparse convolutions operate on the sparse tensor and define kernel maps that specify how the kernel weights connect the input and output coordinates. The main advantage of sparse convolutions is the computational speed-up compared to dense convolutions.

We use a sparse convolutional network developed for 4D semantic segmentation on RGB-D data and adapt it for MOS on LiDAR data. More specifically, we modify the MinkUNet32 [32] from Sec. 2.4.2 by reducing the number of layers. The network is a sparse equivalent of a residual bottleneck architecture with strided sparse convolutions for downsampling the feature maps and strided sparse transpose convolutions for upsampling. The skip connections in a UNet fashion [152] help to maintain details and fine-grained predictions. We reduce the number of feature channels in the network, resulting in a model with 1.8 M parameters, which is comparably low compared to the MOS baseline LMNet [28] using SalsaNext [35] with 6.7 M or RangeNet++ [125] with 50 M parameters backbones. We provide the details on how to optimize these weights in Sec. 3.2.1. The last layer of our network is a 4D sparse convolution with a softmax that predicts moving object confidence scores between 0 and 1 for each point. We can directly threshold these scores to obtain a MOS, or we can follow our receding horizon strategy and fuse them after aggregating more predictions as explained in the following sections, see Sec. 3.1.4 and Sec. 3.1.5.

In contrast to other semantic segmentation methods that use RGB values [32], point coordinates [28, 172], or intensity readings [28, 172] as input features on which the convolutional kernels operate, we initialize voxels occupied by at least one point with a constant feature. In this chapter, we set this constant to 0.5,

representing an unknown initial state as commonly used in occupancy grid mapping. Consequently, our input is a sparse 4D occupancy grid that only stores voxels occupied by at least one point. This design makes it easier to deploy the approach in new environments without estimating the distribution of coordinates or intensity values to standardize the input data as done for semantic segmentation [125]. We further investigate the generalization capability of our approach in Sec. 3.2.3.

### 3.1.4 Receding Horizon Strategy for Segmentation

The fully sparse convolutional architecture introduced in Sec. 3.1.3 jointly predicts moving object confidence scores for all points in the input sequence. At inference time, one option would be to divide the input data into fixed, non-overlapping intervals and predict each sub-sequence once.

Instead, we propose a receding horizon strategy for MOS. When the LiDAR sensor obtains the next point cloud, we add it to the input sequence and discard the oldest scan, resulting in a first in, first out queue, see Fig. 3.2. The main advantage is that we obtain an online estimation of the current scan's moving objects and re-estimate moving objects in past scans based on new observations, effectively increasing the time horizon used for prediction. It is a natural idea to use multiple observations to reduce the uncertainty of semantic estimations and has been well investigated in mapping algorithms like SuMa++ [30]. It is rarely used for online segmentation, and we propose a method to improve the online MOS.

### 3.1.5 Fusion With Static State Binary Bayes Filter

Since our proposed method simultaneously predicts moving objects in $M$ scans, the receding horizon strategy leads to re-estimating the previously predicted $M-1$ scans. These multiple predictions from different timestamps allow for refining moving object estimation based on new observations. Since the binary state of a point being moving or non-moving does not change over time, we propose to infer it by fusing multiple predictions recursively with a static state binary Bayes filter. The fusion effectively increases the time horizon used for the final segmentation. It can correct estimates for points that belong to slowly moving objects that only moved a small distance during the initial time horizon. The Bayesian fusion reduces the false positives and negatives arising from occlusions or noisy measurements.

Specifically, for a scan $\mathcal{P}_j$, we can estimate moving objects at time $t$ by fusing all predicted moving object confidence scores from previously observed point cloud sequences $\mathcal{Z}_{0:t} = \{\mathcal{Z}_0, \mathcal{Z}_1, \ldots, \mathcal{Z}_t\}$ that contain the scan $\mathcal{P}_j$. The term $\mathcal{Z}_t$

denotes the observed input point cloud sequence $\{\mathcal{P}_1, \ldots, \mathcal{P}_M\}$ with $\mathcal{P}_1$ recorded at time $t$. We want to estimate the joint probability distribution of the moving states $\mathcal{M}_j = \{m_{j,i}\}$ of all points $i$ in point cloud $j$ up to time $t$ denoted by

$$p\left(\mathcal{M}_j \mid \mathcal{Z}_{0:t}\right) = \prod_i p\left(m_{j,i} \mid \mathcal{Z}_{0:t}\right), \tag{3.3}$$

where $m_{j,i} \in \{0, 1\}$ is the state of point $\boldsymbol{p}_i \in \mathcal{P}_j$ being moving. Note that this implies that the moving state of two points is independent, which is true for points that are far away, but it does not hold if the points belong to the same object. However, since we do not have instance information, it is a valid assumption that works well in practice. For better readability, we now consider a single point $\boldsymbol{p}_i$ in point cloud $\mathcal{P}_j$ and omit the subscript $j$ without loss of generality.

We apply Bayes' rule to the per-point probability distribution $p\left(m_i \mid \mathcal{Z}_{0:t}\right)$ in Eq. (3.3) and follow the standard derivation of the recursive static state binary Bayes filter [178]. Using the log-odds notation $l(x) = \log \frac{p(x)}{1-p(x)}$ commonly used in occupancy grid mapping, we finally end up with

$$l\left(m_i \mid \mathcal{Z}_{0:t}\right) = \begin{cases} l\left(m_i \mid \mathcal{Z}_{0:t-1}\right) + l\left(m_i \mid \mathcal{Z}_t\right) - l(m_i), & \text{if } t \in \mathcal{T} \\ l\left(m_i \mid \mathcal{Z}_{0:t-1}\right), & \text{otherwise,} \end{cases} \tag{3.4}$$

with $\mathcal{T}$ being the set of timestamps in which we observe point $\boldsymbol{p}_i$ in the input sequence $\mathcal{Z}_t$. Whereas $l\left(m_i \mid \mathcal{Z}_{0:t-1}\right)$ is a recursive term including all predictions for the point $i$ up to time $t-1$, the term $l\left(m_i \mid \mathcal{Z}_t\right)$ denotes the logits of the probability to be moving at time $t$. Note that if we do not observe the point $\boldsymbol{p}_i$ at time $t$, there is no prediction, and we do not update the recursive term $l\left(m_i \mid \mathcal{Z}_{0:t-1}\right)$. The prior probability $0 \leq p_0 \leq 1$ in the last part $l(m_i) = \log \frac{p_0}{1-p_0}$ provides a measure of the innovation introduced by a new prediction. For MOS, the prior determines how much a predicted moving point in a single scan influences the final prediction. We will investigate different priors in Sec. 3.2.5.

At time $t$, our network outputs scores $\mathcal{S}_{t,j} \in \{s_{t,i}\}_{i=1}^{N_j}$ with $s_{t,i} \in \mathbb{R}$ that represent moving object confidence scores for each point cloud $\mathcal{P}_j$ with $N_j$ points given the current input sequence $\mathcal{Z}_t$. We can interpret them as the logits $l\left(m_i \mid \mathcal{Z}_t\right)$ of the posterior probability of point $\boldsymbol{p}_i$ being moving given the observed sequence $\mathcal{Z}_t$.

Fig. 3.3 illustrates the non-overlapping strategy in the upper part and our proposed receding horizon strategy with a static state binary Bayes filter to fuse multiple predictions in the lower part. We obtain the final prediction by converting the recursively estimated per-point logits $l\left(m_i \mid \mathcal{Z}_{0:t}\right)$ to a confidence using

$$p\left(m_i \mid \mathcal{Z}_{0:t}\right) = \frac{\exp\left(l\left(m_i \mid \mathcal{Z}_{0:t}\right)\right)}{1 + \exp\left(l\left(m_i \mid \mathcal{Z}_{0:t}\right)\right)}, \tag{3.5}$$

and consider a point being moving if the confidence is larger than 0.5 and non-moving otherwise.

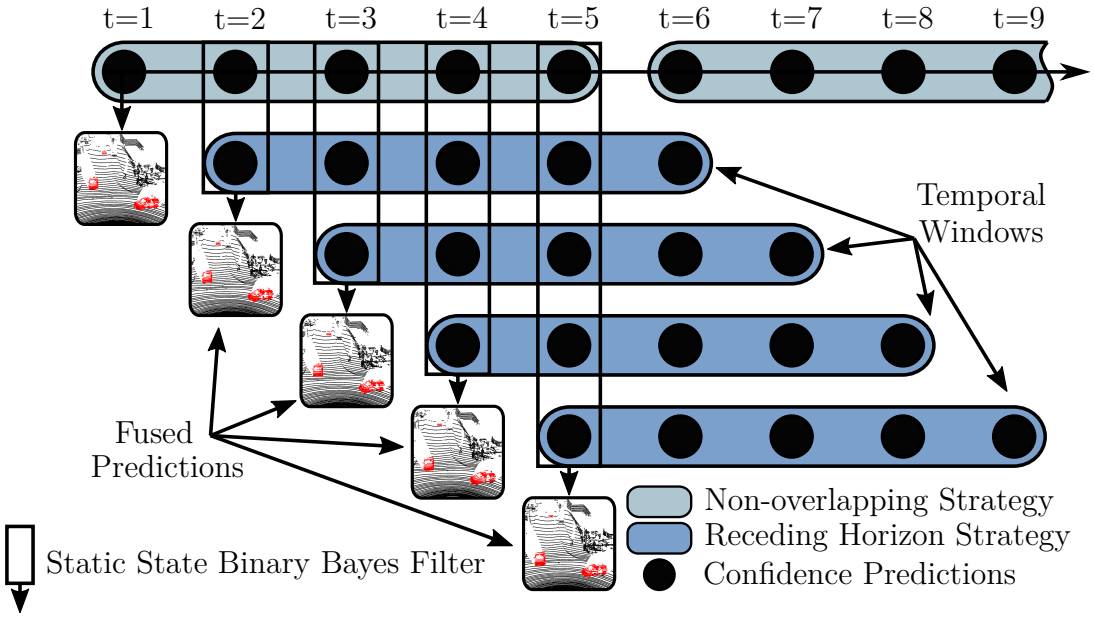



Figure 3.3: Overview of our proposed static state binary Bayes filter. At $t = 5$, the non-overlapping strategy uses the five per-scan confidence predictions, whereas our receding horizon strategy integrates the next observation at $t = 6$ by shifting the temporal window. Our static state binary Bayes filter then fuses multiple moving object confidence scores to improve the prediction.

### 3.1.6 Implementation Details

During training, we optimize the model with a binary cross-entropy loss for all points in the input sequence, a learning rate of 0.0001 and a weight decay of 0.0001 with the Adam optimizer [92]. If not stated differently in the experiments, our input point clouds sequences contain $M = 10$ input scans with a temporal resolution of $\Delta t = 0.1\,\text{s}$ between scans. The spatial voxel size for quantization is $\Delta s = 0.1\,\text{m}$.

## 3.2 Experimental Evaluation

The main focus of this chapter is a method to segment currently moving objects in 3D LiDAR data by exploiting consecutive scans online. Additionally, we carry out the prediction using a receding horizon strategy and integrate new predictions recursively in a static state binary Bayes filter.

We present our experiments to show the capabilities of our method and to support our three key claims: Our approach (i) segments moving objects in LiDAR data more accurately compared to existing methods, (ii) generalizes well to unseen environments without additional transfer learning techniques, and (iii) improves the results by integrating new observations online.

### 3.2.1 Experimental Setup

For our experimental evaluation, we train all models on the SemanticKITTI [16] dataset. We follow the data split from the SemanticKITTI MOS benchmark [28] and use sequences 00-07 and 09-10 for training, 08 for validation, and 11-21 for testing. To increase the diversity of the training data and to avoid overfitting, we follow the data augmentation strategy of Nunes et al. [132] and apply random rotations, shifting, flipping, jittering, and scaling to all points in the same 4D point cloud. We train all networks for less than 60 epochs and keep the model with the best performance on the validation set. We follow the receding horizon strategy presented in Sec. 3.1.4 and combine predictions with the static state binary Bayes filter proposed in Sec. 3.1.5 using a prior of $p_0 = 0.25$.

For the quantitative evaluation, we report the standard intersection over union (IoU) metric [49] for the moving class given by

$$\mathrm{IoU}_{\mathrm{MOS}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \tag{3.6}$$

with true positive TP, false positive FP, and false negative FN classifications of moving points.

To evaluate the generalization capability of our approach across environments, we additionally test our method on another dataset without using transfer learning techniques. We follow the setup of Chen et al. [29] and use the Apollo-ColumbiaParkMapData [113] dataset sequence 2 (frames 22300-24300) and sequence 3 (frames 3100-3600) annotated the same way as SemanticKITTI. SemanticKITTI and Apollo use Velodyne HDL-64E LiDAR scanners, but they are mounted on different cars at different heights and record data in various environments. This leads to a different distribution of point coordinates, such as the relative distance from the sensor to the ground plane.

### 3.2.2 Moving Object Segmentation Performance

Our first experiment evaluates the performance of our model on the SemanticKITTI [16] MOS benchmark [28]. The results support the first claim about segmenting moving objects more accurately than existing published and open-source methods. For a fair comparison, we follow the setup from LMNet [28] and use the provided SemanticKITTI poses estimated with an online SLAM system [17]. We report the result on the hidden test set in Tab. 3.1 and compare it to additional baselines provided by Chen et al. [28].

One can see that a segmentation with SalsaNext [35] based on a single-scan and predicting all movable classes as "moving" leads to low performance of 4.4% $\mathrm{IoU}_{\mathrm{MOS}}$. The same applies to estimating scene flow and thresholding the flow

| Approach | $\text{IoU}_{\text{MOS}}$ [%] |
|---|---|
| SalsaNext [35] (movable classes) | 4.4 |
| SceneFlow [109] | 4.8 |
| SpSequenceNet [164] | 43.2 |
| LMNet [28] | 58.3 |
| KPConv [176] | 60.9 |
| Ours, $M = 10$ Scans, $\Delta t = 0.1\,\text{s}$, $p_0 = 0.25$ | **65.2** |
| LMNet+AutoMOS+Extra [29] | 62.3 |

Table 3.1: Performance on SemanticKITTI [16] MOS benchmark [28]. Baseline results taken from [29]. The best results are in bold.

vectors to determine if an object moves. The online multi-scan semantic segmentation methods SpSequenceNet [164] and KPConv [176] as well as the projection-based MOS approach LMNet [28] show improved results up to 60.9% $\text{IoU}_{\text{MOS}}$, see Tab. 3.1. Our method can outperform all baselines with an $\text{IoU}_{\text{MOS}}$ of 65.2%, demonstrating our approach's effectiveness. Our performance is also better than LMNet+AutoMOS+Extra [29], which uses automatically generated moving object labels for training, emphasizing the strength of our result.

### 3.2.3 Generalization Capabilities

The following experiment evaluates our method's generalization abilities. It supports our second claim that the approach generalizes well on unseen data. We test our model on the Apollo dataset without using transfer learning techniques or re-training and compare it to baselines using different transfer learning types. LMNet [28] uses the same SemanticKITTI [16] sequences for training, whereas LMNet+AutoMOS [29] is LMNet trained on an automatically labeled training set of Apollo. LMNet+AutoMOS+Fine-Tuned [29] is a model pre-trained on SemanticKITTI and fine-tuned on Apollo, see [29] for details. The results in Tab. 3.2 suggest that transfer learning like re-training or fine-tuning improves the results with a maximum $\text{IoU}_{\text{MOS}}$ of 65.9% for the baselines. Our method yields the highest $\text{IoU}_{\text{MOS}}$ of 73.1% without any additional steps, which shows that the approach can predict moving objects in an unknown environment.

We hypothesize that extracting moving object features in a sparse 4D occupancy grid is advantageous since we do not use sensor-specific information like intensity or RGB values. Operating in 4D space also avoids overfitting to a specific sensor location, as in range images, where moving objects are usually found in certain areas of the image. We also do not use information about semantic classes whose distribution can differ between environments.

| Approach | $\text{IoU}_{\text{MOS}}$ [%] |
|---|---|
| LMNet [28] | 16.9 |
| LMNet+AutoMOS [29] | 45.7 |
| LMNet+AutoMOS+Fine-Tuned [29] | 65.9 |
| Ours, $M = 10$ Scans, $\Delta t = 0.1\,\text{s}$, $p_0 = 0.25$ | **73.1** |

Table 3.2: Performance on Apollo [113] dataset. The best results are in bold.

### 3.2.4 Prior for Static State Binary Bayes Filter

This section backs up our third claim that the proposed receding horizon strategy combined with a static state binary Bayes filter improves the MOS results by integrating online new observations. We investigate the effect of using different numbers of input scans $M$ and temporal resolutions $\Delta t$ for prediction as well as fusing with different prior probabilities $p_0$ in the Bayesian fusion presented in Sec. 3.1.5.

We compare models trained on $M = 2$, $M = 5$, and $M = 10$ input and output scans and a model that predicts a single output scan. Since the combination of a receding horizon strategy and the Bayesian fusion of multiple beliefs allows us to use information from a larger time horizon, we additionally compare to two variant setups using $M = 5$ input and output frames but with a different temporal resolution. One uses a resolution of $\Delta t = 0.2\,\text{s}$ between scans, resulting in a total time horizon of 0.8 s, the other one processes 1.2 s of scans that are $\Delta t = 0.3\,\text{s}$ apart. For comparison, the method using $M = 2$ scans with a resolution of $\Delta t = 0.1\,\text{s}$ has a time horizon of 0.2 s, the one with $M = 5$ scans a horizon of 0.4 s and the model using $M = 10$ scans looks at 0.9 s of data. We visualize each variant's time horizons and temporal resolutions in Fig. 3.4 as colored dots on a timeline sampled at 10 Hz.

The Bayesian prior $p_0$ in Eq. (3.4) serves to compute the difference between the new predicted logits and the initially expected logits. Therefore, modifying the prior influences the contribution of newly observed moving objects to the updated prediction. Fig. 3.4 shows the $\text{IoU}_{\text{MOS}}$ on the SemanticKITTI validation set for different priors. With a small prior like $p_0 = 0.01$, we fuse predicted moving objects more aggressively, leading to more true positives but also an increased number of false positives since inconsistent predictions are not filtered out. A large prior like $p_0 = 0.99$ results in a conservative fusion where we only predict objects to move if all predictions agree. We found that a moving object prior between 0.1 and 0.3 works better for the SemanticKITTI validation sequence. We experienced that for many slowly moving objects in the scene, setting a lower prior helps keep them in the final prediction even if we did not detect them as

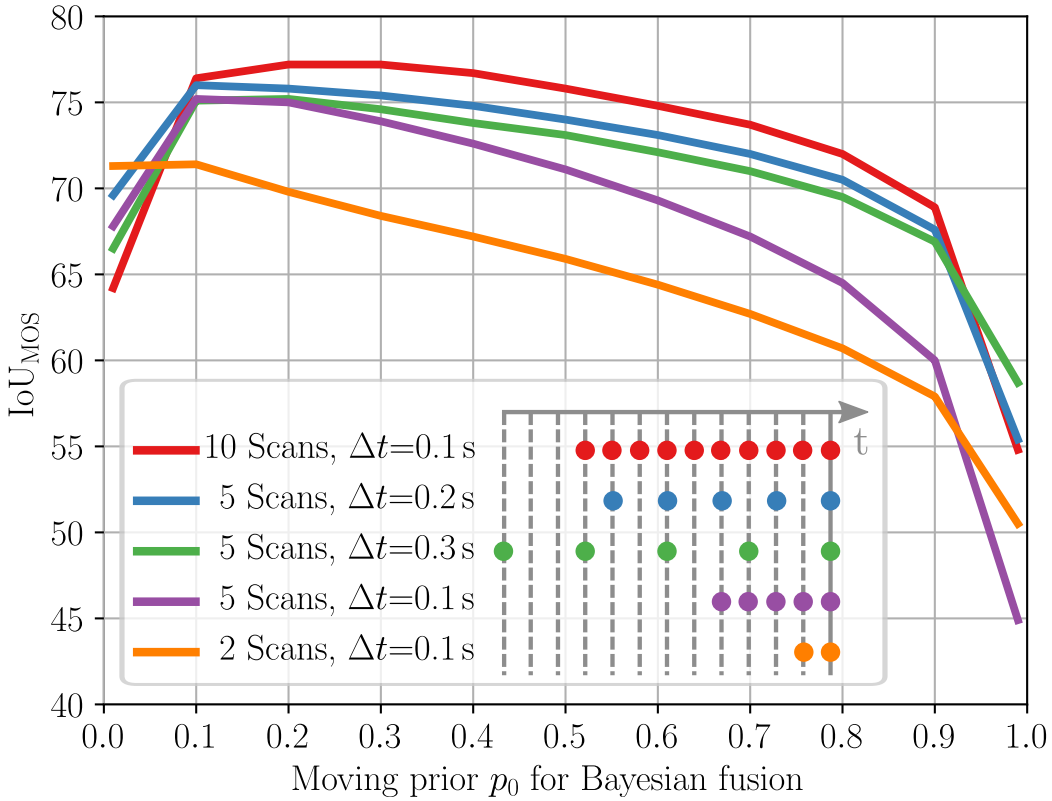



Figure 3.4: Comparison of $\text{IoU}_{\text{MOS}}$ on the SemanticKITTI [16] validation set using different moving object priors for the static state binary Bayes filter. The colors indicate variants of our approach using different time horizons and resolutions. The colored dots on the timeline visualize which past scans we input to the model for prediction at time $t$.

"moving" from all available instances. We achieve the best result with a model using $M = 10$ input and output frames and a Bayesian prior of $p_0 = 0.25$, which is the setup for the experiments presented in Sec. 3.2.2 and Sec. 3.2.3.

In general, processing more scans and fusing multiple predictions with the receding horizon strategy to achieve a larger time horizon results in better MOS performance. Our approach also works with fewer scans but with a larger temporal resolution, reducing computational effort. The models using $M = 5$ input scans with a larger temporal resolution of $\Delta t = 0.2\,\text{s}$ and $\Delta t = 0.3\,\text{s}$ between scans outperform the model with the same number of processed scans but a smaller resolution of $\Delta t = 0.1\,\text{s}$, which is the sensor frame rate. This experiment shows that a larger temporal resolution leads to better segmented slowly moving objects since their motion is more visible in the sequence.

### 3.2.5 Parameter Study

To further support our third claim and show the effectiveness of individual proposed components of our approach, we train different variants of our network and evaluate their performance on the validation set. We train all models for up

| Variant | # Inputs | # Outputs | Poses | $\Delta t$ | $\text{IoU}_{\text{MOS}}$ [%] | |
|---|---|---|---|---|---|---|
| | | | | | w/o BF | w/ BF |
| [A] Five Scans | 5 | 5 | ✓ | 0.1 s | 69.1 | 74.5 |
| [B] $\Delta t = 0.2$ s | 5 | 5 | ✓ | 0.2 s | 71.8 | 75.6 |
| [C] $\Delta t = 0.3$ s | 5 | 5 | ✓ | 0.3 s | 71.6 | 74.9 |
| [D] No Poses | 5 | 5 | ✗ | 0.1 s | 35.6 | 39.9 |
| [E] Single Output | 5 | 1 | ✓ | 0.1 s | 66.5 | - |
| [F] Two Scans | 2 | 2 | ✓ | 0.1 s | 64.9 | 69.0 |
| [G] Ten Scans | 10 | 10 | ✓ | 0.1 s | 74.3 | **77.2** |

Table 3.3: Parameter study on different variants of our approach with and without the proposed receding horizon strategy and Bayesian fusion (BF) using a prior $p_0 = 0.25$. We denote the temporal resolution between scans by $\Delta t$. We underline the best result for each row, and the best result among all is bold.

to 60 epochs and report the best $\text{IoU}_{\text{MOS}}$ on the validation set during training, see Tab. 3.3. For all methods, we compare two prediction strategies: first, a non-overlapping strategy that divides the input sequence into sub-sequences and predicts each sub-sequence independently, see the upper part in Fig. 3.3. Second, our receding horizon strategy proposed in Sec. 3.1.4, which generates multiple predictions for the same scan and fuses them in a static state binary Bayes filter (again using a prior of $p_0 = 0.25$). We visualize this combination in the lower part of Fig. 3.3.

We generally see an improvement of up to 5.4 percentage points of $\text{IoU}_{\text{MOS}}$ for all models using the proposed receding horizon strategy. More precisely, using the static state binary Bayes filter with model [A] reduces the number of false negatives by 8.2% and the number of false positives by 18.9%. This result indicates that the proposed approach successfully integrates more observations into the estimation and is more robust to false predictions due to occlusions or noisy measurements. If we compare the performance of model [A] using $M = 5$ scans which are $\Delta t = 0.1$ s apart to the networks trained with larger temporal resolutions of $\Delta t = 0.2$ s [B] and $\Delta t = 0.3$ s [C], we again see that we can further improve the results by considering a larger time horizon, see also Sec. 3.2.4. If we do not transform the point clouds into a common viewpoint, the method [D] can still infer moving objects but with a reduced performance of $\text{IoU}_{\text{MOS}} = 39.9\%$ with Bayesian fusion. The network must infer both the sensor's ego motion and the objects' relative motion. When only training to predict a single output scan [E], the result is worse, and fusing more predictions is not possible since no additional predictions are available. Next, our method can also achieve MOS only with two scans ([F]) but with the worst performance. The best-performing model [G] takes $M = 10$ input scans with a temporal resolution of $\Delta t = 0.1$ s and fuses the

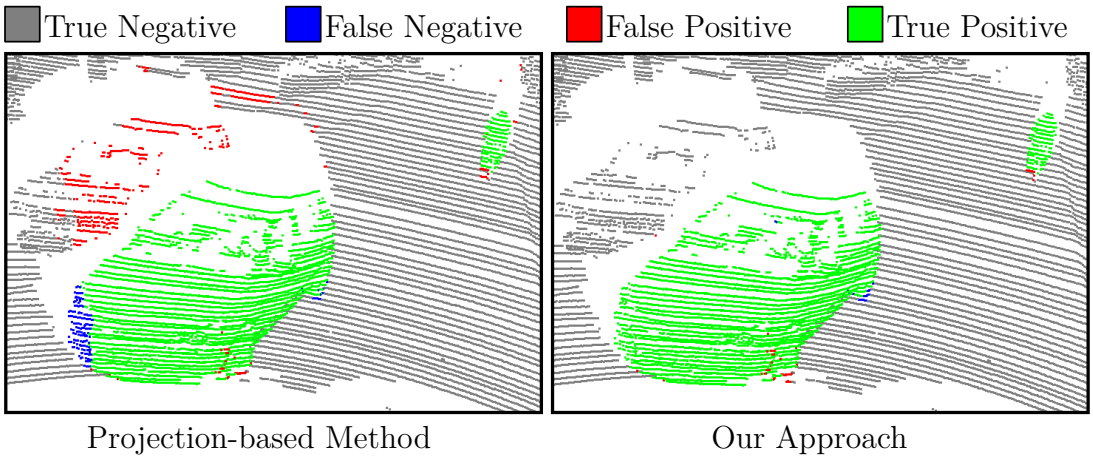



Figure 3.5: Qualitative comparison of segmentation accuracy. *Left*: Prediction by range image-based LMNet [28] after k-nearest neighbor post-processing. *Right*: Our sparse voxel-based approach without further post-processing. The figure is best viewed in color.

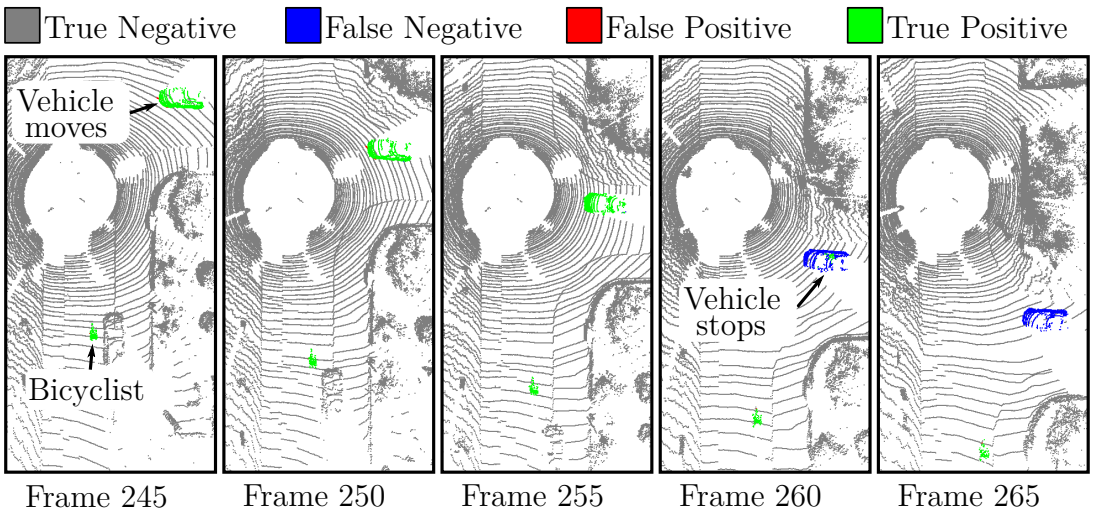



Figure 3.6: Change in MOS if an object stops moving. The figure is best viewed in color.

predictions resulting in an $\text{IoU}_{\text{MOS}}$ of 77.2%. The results show that we achieve a better MOS after increasing the time horizon by processing more scans, increasing the temporal resolution, and using the proposed receding horizon strategy with a static state binary Bayes filter.

### 3.2.6 Qualitative Results

This section illustrates that our method predicts currently moving objects in 3D space without requiring geometric post-processing like clustering. We use the model from Sec. 3.2.2 trained on $M = 10$ scans with a temporal resolution of $\Delta t = 0.1\,\text{s}$. In Fig. 3.5, we show the segmentation of scan 1638 from the SemanticKITTI [16] validation sequence 08. We compare the range image-based method LMNet [28] to our sparse voxel-based approach. One can see that de-

| Odometry | SuMa [17] Frame To Model | SuMa [17] Frame To Frame | CT-ICP [41] | G-ICP [162] Frame To Frame |
|---|---|---|---|---|
| $\text{IoU}_{\text{MOS}}$ [%] | 77.2 | 76.9 | 76.5 | 76.3 |

Table 3.4: Influence of different odometry information sources on the $\text{IoU}_{\text{MOS}}$ performance of our best method on the SemanticKITTI validation sequence 08.

spite the geometric-based k-nearest neighbor post-processing, the baseline still shows artifacts and bleeding labels behind the moving car illustrated as red-colored false positives. In contrast, our method directly predicts in the space without boundary effects.

Next, Fig. 3.6 shows how the prediction changes for a scene in the validation set where a vehicle stops moving. Since our method does not use semantic understanding of objects, it only reasons how the points move in space for the given time horizon. Since the vehicle stops moving to yield at the intersection, our method's prediction changes from moving to static. We successfully classify the bicyclist in the back as "moving". Note that this results in a false negative indicated in blue since the ground truth SemanticKITTI labels consider if an object has moved throughout the data collection and not based on recent movement.

### 3.2.7 Odometry Sources

As shown in Sec. 3.2.5, our approach works best when expressing the points of the LiDAR sequence in a common reference frame using estimated poses. In this section, we investigate the effect of different odometry estimation approaches on the performance on the SemanticKITTI validation sequence 08 and report the results in Tab. 3.4. We compare with variants of SuMa [17] called "Frame To Model" and "Frame To Frame", CT-ICP [41] and G-ICP [162] with a "Frame To Frame" error function. Note that for the SuMa "Frame To Model", we estimate the poses without loop closing since our method does not need global accuracy. We take the odometry results for G-ICP from Vizzo et al. [185]. As we can see, our approach also works well if provided with odometry estimated using different methods.

### 3.2.8 Runtime

With our Pytorch implementation, the network requires on average 0.078 s for predicting moving objects in 10 scans and 0.047 s for 5 scans both using an NVIDIA RTX A5000 GPU. Note that we can further reduce the runtime by, for example, clipping the points at a specific range because closer objects are

usually of higher interest than far-away points. Our static state binary Bayes filter only adds a small overhead of 0.008 s on average for fusing 10 predictions and 0.004 s for fusing 5 predictions.

## 3.3 Related Work

The identification of dynamic objects in LiDAR data gained increasing interest in research to obtain a static environment representation for various downstream tasks. This section reviews related works from closely connected research fields to give context for our proposed sequence-based MOS approach. In general, there is no standard definition of moving objects, and it depends on the application which entities we consider moving or dynamic to process them further.

**Movable Object Segmentation.** Segmentation approaches identify movable objects based on their semantic appearance in a single LiDAR scan and reject these for data association in SLAM [30, 142, 191]. For example, Wang et al. [191] apply a graph-based bottom-up clustering approach to segment foreground objects that could move in 3D LiDAR data, like cars, pedestrians, and bicyclists. More specifically, they compute point normals based on a principal component analysis of neighboring point distances and represent the points as an undirected graph. Next, they cluster the points into patches based on the similarity of the normals. Lastly, the authors extract additional handcrafted features and classify the patches using a support vector machine with non-linear radial basis function kernels. They train multiple one-versus-all binary classifiers based on labeled training data. The resulting segmentation only considers movable classes and does not reason about the actual movement of the objects. Besides that, it is restricted to the pre-defined classes on which the classifiers are trained.

Such a semantic segmentation of movable objects can help avoid wrong data associations in SLAM, as shown by Chen et al. [30], who exploit the range image-based semantic segmentation approach RangeNet++ [125] to get point-wise semantic labels including movable objects. The authors enforce a semantic consistency between new observations and the map and remove parts containing inconsistent semantic labels. This allows the removal of moving objects and the preservation of objects like parked cars, which are movable but remain static. In contrast, our approach directly infers moving objects and does not require additional consistency checks with a map to identify them.

Pfreundschuh et al. [142] further demonstrate the effectiveness of MOS in LiDAR data for data associations in dynamic object aware LiDAR SLAM. They build upon LOAM [214], a LiDAR odometry and mapping approach that estimates the robot's pose by aligning planar and edge features to a sparse feature map. To avoid associations with dynamic objects, Pfreundschuh et al. [142] de-

ploy a 3D-MiniNet [6] to predict the semantic labels of points within a single scan. The authors additionally propose a labeling pipeline based on occupancy grid mapping to generate moving labels for training the network. The data used in the experiments only contains pedestrians, so it is unclear how the approach performs with additional classes like cars or bicyclists. Besides that, the segmentation module is again only capable of identifying movable objects because it operates on a single scan.

The semantic class of the points is only sometimes of interest. For example, Ruchti and Burgard [153] use a deep neural network to predict dynamic probabilities for each point in a range image before fusing them with a map. The authors infer the ground truth dynamic probability by projecting the laser scans into the camera frames and transferring the semantic classes from labeled bounding boxes, which are available for movable objects. We follow this idea not to classify the individual semantics but to infer a dynamic state of the points – in our case, if they currently move instead of just being movable.

So far, we mainly considered approaches inferring movable objects like cars, pedestrians, or bicyclists. However, long-term changes like moved furniture or opened and closed doors can influence the robot's localization Thomas et al. [177] propose a self-supervised method for classifying indoor LiDAR points into dynamic labels. The authors explicitly distinguish between short-term and long-term movable objects to treat them differently in localization and planning. Other researchers encode non-static objects like open and closed doors into the map by maintaining clusters of possible world configurations and estimating multi-modal states [171].

This thesis focuses on currently moving objects instead of being movable. Knowing which objects are actively changing in the environment is critical for online path planning, collision avoidance, or behavior prediction. We do not further distinguish the objects' semantic class, making it easier to obtain labeled data, train the approach, and deploy it to new, unseen environments.

**Projection-based Moving Object Segmentation.** Online LiDAR MOS aims at segmenting LiDAR scans into moving and non-moving points. In contrast to the methods mentioned above, we are interested in objects that currently move, independent of their semantic class. To do this, we can no longer rely on a single scan but must consider multiple LiDAR scans. For example, Yoon et al. [207] identify moving objects based on the residual between two scans, free space filtering, and region growing post-processing. One drawback of such an approach is that objects can be temporarily occluded, making it hard to identify motion from two scans. Subsequent works extend the temporal horizon of past information used for prediction. However, this comes at a higher computational cost due to the large amount of data that has to be processed.

Projection-based methods reduce the size of the 3D data stream by projecting it to lower-dimensional representations like range images [28, 91, 172]. Chen et al. [28] develop LMNet based on the previously mentioned RangeNet++ [125], which performs semantic segmentation of a single range image. They extend the network by adding additional residual range images as inputs to provide the temporal information. The authors compute the residual images between the current and previously received scans and transform them into the current frame of the sensor. A high value in the residual image indicates a discrepancy between both measured points and serves as a cue for the network to segment moving objects in the image.

Recently, Mohapatra et al. [126] introduce a method using two successive bird's eye view images for MOS and achieve faster runtime but inferior performance compared to LMNet. MotionBEV [216] further improves the performance of bird's eye view-based LiDAR MOS by leveraging the height differences of two polar bird's eye view grids to obtain the temporal information. Additionally, they extract the appearance information of a query bird's eye view image and combine temporal and appearance features to get a final semantic segmentation. The approach shows a promising performance on SemanticKITTI but requires semantic labels for training. Their experiments demonstrate that re-training is necessary to outperform 4DMOS on the SipailouCampus dataset [216]. Our approach does not require semantic labels or has to be re-trained when inferring moving objects using a different sensor modality or environment.

In general, projection-based methods often suffer from information loss or back-projection artifacts, called "label bleeding" for range image-based segmentation [125]. Such methods usually require additional steps like k-nearest neighbor clustering [28, 35, 50, 125] or additional point refinement modules [172]. In contrast, we predict moving objects in the voxelized 4D space without prior projection to address the problem of label bleeding. We assume that the motion of an object is visible within a limited time horizon of consecutive past scans, which we aggregate to a sparse 4D point cloud. By shifting this temporal window, we can refine the prediction of previous scans by fusing them in a point-wise static state binary Bayes filter, effectively increasing the temporal information used for prediction.

**Spatio-Temporal Data Processing.** Extracting temporal information from sequential point cloud data is gaining more attention in research since it increases temporal consistency for classification tasks or predicting future states of the environment. To fuse independent semantic single-scan predictions, Dewan and Burgard [45] use a static state binary Bayes filter by propagating previous predictions to the following scan using scene flow. In contrast, Duerr et al. [50] optimize a recurrent neural network to temporally align range image

features from a single-scan semantic segmentation network. Some works project the spatial information into 2D representations like range images [28, 50, 91, 98, 120] or bird's eye view images [114, 126, 198, 216] and then apply 2D or 3D convolutions to reduce the computational burden of jointly processing 4D data.

Point-based methods directly operate on the input point cloud without the need for projection or voxelization [52, 53, 110]. Fan et al. [53] propose PointRNN, which extends the RNN framework to operate on point clouds. The authors provide point-based versions of a gated recurrent unit and an LSTM. The main drawback of such recurrent point-based methods is that they are not straightforward to train and are often only applied on smaller point clouds instead of large, outdoor LiDAR scans like ours.

To alleviate the computational complexity of processing large, full-scale point cloud sequences, researchers investigated the use of higher-dimensional convolutions on sparse voxel grids. Representing point clouds as sparse tensors can also circumvent the back-projection issue and makes it possible to apply sparse convolutions efficiently. For example, Shi et al. [164] develop SpSequenceNet for 4D semantic segmentation, which processes two LiDAR frames with sparse 3D convolutions and combines their temporal information with a cross-frame global attention module. To apply convolutions across time, Choy et al. [32] propose Minkowski networks for semantic segmentation using sparse 4D convolutions on temporal RGB-D data.

Our architecture to segment moving objects in a sequence of LiDAR scans is based on the MinkowskiEngine, enabling us to extract spatio-temporal information using sparse convolutions efficiently.

**Scene Flow Estimation.** A related research field is scene flow estimation, which aims to model how point clouds change over time. Previously, scene-flow methods first classify moving points and then estimate separate flows for static and moving objects between two point clouds [14, 60, 181]. In more detail, Baur et al. [14] estimate the 3D scene flow between two point clouds composed of a rigid body motion for static and a per-point flow for moving objects. Based on the discrepancy between per-point flow and rigid body motion, they use a self-supervised motion segmentation signal to train their network.

Even though MOS can be a by-product of scene flow estimation, most methods only consider two consecutive frames, which could be a too short time horizon for classifying slowly moving objects. Since the resulting point clouds from LiDAR scanners are irregularly sampled from the environment, there is no clear correspondence between points of consecutive scans. We, therefore, focus on only segmenting the scans instead of modeling the actual motion.

## 3.4 Conclusion

This chapter presented a novel approach to segmenting moving objects from a sequence of 3D LiDAR scans and directly addressed our first research question of "What is moving?". We built upon sparse spatio-temporal convolutions on a sparse voxel grid to directly infer moving objects in the 3D space over time. In contrast to previous methods, our approach does not require a data projection to lower dimensional spaces, and we can apply it to large, outdoor LiDAR point clouds. Our method jointly predicts moving objects for all scans in the input sequence and operates using a receding horizon strategy.

We report improved performance on the SemanticKITTI MOS benchmark and show that the approach generalizes well on unseen data. The experiments show that fusing multiple predictions over a longer time horizon with our proposed receding horizon strategy in combination with a static state binary Bayes filter increases the robustness to false positive and false negative predictions. Since our approach provides point-wise MOS predictions, we can build a static map by only integrating non-moving points. However, once we integrate a point into the map, we cannot recover it if it turns out to be a moving point. Besides that, we currently estimate moving points based on a limited number of past frames. In some scenarios, when objects are, for example, temporarily occluded, it is more beneficial to consider all available past information.

In the next chapter, we aim to estimate moving objects based on a local map that considers all past measurements. Additionally, we propose maintaining a spatial belief model that allows us to integrate predictions to update our internal representation of the dynamic environment.

# Chapter 4

# Map-Based Moving Object Segmentation

Mobile robots that navigate in the real world often rely on a static representation of their environment in the first place. Additionally, they need to be constantly aware of the dynamic objects in their surroundings as this is relevant information for mapping [153, 187], localization [71, 142], planning [97], or occupancy prediction [73]. Thus, it is crucial to reason about moving objects in the current observation while maintaining an internal representation of the static world.

If the robot requires such a static map at runtime, one way to build such a model is to segment each incoming scan into moving and non-moving points and then integrate only the static parts into the map [28]. In this setup, the robot would perform each segmentation independently of previous predictions and add the non-moving points to the static map. The downside of such an approach is that moving objects falsely classified as non-moving will remain in the static map, and it is not straightforward to remove them. These so-called "ghost artifacts" will negatively impact path planning based on such a model. Consequently, mobile robots need to be able to re-estimate previously missing moving objects and to correct their internal belief about the dynamic environment to effectively answer the question of "What is moving?" introduced earlier.

In the previous Chap. 3, we presented 4DMOS to segment moving objects in a sequence of LiDAR scans. Using our proposed receding horizon strategy, we re-estimated moving objects in a scan after receiving more observations and fused them in a static state binary Bayes filter. The experiments in Sec. 3.2.5 demonstrate two main findings: First, the length of the time horizon considered for prediction is critical to segment moving objects successfully. Second, fusing multiple predictions for the same points obtained at different timestamps leads to a more robust estimation.

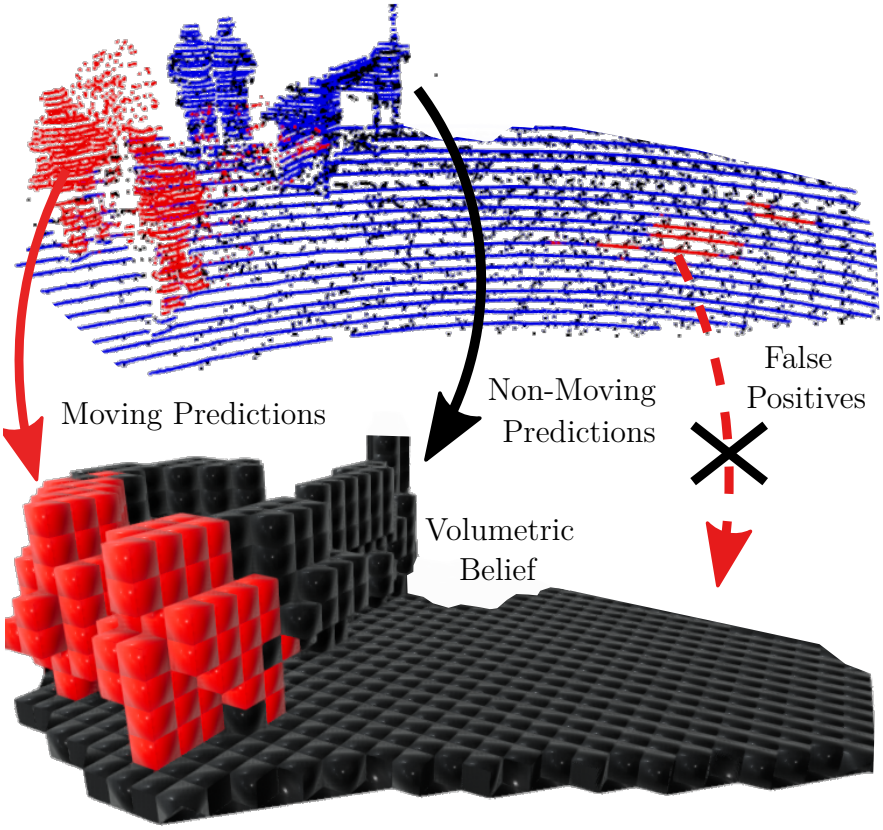



Figure 4.1: Our approach identifies moving objects *(red)* in the current scan *(blue)* and the local map *(black)* of the environment. We maintain a volumetric belief map representing the dynamic environment and fuse new predictions probabilistically. This fusion allows us to reject false positive predictions that contradict our volumetric belief.

The main limitation of 4DMOS is that we can only improve the MOS if the corresponding scan is within the limited buffer of past scans that 4DMOS and related approaches consider for prediction. In addition, we only fuse the predictions on the point level, such that we treat points close to each other independently – an assumption that works well for MOS, but we will revisit it in this chapter for online static mapping. Lastly, using a limited window for MOS assumes that we can identify the movement of an object from consecutive measurements. This assumption usually holds for most rotating LiDAR scanners that scan the surroundings with a regular scanning pattern at high frequency, but not for scanners with a limited field of view or irregular sampling patterns [106]. Therefore, we propose collecting more environmental measurements to segment moving objects.

In this chapter, we take inspiration from 4DMOS, but use all past information available and do not restrict the estimation to a limited sequence of observations as done in Chap. 3. To do this, we maintain a local map that contains information from all point clouds we measured in our local neighborhood. We jointly estimate moving objects in the current 3D LiDAR scan and this local map of the environ-

ment. We use sparse 4D convolutions to extract spatio-temporal features from the scan and local map and segment all 3D points into moving and non-moving.

To model the dynamic environment, we maintain a 3D volumetric belief about which part of the space can contain moving objects as depicted in Fig. 4.1. We update the belief online by fusing our predictions probabilistically to increase precision and recall of MOS similar to the static state binary Bayes filter in Sec. 3.1.5. Our experiments show that our approach outperforms existing moving object segmentation baselines and even generalizes to different types of LiDAR sensors. We demonstrate that our volumetric belief fusion increases the precision and recall of moving object segmentation and even corrects the estimation of moving objects falsely classified as non-moving in an online mapping scenario.

## 4.1 MapMOS – Building Volumetric Beliefs For Dynamic Environments

The main contributions of this chapter are two-fold. First, we propose an approach to predict moving objects in a local map constructed using all past LiDAR measurements recorded in this area without limiting the time horizon. Second, we build and maintain a volumetric belief map and fuse new predictions in a voxel-wise static state binary Bayes filter to previous estimates online, which increases robustness and corrects previously wrong predictions. In sum, we make four key claims: Our approach, MapMOS, can (i) accurately segment an incoming LiDAR scan into moving and non-moving objects based on a local map of past observations, (ii) generalize well to new environments and sensor setups while achieving state-of-the-art performance, (iii) increase the precision and recall of MOS by fusing multiple predictions into a volumetric belief, (iv) recover from wrong predictions for online mapping through a volumetric belief. The content of this chapter and our experimental evaluation back up these claims. Our code, pre-trained models, and labels for evaluation are available at `https://github.com/PRBonn/MapMOS`.

### 4.1.1 Overview

We propose segmenting moving objects based on the discrepancy between the current LiDAR frame and a local map of the previously measured scans in that area. Given the current LiDAR frame at time $t$, we first register it to our current local map as explained in Sec. 4.1.2. Next, we jointly predict moving objects in the aligned scan and the local map, see Sec. 4.1.3. After that, we fuse these predictions into a probabilistic volumetric belief to maintain a representation of the dynamic environment, see Sec. 4.1.4. We can query the volumetric belief for

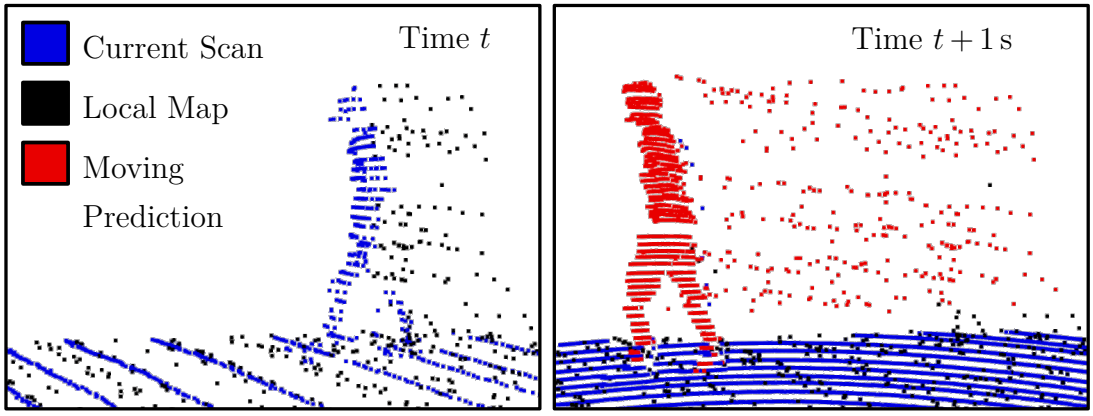



Figure 4.2: Current Scan and local map for two different times with our moving predictions in red. Although our method initially failed to fully identify the moving pedestrian in the beginning *(left)*, we successfully predict it at a later point in time, and our method backtraces the corresponding points in the local map *(right)*.

a set of points as explained in Sec. 4.1.5 to obtain the current belief that these points belong to moving objects.

### 4.1.2 Scan Registration with KISS-ICP

Our approach does not require given poses like 4DMOS presented in Chap. 3 or even ground truth poses but only relies on sequential 3D LiDAR data. In contrast, we compute the pose incrementally on the fly via scan matching, and our MOS approach can directly operate based on this data. When a new measurement is available, we register the scan to the previous ones stored in a map using KISS-ICP [188]. This robust odometry pipeline generalizes well to varying motion profiles and sensor platforms without changing parameters.

We use the map to store past measurements and represent it by a sparse voxel grid as done in KISS-ICP. We remove points further away to reduce the memory overhead and refer to it as local map. We maintain the original coordinates of the points in the voxels to avoid discretization errors. In contrast to the original KISS-ICP implementation, we additionally store the timestamp of the scan it stems from for each point to maintain temporal information in the local map. Our method uses this temporal information to predict moving objects for the registered scan and the local map.

### 4.1.3 Map-Based Moving Object Segmentation

This section explains how to jointly predict moving objects after registering a new LiDAR frame. We exploit two different mechanisms to segment moving objects. First, we consider the spatial discrepancy between the current scan and the local map. This information indicates if an object may have moved with respect to all

previous measurements in that area. Second, we identify the motion of objects based on the evolution of the timestamps given by the feature attached to each point. This information allows us to segment both the current scan and the local map into moving and non-moving parts.

In contrast to 4DMOS or other previous works [28, 91, 172], we do not restrict our method to a fixed set of past scans. This is advantageous in cases where a moving object is not fully visible within a short time horizon due to occlusion, limited field of view, or an irregular shooting pattern of the LiDAR. In practice, this makes a substantial difference.

Additionally, instead of predicting moving objects in the current scan or a limited buffer of scans, we segment both the current scan and the local map. Segmenting the local map enables us to identify traces of moving objects that we did not segment in previous scan predictions. This backtracing of dynamic objects allows us to correct initial false negative predictions as shown in Fig. 4.2.

Our local map is the voxel grid structure of KISS-ICP, but we store for every point its 4D coordinate (position plus time). To maintain the ordering of scan and local map during the convolutions, we organize them in a 4D tensor. We use the timestamps as features for the points and normalize them based on the minimum and maximum values since we are only interested in their relative difference. This normalization avoids the model overfitting to the sequence lengths and, therefore, the maximum timestamps it has seen during training.

At time $t$, we voxelize the 4D point cloud $\mathcal{P}_t$ of scan and local map and represent it as a sparse 4D tensor using the MinkowskiEngine [32]. Sparse tensors are a more memory-efficient representation for 4D tensor data and allow direct application of sparse convolutions. We jointly extract spatial and temporal features with sparse 4D convolutions. Our network architecture is a 4D MinkUNet [32] with 1.8 M parameters, see Sec. 2.4.2. This network first downsamples the points and features in an encoder to extract high-level information and then upsamples both to the original resolution in a decoder. Residual blocks and skip connections help to maintain detailed information about the points and their corresponding features. The last layer predicts the logits $\mathcal{S}_t = \{s_{t,1}, s_{t,2}, \ldots, s_{t,N}\}$ with $s_{t,j} \in \mathbb{R}$ of $N$ points from both the current scan and local map points being moving. Fig. 4.3 depicts an overview of our approach.

### 4.1.4 Volumetric Belief Update

This section presents our approach to fusing per-point MOS predictions into a probabilistic volumetric belief. We demonstrated in Chap. 3 that fusing multiple independent per-point predictions over time can filter out prediction errors from the neural network. Instead of fusing per-point predictions in this chapter, we aim to model which parts of the environment have a higher probability of containing

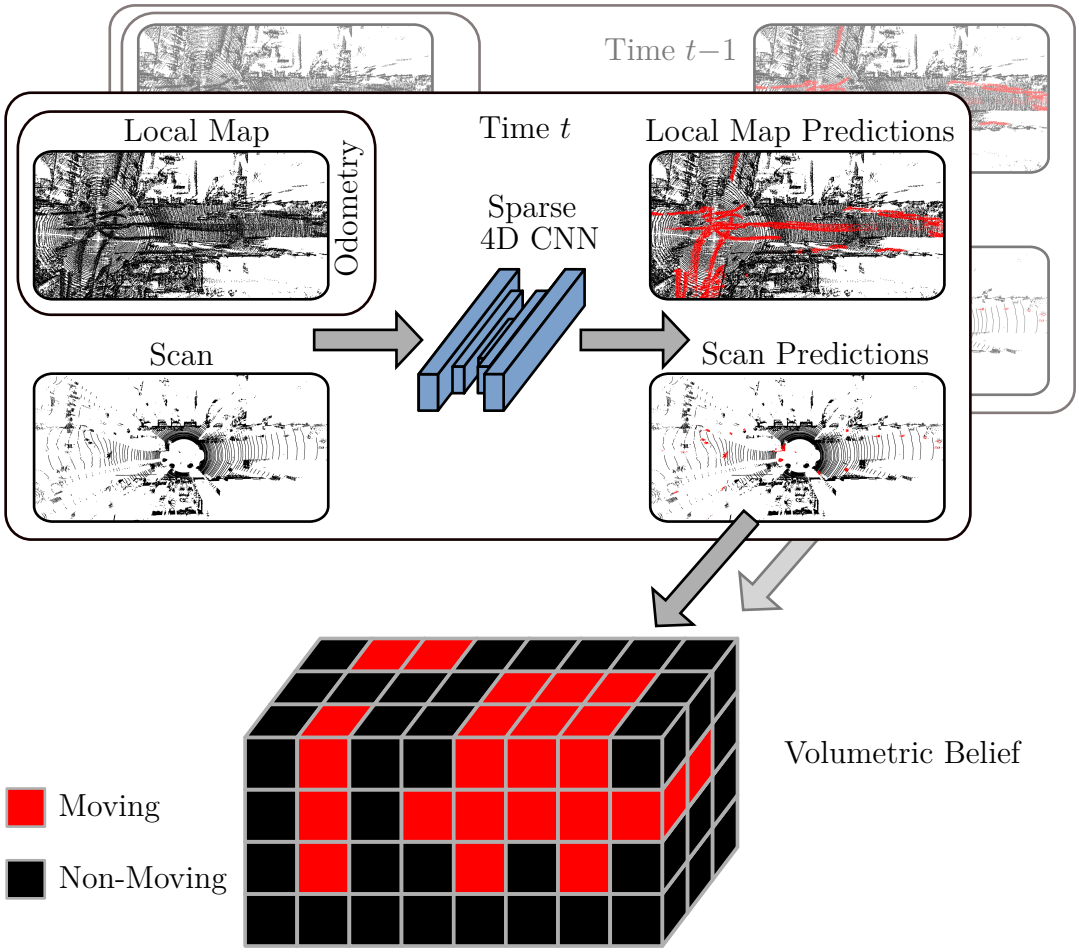



Figure 4.3: Overview our proposed MOS approach and volumetric belief fusion. At time $t$, we predict moving objects in the current scan and local map using sparse 4D convolutions. Next, we update our volumetric belief about which parts of the environment can contain moving objects based on the previous volumetric belief at time $t-1$ and our new predictions.

a dynamic object. In this case, we do not want to identify current dynamics but rather determine which map portion is traversed by moving objects. We define this property as *dynamic occupancy*.

We assume the binary state $m_i \in \{0, 1\}$ of dynamic occupancy for a voxel $\boldsymbol{v}_i$ does not change over time. Intuitively, this means that if a point falls into a voxel previously occupied by dynamics, we assume that this point also belongs to a moving object. On the other hand, if static points occupied a voxel, we do not expect to observe a moving object in this volume. Note that this state definition differs from occupancy grid mapping, where the world is assumed to be static, and a fixed occupancy probability of a cell is estimated.

At time $t$, we predict $N$ logits $\mathcal{S}_t = \{s_{t,1}, s_{t,2}, \ldots, s_{t,N}\}$ with $s_{t,j} \in \mathbb{R}$ for $N$ points $\mathcal{P}_t = \{\boldsymbol{p}_{t,1}, \boldsymbol{p}_{t,2}, \ldots, \boldsymbol{p}_{t,N}\}$ with $\boldsymbol{p}_{t,j} \in \mathbb{R}^4$ as described in Sec. 4.1.3. It is possible to fuse the logits for the current scan but also for the local map points. We provide an experiment in Sec. 4.2.4 to showcase the results for different fusion

strategies. Note that our volumetric belief is not restricted to our logits, but one could integrate predictions from different sources. Our goal is to estimate the joint probability distribution of the volumetric belief map state for all voxels $\mathcal{M} = \{m_i\}$ reading

$$p\left(\mathcal{M} \mid \mathcal{P}_{1:t}\right) = \prod_i p\left(m_i \mid \mathcal{P}_{1:t}\right), \tag{4.1}$$

with $\mathcal{P}_{1:t} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_t\}$ being the previously measured points up to time $t$.

After applying Bayes' rule to the right-hand side probability distributions of the voxels $i$, we can derive the recursive static state binary Bayes filter equations according to Thrun et al. [178]. We use the log-odds notation $l(x) = \log \frac{p(x)}{1-p(x)}$ resulting in

$$l\left(m_i \mid \mathcal{P}_{1:t}\right) = l\left(m_i \mid \mathcal{P}_{1:t-1}\right) + l\left(m_i \mid \mathcal{P}_t\right) - l(m_i), \tag{4.2}$$

for updating a single voxel cell belief $l\left(m_i \mid \mathcal{P}_{1:t}\right)$. Here, $l\left(m_i \mid \mathcal{P}_{1:t-1}\right)$ is the recursive term currently stored in the voxel, which aggregates the previous predictions, $l\left(m_i \mid \mathcal{P}_t\right)$ is the update term for the voxel which integrates the predictions at the current time $t$, and $l(m_i)$ is the logits of the prior probability $p_0$. We do not assume prior knowledge about the dynamic occupancy of a voxel $\boldsymbol{v}_i$ and set it to $p_0 = 0.5$.

The remaining step is to get a per-voxel update $l\left(m_i \mid \mathcal{P}_t\right)$ from the points $\mathcal{P}_t$ and logits $\mathcal{S}_t$. The prediction $s_{t,j} \in \mathcal{S}_t$ at time $t$ for a single point $\boldsymbol{p}_{t,j}$ with index $j$ indicates if it belongs to a moving object or not. Since multiple points with different logits can end up in the same voxel, we need to aggregate their information and take the arithmetic mean of logits inside a voxel $\boldsymbol{v}_i$, resulting in

$$l\left(m_i \mid \mathcal{P}_t\right) = \frac{\sum_{j \in \mathcal{V}_{t,i}} s_{t,j}}{|\mathcal{V}_{t,i}|}, \tag{4.3}$$

where $\mathcal{V}_{t,i} = \{j \mid \boldsymbol{p}_{t,j} \in \boldsymbol{v}_i\}$ is the set of points falling into the voxel $\boldsymbol{v}_i$ at time $t$ and $|\mathcal{V}_{t,i}|$ is the cardinality of the set. Taking the arithmetic mean of per-point logits corresponds to the geometric mean of the individual likelihoods of a point being moving. Likelihood aggregation using the geometric mean has been previously used in Monte-Carlo localization sensor model designs [218].

We implement our volumetric belief as a hash table, a more memory-efficient representation than dense 3D arrays [130, 188]. Each 3D voxel $\boldsymbol{v}_i$ stores the logits belief $l\left(m_i \mid \mathcal{P}_{1:t}\right)$ about its dynamic occupancy state $m_i \in \{0, 1\}$ after integrating predictions up to time $t$.

### 4.1.5 Volumetric Belief Query

For a given set of points, we can query our volumetric belief by indexing the corresponding voxels $\boldsymbol{v}_i$ and converting the logits beliefs $l\left(m_i \mid \mathcal{P}_{1:t}\right)$ to a posterior probability $p$ using $p(x) = \frac{\exp(l(x))}{1+\exp(l(x))}$. We assume a point is moving if the

probability is larger than 0.5. Note that the voxel size of our volumetric belief needs to be appropriate since the underlying assumption is that all points inside a voxel share the same dynamic occupancy state. This assumption is violated if the voxel size is too large. Therefore, the voxel size indicates how close points can be to share the same dynamic occupancy. However, the downside of a small voxel size is the increased runtime and memory overhead.

### 4.1.6 Online Mapping

For online mapping, we are interested in accurately removing moving points. We experienced discretization effects at the boundaries of moving objects, for example, false negative predictions on the wheels of vehicles close to the ground. To achieve sub-voxel accuracy and a high recall for identifying moving objects, we combine the filtered voxel-wise volumetric belief with the point-wise scan prediction for online mapping. We demonstrate this in an experiment in Sec. 4.2.5.

### 4.1.7 Implementation Details

We set the voxel size for downsampling and the local map in our odometry system to 0.5 m, which is half the default value of KISS-ICP. This gives us a denser local map for MOS and still runs reasonably fast. For the volumetric belief map, we must trade-off between computational efficiency and accuracy due to the discretization error by assuming that each voxel has a single dynamic state. For our experiments, we set the voxel size to 0.25 m and clip the volumetric belief map at 150 m to limit the memory footprint. We chose 0.25 m because we expect the distance between moving and non-moving objects, for example, a pedestrian and a wall or a vehicle and the ground, to be around this threshold.

## 4.2 Experimental Evaluation

The main focus of this work is an approach to identify moving objects in the current LiDAR frame and a local map of aggregated past scans and to fuse these predictions into a probabilistic volumetric belief map. We present our experiments to show the capabilities of our method. The results of our experiments also support our key claims, which are: Our approach (i) accurately segments an incoming LiDAR scan into moving and non-moving objects based on a local map of past observations, (ii) generalizes well to new environments and sensor setups while achieving state-of-the-art performance, (iii) increases the precision and recall of MOS by fusing multiple predictions into a volumetric belief, (iv) recovers from wrong predictions for online mapping through a volumetric belief.

### 4.2.1 Experimental Setup

For the following experiments, we follow the official SemanticKITTI MOS benchmark setup for a fair comparison. Therefore, we train all models on the moving labels of the SemanticKITTI [15, 16] training sequences 00-07 and 09-10 and use sequence 08 for validation. We do not use the pose information provided by the KITTI dataset since we register the scans using KISS-ICP [188] as described in Sec. 4.1.2. We train our 4D CNN by supervising the prediction for scan and local map points using the cross-entropy loss for 100 epochs and save the model performing best on the validation set. Since some training set sequences do not contain many moving objects, we skip a batch if the ratio between moving and static points is less than 0.1 %. Next, we crop a rectangular patch of the scenes and augment the batch by rotating, flipping, and scaling. Lastly, we randomly drop points with a dropout rate sampled from the interval $[0, 0.5]$ to vary the density of the point clouds. One epoch takes less than 25 min on an NVIDIA RTX A5000 GPU.

Besides the commonly used SemanticKITTI MOS benchmark [28] based on the SemanticKITTI labels, we also evaluate and compare our approach on a labeled sequence from the KITTI Tracking [58] dataset recorded with the same sensor setup in a street with a lot of moving pedestrians. We additionally report results on a subset of the Apollo Columbia Park MapData [113] with labels provided by Chen et al. [29]. This data is recorded with the same sensor but in a different city environment. To push the generalization capabilities of MOS approaches, we test the models trained on SemanticKITTI with 64 vertical beams at 10 Hz frequency on the nuScenes [23] dataset, which has 32 vertical beams at 20 Hz. We evaluate the MOS for nuScenes based on the moving labels from the annotated keyframes of the 150 validation sequences.

We compare our method to our previous work 4DMOS presented in Chap. 3, which also applies sparse 4D convolutions but on a limited buffer of aggregated registered past scans. Additionally, we report results from the projection-based baselines LMNet [28], MotionSeg3D [172], and RVMOS [91]. For MotionSeg3D, we show the results without (v1) and with the proposed point refinement (v2), see [172] for details. In all experiments, we assess the performance using the commonly known IoU [49] of the moving points and additionally precision and recall in Sec. 4.2.4.

### 4.2.2 Moving Object Segmentation Performance

In the first experiment, we evaluate how well our approach segments a scan into moving and non-moving points using a local map of past observations. We show the originally reported baseline results on the SemanticKITTI validation set and

| Method | Test 11-21 | Validation 08 |
|---|---|---|
| LMNet [28] | 58.3 | 66.4 |
| MotionSeg3D, v1 [172] | 62.5 | 68.1 |
| MotionSeg3D, v2 [172] | 64.9 | 71.4 |
| 4DMOS, delayed [121] | 65.2 | 77.2 |
| Ours, Scan | 65.9 | 83.8 |
| Ours, Volumetric Belief | **66.0** | **86.1** |
| RVMOS [91]* | **73.3*** | 71.2* |

Table 4.1: Comparison of average moving IoU on the SemanticKITTI validation sequence 08 and the SemanticKITTI MOS benchmark [28]. The best results are in bold. The * indicates that the approach additionally *exploits semantic labels* and thus needs representative training data from the domain.

the SemanticKITTI MOS benchmark. We only consider approaches trained and validated on the original SemanticKITTI split to provide fair comparisons and eliminate the positive bias of additional training data [29, 172].

We evaluate the predictions of the current scan (referred to as "Scan") and the volumetric belief with a delay of 10 scans (referred to as "Volumetric Belief"). The choice of 10 scans is an initial estimate that trades off the ability to correct previous wrong estimates and the required waiting time. Besides fusing all scan predictions, we integrate only the local map points we predict to be moving. This has two reasons: First, we are mainly interested in the moving objects in the local map that we have falsely classified as non-moving in previous scan predictions. Second, integrating all local map points reduces the runtime of the system. The delay of 10 scans helps to get a more informed belief about the voxels with additional local map predictions before querying their state.

One can see in Tab. 4.1 that our volumetric belief helps to improve the results on the validation sequence, whereas the effect is more negligible on the test set. We further investigate the impact of the volumetric belief in Sec. 4.2.4. Our approach outperforms 4DMOS, showing that not limiting past information is beneficial for MOS. In general, we rank second best on the hidden test set. We are only outperformed by RVMOS, which requires additional semantic labels for training, while all other approaches use the moving object labels. Our approach using the volumetric belief achieves the highest result on the validation set with 86.1 % IoU for the moving points.

Our MOS model runs at 12 Hz on the full SemanticKITTI MOS benchmark test scans using an NVIDIA RTX A5000 GPU. We implemented the volumetric belief update and querying in C++, and it runs at 44 Hz on an Intel Xeon W-1290P CPU with multi-threading. Like 4DMOS in Sec. 3.2.8, we can fur-

| Method | KITTI [58] Tracking 19 | Apollo [113] | nuScenes [23] Validation |
|---|---|---|---|
| LMNet [28] | 45.3 | 13.7 | n/a |
| MotionSeg3D, v1 [172] | 54,6 | 6.5 | n/a |
| MotionSeg3D, v2 [172] | 54,8 | 8.8 | n/a |
| 4DMOS, delayed [121] | 75.5 | 70.9 | **44.8** |
| 4DMOS, online | 71.1 | 68.7 | 34.6 |
| Ours, Scan | 77.0 | 79.2 | 36.8 |
| Ours, Volumetric Belief | **78.4** | **81.7** | 40.3 |

Table 4.2: Generalization capabilities of different methods on datasets outside the training distribution. We report the average moving IoU. The best results are in bold.

ther reduce the runtime by reducing the voxelization resolution or limiting the maximum range of the scans used for MOS.

### 4.2.3 Generalization Capabilities

The following experiment analyzes how well our approach generalizes to new environments and sensor setups. Since MOS is often a supervised task and labeling is expensive, generalization is an important property. We provide an experiment in Tab. 4.2 that realizes different domain shift levels and compares how well the approaches generalize.

All baselines require external pose information. For a fair comparison, we provide the poses computed with KISS-ICP. In the case of 4DMOS, we also report the result of segmenting the most recent scan to compare the online performance before refining with the originally proposed receding horizon strategy and static state binary Bayes filter. Unfortunately, the code for RVMOS is not publicly available, so we cannot evaluate its performance on additional datasets.

One can see that the projection-based approaches LMNet and MotionSeg3D perform worse on the highly crowded KITTI Tracking sequence 19. Their performance drops even further on the Apollo dataset. We believe this is because they implicitly overfit to the calibration of the LiDAR sensor, such as mounting location and intensity measurements.

In contrast, 4DMOS and our approach only use the temporal information of the scans and therefore generalize well to a new sensor calibration. We again obtain the best result using our volumetric belief with a delay of 10 scans (referred to as "Volumetric Belief") and outperform 4DMOS. The performance of our approach with and without the volumetric belief in terms of $\text{IoU}_{\text{MOS}}$ is close, but we will investigate their differences with respect to the precision and recall more closely in Sec. 8.2.5.

For the nuScenes dataset, we cannot evaluate the pre-trained models for the projection-based approaches in a fair comparison because the range image dimensions change due to the different vertical resolutions of the sensors. Both 4DMOS and our approach can still segment moving objects, but the average moving IoU is lower. Here, the strategy of 4DMOS shows the best results. When comparing the current scan predictions only, we achieve a better result in moving IoU.

### 4.2.4 Volumetric Belief

Next, we conduct experiments showing how our proposed volumetric belief can improve moving IoU, recall, and precision. We compare the prediction of our model for the current scan (referred to as "Scan") to our volumetric belief after fusing only the scan prediction (referred to as "Volumetric Belief, Scan Only").

One can see from Tab. 4.3 that the probabilistic fusion using a static state binary Bayes filter consistently increases the precision of our scan prediction by rejecting false positives in previously predicted regions. At the same time, the recall drops due to the discretization error between ground points and the boundary of moving objects. Next, we additionally fuse the local map points that we predict to be moving (referred to as "Volumetric Belief, No Delay"). The results indicate that additionally fusing the local map predictions increases the recall compared to the volumetric belief that only integrates scan predictions.

Our last setup (referred to as "Volumetric Belief") first integrates 10 scan and moving local map predictions into our volumetric belief before querying it for evaluation as explained in Sec. 4.2.2. Again, this setup achieves the best result in most sequences in terms of IoU since we can now use the local map predictions to identify traces of moving objects and update the volumetric belief accordingly, even if the previous scan-based prediction was static. In the case of Apollo, the setup using the volumetric belief only fusing scan predictions is slightly better in moving IoU. Since the recall of moving objects in the scan predictions is already high for Apollo, we believe that the negative impact of discretization errors from additionally fusing moving local map points is more dominant in the final IoU than the improvement from correcting false negatives.

### 4.2.5 Online Mapping

Finally, we analyze how we can use our approach and the corresponding volumetric belief for online mapping. We use the VDBFusion [187] library that provides a reconstruction pipeline based on truncated signed distance functions using the VDB data structure to build a final 3D model [128]. We show the results in Fig. 4.4 for the CYT_02 sequence from the Loam_livox dataset [106] (top row) and for the KITTI Tracking sequence 19 (bottom row). The CYT_02 data

| Method | SemanticKITTI [16, 28] Validation 08 | | | KITTI [58] Tracking 19 | | | Apollo [113] | | | nuScenes [23] Validation | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IoU | R | P | IoU | R | P | IoU | R | P | IoU | R | P |
| Scan | 83.8 | 87.5 | 95.3 | 77.0 | **84.6** | 89.6 | 79.2 | **93.0** | 84.5 | 36.8 | 43.4 | 70.0 |
| Volumetric Belief, Scan Only | 84.0 | 86.4 | **96.8** | 76.7 | 80.3 | **94.4** | **82.1** | 92.3 | **88.6** | 36.6 | 40.8 | **81.1** |
| Volumetric Belief, No Delay | 83.9 | 86.7 | 96.3 | 76.9 | 81.8 | 92.8 | 81.3 | 92.4 | 87.7 | 36.9 | 41.7 | 79.4 |
| Volumetric Belief | **86.1** | **88.7** | **96.8** | **78.4** | 83.4 | 92.9 | 81.7 | 92.9 | 87.7 | **40.3** | **45.7** | 77.9 |

Table 4.3: Ablation study on average moving IoU, recall (R), and precision (P) in % for our scan-based prediction and different volumetric belief fusion strategies.

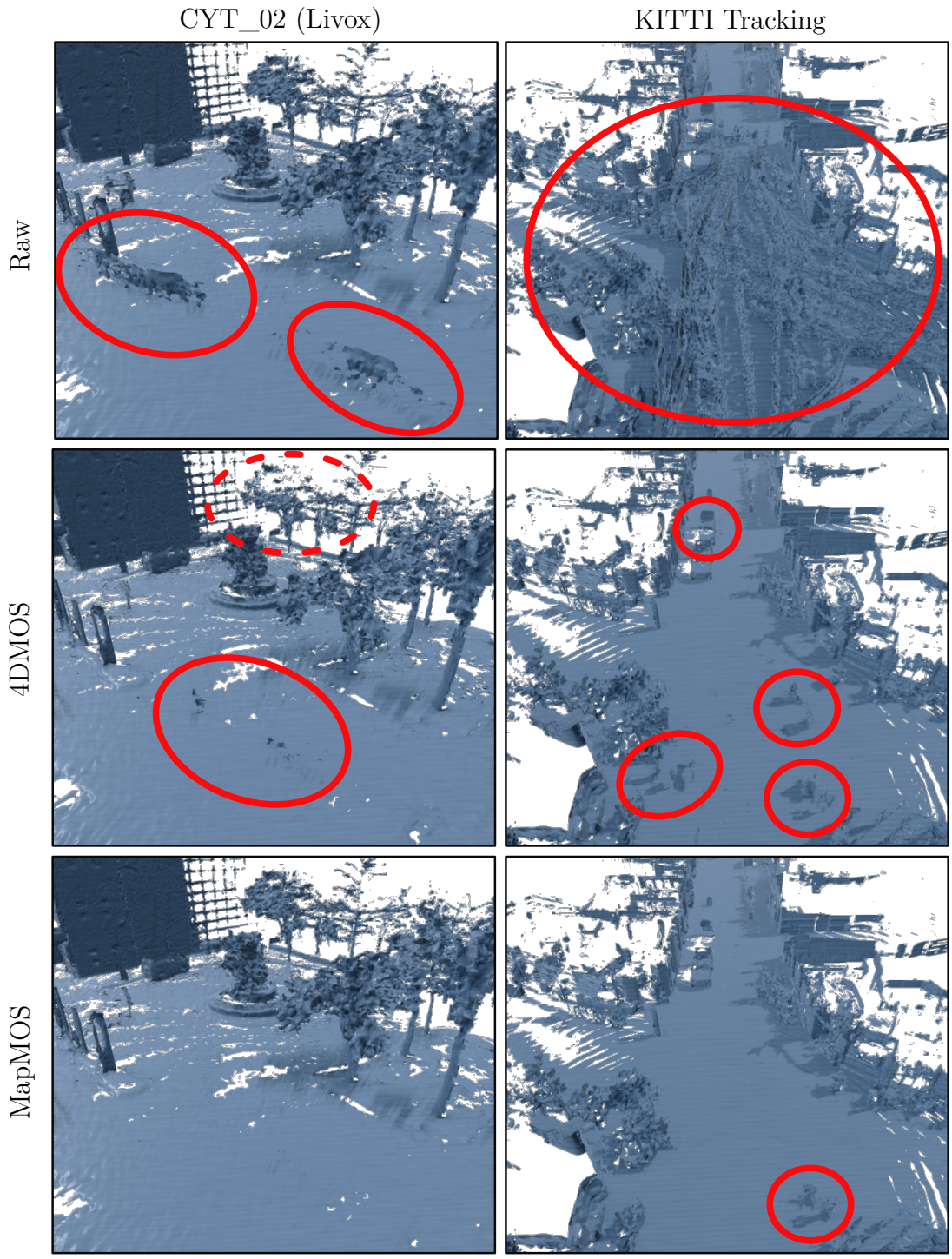



Figure 4.4: Reconstructed surfaces obtained with VDBFusion [187] for CYT_02 [106] using a Livox MID40 scanner in the left column and KITTI Tracking [58] sequence 19 in the right column. *Top:* Integrating both moving and non-moving points. *Middle:* Integrating static points based on 4DMOS [121] predictions using the receding horizon strategy, see Chap. 3 for further details. *Bottom:* Integrating static points using our volumetric belief after waiting 10 frames. Solid markers show remaining traces from moving objects in the map, and the dashed marker indicates a static area that 4DMOS removed due to false positive predictions.

was obtained with a Livox MID40 scanner, which has a smaller field of view and an irregular sampling pattern compared to rotating 3D LiDARs. This makes it harder to identify moving objects from a limited sequence of frames.

The first column shows the reconstructed surfaces from integrating all scans, including moving points. The map shows traces of moving objects, which are undesirable for planning.

The middle column shows the reconstruction after integrating the static predictions from 4DMOS using the receding horizon strategy. Although 4DMOS removes most of the dynamic traces, some moving objects remain in the map, as indicated by the solid markers in Fig. 4.4. When used with the Livox scanner, 4DMOS removes a lot of static points due to the irregular sampling pattern and the limited number of past scans as encircled by the dashed marker in Fig. 4.4. We show our final map in the right column.

Based on the high recall achieved in Sec. 4.2.4, we query the volumetric belief after fusing 10 scan and local map predictions. To additionally handle the discretization error close to the ground as explained in Sec. 4.1.6, we only integrate points for which both the map belief and the corresponding scan prediction are static. Doing so can achieve sub-voxel accuracy and remove moving points near the ground.

## 4.3 Related Work

In this chapter, we aim to segment moving objects and build and maintain a representation of our dynamic environment. We already reviewed related approaches for MOS in Sec. 3.3 and will focus in this section on how dynamic environments are represented in the literature. In the past, researchers mainly focussed on modeling the static parts of the environment.

**Visibility-based Map Cleaning.** When building maps for localization or planning, previous works aim at identifying and removing points from objects that have moved throughout the mapping process. Moving objects often lead to so-called "ghost artifacts" in the map, which we can filter out based on their visibility in a query scan: If a point measured in a query scan lies behind a point in the map, the map point most likely originates from a moving object which is no longer present in the query scan. Pomerleau et al. [144] propose a system based on this idea that estimates the dynamic state of map points. Like our proposed 4DMOS in Chap. 3, the authors fuse multiple predictions for the same point in a Bayesian filter. Kim and Kim [90] further develop Removert to remove moving objects in point cloud maps as a post-processing step. The core idea is to compare the pixel-wise visibility of points in multiple rendered range images of a point cloud map and a query scan. The central assumption is that moving objects will lead

to discrepancies between the scan and the map, and the idea is inspired by prior work on image-based change detection by Palazzolo and Stachniss [137]. However, visibility-based approaches often suffer from ambiguities for large incidence angles between the ray of a query point and the normal of the close-by map points. In this example, far-away ground points can appear in front of a query point and are falsely considered moving. Pomerleau et al. [144] address this issue by additionally checking the incidence angles.

It is also possible to check the consistency between a query scan and a map based on the traversability on the ground, assuming that moving objects are usually moving on the ground. ERASOR [104] uses the height difference when comparing the height of an object in the map to the height of the points at the same location in a query scan. Recently, Chen et al. [29] propose a pipeline to label moving objects offline automatically by further refining the dynamic points identified with ERASOR. They cluster and track the initial moving point candidates to reduce the number of false positives.

The mentioned map cleaning approaches usually require an existing environment model and are often a post-processing step run offline. Instead, our proposed approach does not require an initial map and runs online in unknown environments.

**Static Map Building.** Other researchers focused on directly modeling the static environment while building the map. For example, occupancy maps divide the space into occupied, free, and unobserved areas [178]. The static belief of voxels is updated by ray-tracing and recursive Bayesian estimation using an inverse sensor model [178]. We can use the final map to decide if a new measurement belongs to a dynamic object or not [194]. Stachniss and Burgard [171] propose an approach for 2D grid-based localization in non-static environments by clustering possible configurations of the changing environment, which improves localization. To cover the full spectrum of environmental temporal changes, Biber and Duckett [19] update a map based on different time scales.

To deal with 3D LiDAR data, Wurm et al. [199] and Hornung et al. [75] introduce OctoMap, which extends occupancy grid mapping to the 3D space by using an octree data structure. Ray-tracing approaches like OctoMap maintain a volumetric representation of the free space, making it possible to remove dynamic objects from a set of LiDAR scans [57, 156]. Pagad et al. [136] use an octree to build an occupancy grid map by first detecting ground and object points and then using ray-tracing to update voxel occupancy. They propose to use a different voxel update strategy depending on the classification to improve the quality of the occupancy map. In general, ray tracing in 3D is computationally expensive, and its performance depends strongly on the voxel size. To reduce the computational complexity, Arora et al. [9, 10] explore ground segmentation with ray-casting

to remove dynamic objects in LiDAR scans coarsely. The ground segmentation helps to reduce the number of points that are considered for ray-casting. Another prominent example is Dynablox [157], which estimates the free space of voxels based on ray-casting through a truncated signed distance grid and maintaining a dynamic state for each voxel.

In contrast to these approaches, our proposed method is independent of ray-casting because it can be computationally demanding for large, outdoor point clouds. Instead, we rely on a segmentation of a query scan and a local map using a sparse 4D convolutional neural network and integrate the estimated moving points into a probabilistic representation of the dynamic environment.

**Dynamic Environment Representation.** Modeling dynamic environments has been addressed for different tasks in the related literature. Nuss et al. [134] explicitly model the dynamic state of cells in a dynamic occupancy grid map. This dynamic state contains, for example, the velocity estimate for a cell, which we can further use to predict future occupancy [73]. Explicit modeling of moving objects is also common for scene flow analysis. Huang et al. [79] target the reconstruction of moving objects for 3D scene analysis by registering multiple point clouds and estimating offset vectors of previously classified moving points. The accumulation of multiple point clouds is supposed to deal with the sparse measurements from moving objects. Other approaches focus on long-term environmental changes, for example, due to moved furniture. Panoptic Mapping [159] identifies changes in a map on the object level based on semantic consistency. Recently, Khronos [158] demonstrates a spatio-temporal metric-semantic SLAM framework for changing environments, targeting long-term and short-term changes due to moving objects.

Instead of modeling the dynamic environment on the semantic object level, this chapter aims to close the gap between point-wise online MOS and an offline volumetric representation of the dynamic environment. We propose an approach that segments the current scan and previously received measurements into moving and non-moving points and fuses these predictions in a 3D volumetric representation. In contrast to most of the approaches mentioned above, we maintain this belief online and use it to robustify the current prediction and retrieve moving objects we falsely classified as non-moving in previous estimations for online mapping.

## 4.4 Conclusion

This chapter presented a novel approach for moving object segmentation that operates in the current LiDAR scan and local map. We used a sparse 4D CNN to jointly extract spatio-temporal features based on the discrepancy between scan

and map and the relative timestamps between points. Additionally, we suggested fusing our predictions into a probabilistic volumetric belief. This allowed us to successfully segment moving objects and recover from false positive predictions, an essential property of a spatio-temporal perception system that addresses the question of "What is moving?".

We evaluated our approach on different datasets with different sensor setups and demonstrated its effectiveness and generalization capabilities. Compared to other state-of-the-art methods, our approach showed a promising MOS performance. Additionally, we provided a generalization study that investigated how well different approaches translate to new environments and sensor configurations. We could see that both 4DMOS from Chap. 3 and the proposed MapMOS can successfully segment moving objects, whereas the projection-based baselines did not work well or do not run at all. We carried out experiments to assess the impact of our volumetric belief and showed that it improves the precision and recall of our MOS and can be effectively used to construct a static representation of the environment online. We demonstrated how we can query the updated volumetric belief to maintain a static map of the environment, which can be used for online path planning or localization.

A remaining challenge lies in the presence of points that represent long-term changes like construction sites or vegetation. The localization can fail in those cases because there will be no correspondence for these points in the map. We will address this problem in Chap. 6 by segmenting moving points and generally unstable points for tasks like localization. Additionally, our mapping experiment using a Livox sensor shows that a sensor's different field of view and scanning patterns can affect the segmentation results. However, we cannot quantitatively evaluate the MOS performance because no moving object labels are available for that dataset. Therefore, we address the necessity of such a labeled dataset in the next chapter.

# Chapter 5

# Evaluating Moving Object Segmentation Performance

Moving object segmentation using a 3D LiDAR sensor is crucial for scene understanding and identifying moving objects. Despite the availability of various types of 3D LiDAR sensors in the market, MOS research predominantly focuses on 3D point clouds from mechanically spinning omnidirectional LiDAR sensors. Meanwhile, diverse 3D LiDAR sensors have been developed, including rotating prism, solid-state, frequency-modulated continuous wave, and flash types.

In Chap. 4, we demonstrated qualitatively that our approach works well with a Livox scanner with an irregular scanning pattern. However, evaluating the performance quantitatively was impossible because no labels were available. Consequently, the community lacks a dataset with MOS labels for point clouds from diverse LiDAR sensors to properly evaluate existing approaches with respect to their applicability to different scanning patterns and fields of view. Spatio-temporal perception systems should not only be able to answer the "What is moving?" question but also be robust to the type of LiDAR sensor.

We see that existing public datasets have two limitations in assessing the generalization capabilities of MOS across heterogeneous LiDAR sensor setups. First, the aforementioned heterogeneous LiDAR datasets mainly focus on evaluating place recognition [84] or pose estimation [150] approaches without providing point-wise MOS labels. Second, while multiple datasets that provide point-wise MOS labels [16, 138] exist, these datasets are only acquired by a single omnidirectional LiDAR sensor. Thus, we still lack publicly available datasets with point-wise MOS labels for heterogeneous LiDAR setups.

This part of the thesis is based on joint work with Hyungtae Lim and Seoyeon Jang from KAIST, Republic of Korea, who were the lead researchers. I contributed to the conceptual design of the dataset and the experimental setup and

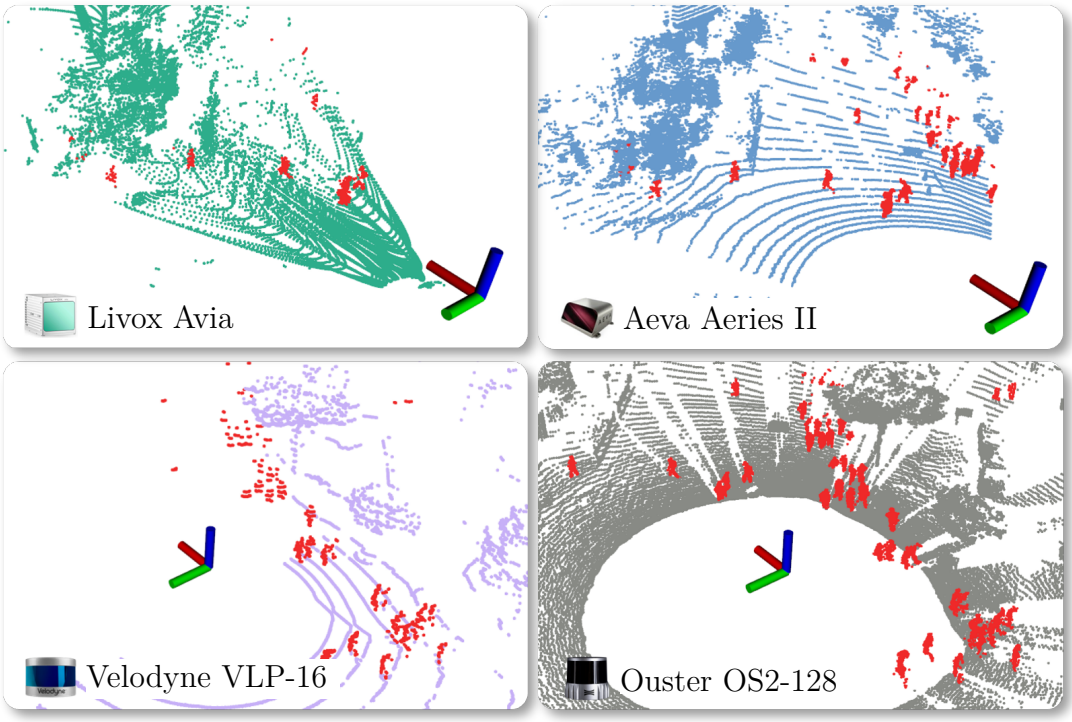



Figure 5.1: Qualitative examples of our dataset, called HeLiMOS. Our dataset provides point-wise MOS annotations for point clouds acquired by heterogeneous 3D LiDAR sensors from the HeLiPR dataset [84]. Red points indicate the annotated points from moving objects. The figure is best viewed in color.

evaluation of 4DMOS from Chap. 3 and MapMOS from Chap. 4 with the labeled heterogenous LiDAR data. We present the entire work resulting in a dataset paper [103] for the thesis to be self-consistent.

## 5.1 HeLiMOS – A Dataset for Moving Object Segmentation Evaluation

The main contribution of this chapter is HeLiMOS, a new dataset for training and evaluating MOS approaches with heterogeneous sensors with different fields of view and scanning patterns. We tackle the insufficiency of MOS labels for heterogeneous LiDAR sensors and build upon the existing HeLiPR dataset [84]. We provide MOS labels that enable the evaluation of MOS across four heterogeneous LiDAR sensor setups and visualize exemplary scans with the new moving object labels in Fig. 5.1. Furthermore, following the state-of-the-art automatic MOS labeling framework AutoMOS [29], we propose a novel instance-aware automatic labeling framework to substantially reduce the time needed for manual labeling. Finally, we set up initial benchmarks for evaluating MOS from a sensor-centric perspective and static map building from a map-centric perspective.

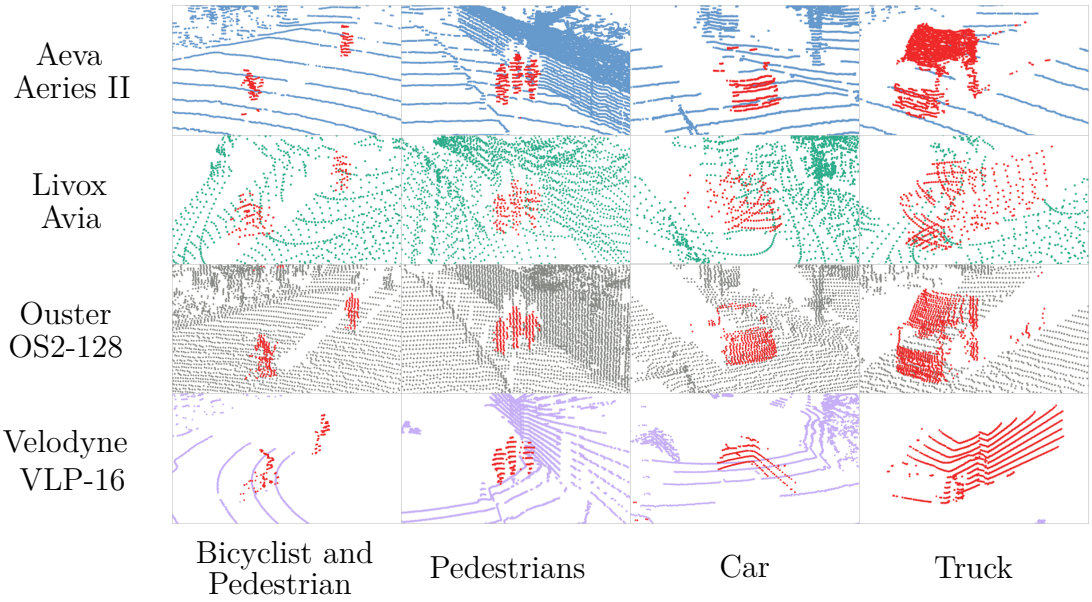



Figure 5.2: We show examples of moving objects in our dataset as red points. From top to bottom, these examples show the zoomed point clouds captured by Aeva Aeries II, Livox Avia, Ouster OS2-128, and Velodyne VLP-16. Note that even though we show the same objects, they have different patterns owing to the difference in scanning techniques and sensors' fields of view. MOS labels of (a) a bicyclist and pedestrian, (b) crowded pedestrians, (c) a car, and (d) a truck. The figure is best viewed in color.

We provide experimental results regarding our previously presented 4DMOS from Chap. 3 and MapMOS from Chap. 4 on HeLiMOS. These results emphasize the need for a new research direction for a sensor-agnostic MOS, which generally works regardless of the type of LiDAR sensors used to capture 3D point clouds. Our dataset is available at `https://sites.google.com/view/helimos`.

In summary, we make three main claims: This work (i) provides point-wise annotations for a sequence of the HeLiPR dataset, which are captured by real-world multiple heterogeneous LiDAR sensors, (ii) evaluates state-of-the-art MOS approaches with heterogeneous LiDAR sensor setups as initial benchmarks, and (iii) proposes an efficient instance-aware automatic labeling framework by employing an instance-aware static map building approach, ERASOR2 [105], and tracking-based false label filtering [81].

Our aim is that this dataset will stimulate further research, suggest new research directions, and enable reliable evaluation of novel algorithms. We also make our MOS labeling tools publicly available.

### 5.1.1 Overview

We base our dataset on the *KAIST05* sequence of HeLiPR [84], which contains various moving objects, such as buses, pedestrians, bicyclists, and cars. The dataset is acquired by four LiDAR sensors simultaneously: Velodyne VLP-16,

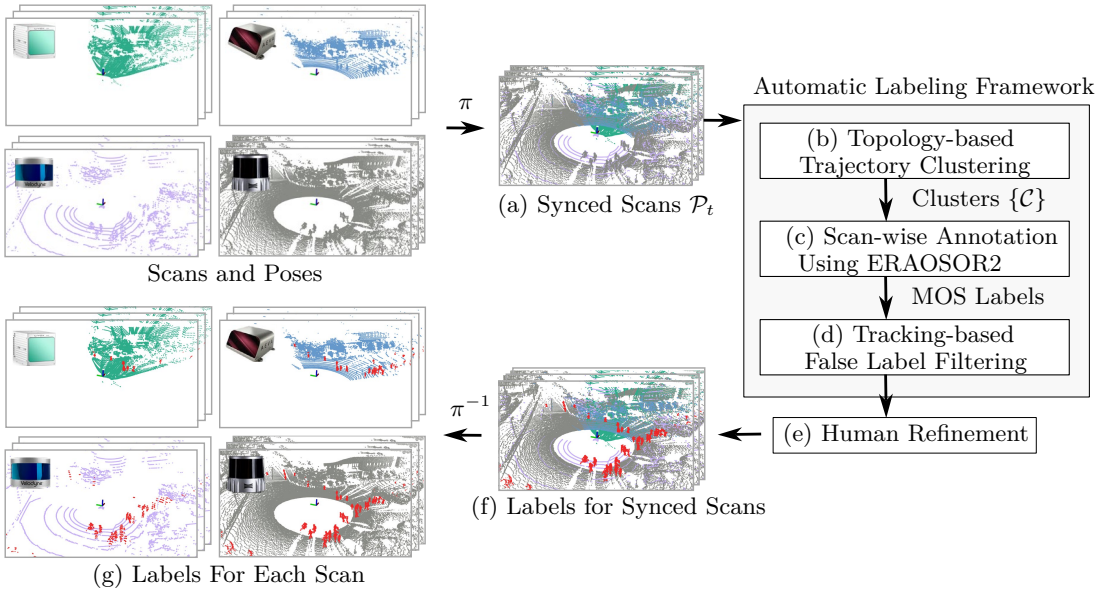



Figure 5.3: Overview of our merging-and-splitting-based labeling framework. (a) Synchronization of the point clouds from the four LiDAR sensors at a software level. (b) Next, we segment trajectories into multiple clusters. (c) For each trajectory cluster $\mathcal{C}$, we apply an instance-aware static map building, ERASOR2 [105], that produces initial scan-wise annotated labels. (d) We use tracking-based false label filtering to reduce false positive and false negative MOS labels. (e) Afterwards, we manually correct these labels. (f)-(g) Finally, we backpropagate the refined labels of synchronized scans to individual point clouds, which we denote by $\pi^{-1}$. Red points indicate the annotated dynamic points. The figure is best viewed in color.

Ouster OS2-128, Livox Avia, and Aeva Aeries II. The Velodyne VLP-16 outputs a sparse point cloud, whereas the Ouster OS2-128 provides a dense output with eight times more beams. Both sensors are omnidirectional because of internally mechanically rotating parts that enable scanning 360 degrees of the environment. The other two scanners are directional, resulting in a limited field of view. The Livox Avia scans the environment with a non-repetitive but sparse and irregular scanning pattern in contrast to the Aeva Aeries II, whose pattern offers a high density. For brevity, we denote these sensors as Velodyne ($V$), Ouster ($O$), Livox ($L$), and Aeva ($A$) in this chapter, respectively.

We aim to provide a label for each point in all scans of the different LiDAR sensors. Thus, we propose a merging-and-splitting-based efficient automatic MOS labeling framework, which we illustrate in Fig. 5.3. Our approach to labeling mainly consists of four steps. First, by transforming them into the Ouster frame, we accumulate four point clouds from the four LiDAR sensors whose timestamps are closest to each other. By doing so, we synchronize the point clouds of these four LiDAR sensors at a software level, resulting in the accumulated point cloud $\mathcal{P}_t$. Second, we cluster the accumulated scans based on their topology, such as intersections, and refine the individual poses as explained in Sec. 5.1.2. In the same step, we obtain initial MOS labels using a static mapping approach as presented

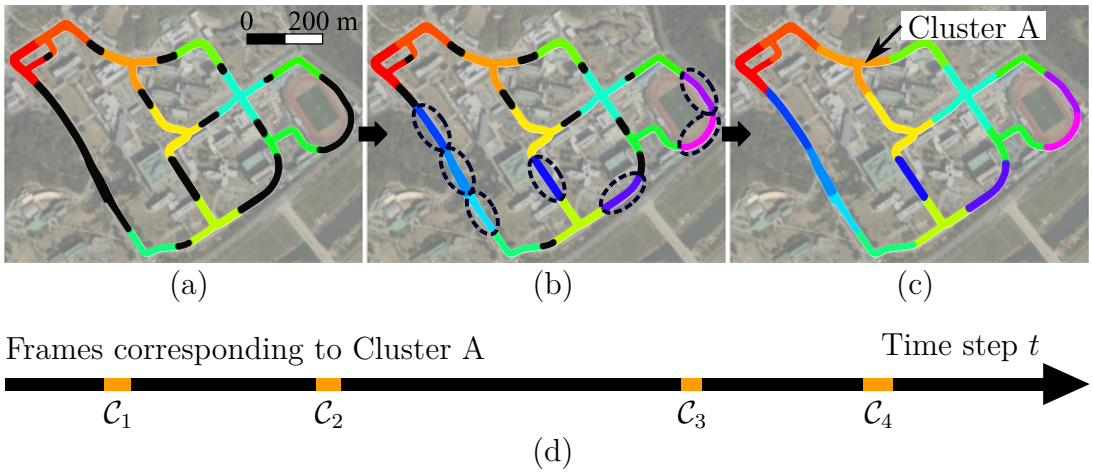



Figure 5.4: Procedure of our topology-based trajectory clustering. The black trajectory indicates unclustered frames; each color represents a different cluster. The figure is best viewed in color. (a) First, we prioritize intersections because these scenes will likely have multiple revisits. The colors indicate different clusters. (b) Next, we cluster the frames from revisited places that are not intersections and consecutive frames without revisits but with sufficiently large intervals, as indicated by the black dashed circles. (c) We merge each unclustered frame into the adjacent cluster with the closest frame interval. (d) For an exemplary cluster A, we visualize the frames along the time axis with their corresponding cluster $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$.

in Sec. 5.1.3 and correct the labels using multi-object tracking in Sec. 5.1.4. Third, we refine the labels under human supervision. In the last and fourth step, we backpropagate the refined MOS labels to the individual point clouds. We explain the details in the following subsections.

### 5.1.2 Trajectory Clustering and Pose Correction

In recent static map building approaches [104, 105], discrepancies in geometry or occupancy between individual scans and the map have often been used to estimate the dynamic points in the scans. However, these approaches heavily rely on the assumption that the given poses are accurate and, thus, the scans are sufficiently aligned with each other. Unfortunately, although reference poses are provided, undesirable errors exist in the poses for revisited scenes, i.e., loop-closed scenes. These pose errors probably stem from global navigation satellite system errors or sensor odometry drift. In our particular case, pose errors also arise from the initial alignment of the four point clouds because the point clouds were not initially synchronized at the hardware level. Consequently, these errors make automatic labeling incorrectly classify static points as dynamic points, leading to false positives and negatives. To address this issue, we divide the trajectory with poses corresponding to $\mathcal{P}_t$ with $t$ being the timestamp of the scan into multiple clusters and correct their poses to align their reference frames.

We propose a topology-based trajectory clustering that prioritizes revisited

sections, likely to have inherent pose errors owing to the time differences between scans captured during initial visits and those upon revisiting. Time discrepancies can lead to pose drift, which may not be fully minimized even after a pose graph optimization. As illustrated in Fig. 5.4, our trajectory clustering follows three steps. First, we identify areas, such as intersections or places where left/right turns occur, by examining the yaw differences within the trajectory and grouping neighboring frames into a cluster based on their locations. Second, we cluster the frames from revisited places that are not intersections and consecutive frames without revisits but with sufficiently large intervals, respectively. Finally, we merge the remaining unclustered frames into the adjacent cluster with the closest frame index.

Formally, let $\mathcal{C}$ be a cluster of the trajectory and the $i$-th consecutive frame set (or a subcluster) in $\mathcal{C}$ be $\mathcal{C}_i$, which satisfies $\mathcal{C} = \bigcup_{n=1}^{N_c} \mathcal{C}_i$, as visualized in Fig. 5.4 (d); $N_c \geq 1$ denotes the number of the subclusters. We correct the poses corresponding to frames within the same cluster to minimize errors between the reference frames for each subcluster. Based on the assumption that the poses in $\mathcal{C}_i$ are locally consistent, we use a reference frame $f_i$ for each subcluster $\mathcal{C}_i$. We can express each frame $f_t$ in the cluster relative to its reference using the transformation matrix ${}^{f_i}\mathsf{T}_{f_t}$. Then, we define the $i$-th submap $\mathcal{M}_i$, which corresponds to $\mathcal{C}_i$, as follows:

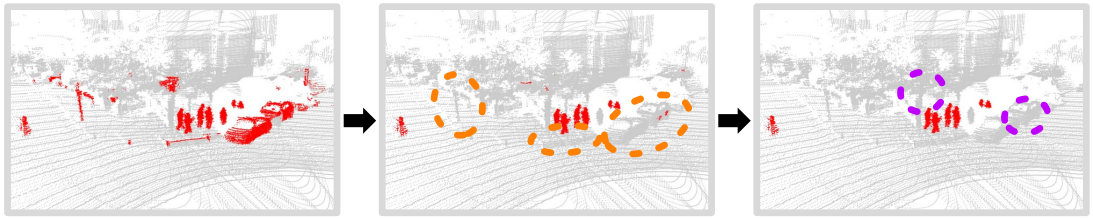
$$\mathcal{M}_i = \nu \left( \bigcup_{t \in \mathcal{C}_i} \nu \left( \left\{ {}^{f_i}\mathsf{T}_{f_t} \mathbf{p} \mid \mathbf{p} \in \mathcal{P}_t \right\} \right) \right), \tag{5.1}$$

where $\nu(\cdot)$ denotes a voxel downsampling function with the voxel size $\nu_{\text{map}}$, $\mathcal{P}_t$ is the synchronized scan whose origin is the local frame $f_t$, and ${}^{f_i}\mathsf{T}_{f_t}\mathbf{p}$ means that we transform a homogeneous point $\mathbf{p}$ expressed in the local frame $f_t$ into the reference frame $f_i$.

Finally, to locally unify the coordinate system into the reference frame of $\mathcal{M}_1$, i.e., frame $f_1$, we perform a submap-to-submap iterative closest point alignment between $\mathcal{M}_1$ and $\mathcal{M}_i$ to estimate the relative transformation ${}^{f_1}\hat{T}_{f_i}$. We update the transformation matrix for each local frame $f_t$ in $\mathcal{C}_i$ as ${}^{f_1}\hat{T}_{f_t} = {}^{f_1}\hat{T}_{f_i}\,{}^{f_i}\mathsf{T}_{f_t}$, respectively. Thus, we perform the iterative closest point algorithm [18] $N_c - 1$ times for each cluster.

### 5.1.3 Instance-Aware Initial Data Annotation

Next, we take the corrected poses and corresponding synchronized scans of $\mathcal{C}$ as inputs and run our instance-aware annotation pipeline to generate initial scan-wise MOS labels by utilizing instance segmentation information [105], which corresponds to Fig. 5.3 (c). The previous automatic labeling approach by Chen et



(a) (b) (c)

Figure 5.5: The annotation results in our proposed labeling framework. Red points denote the annotated dynamic points, while gray points represent points estimated to be static. The figure is best viewed in color. (a) The initial result obtained by using ERASOR2. (b) Refined annotation through our tracking-based filtering. Orange dashed circles indicate that we successfully rejected false positive points. (c) Final annotation after human supervision. Purple dashed circles highlight the refined areas by a human labeler.

al. [29] employed ERASOR [104] to initially annotate MOS labels, and then clustering was applied, which was referred to as the *detect-then-cluster* scheme. As ERASOR does not account for instance information, it potentially fails to identify all dynamic points from a moving object, considering some partial dynamic points as static.

In contrast, ERASOR2 [105] is a *cluster-then-detect* approach. It first performs instance segmentation, followed by dynamic point removal at the instance level by checking geometrical discrepancies between each scan and the map cloud to determine which regions are temporarily occupied. Doing so can generate more accurate and reliable MOS labels.

### 5.1.4 Tracking-Based False Label Filtering and Human Refinement

The static map building approach-based automatic labeling will likely remove dynamic points aggressively because static map building approaches were initially designed to preserve definite static points for performing localization or navigation. We show this in Fig. 5.5 (a), where many static points are wrongly classified as dynamic points at the scan level. To address this issue, we leverage multi-object tracking-based filtering [81] similar to Chen et al. [29], who also employ tracking-based filtering primarily to reject false positive points. However, we propose a bounding box augmentation that reduces the number of false negative points. We augment additional bounding boxes in the frames where tracking is temporarily lost by interpolating the centroids of bounding boxes tracked in the previous frame and next frame. Subsequently, we also classify points within these augmented bounding boxes as dynamic. As a result of our tracking-based filtering, we obtain more refined MOS labels without human effort, as shown in Fig. 5.5 (b). Nevertheless, these procedures do not ideally reject all false

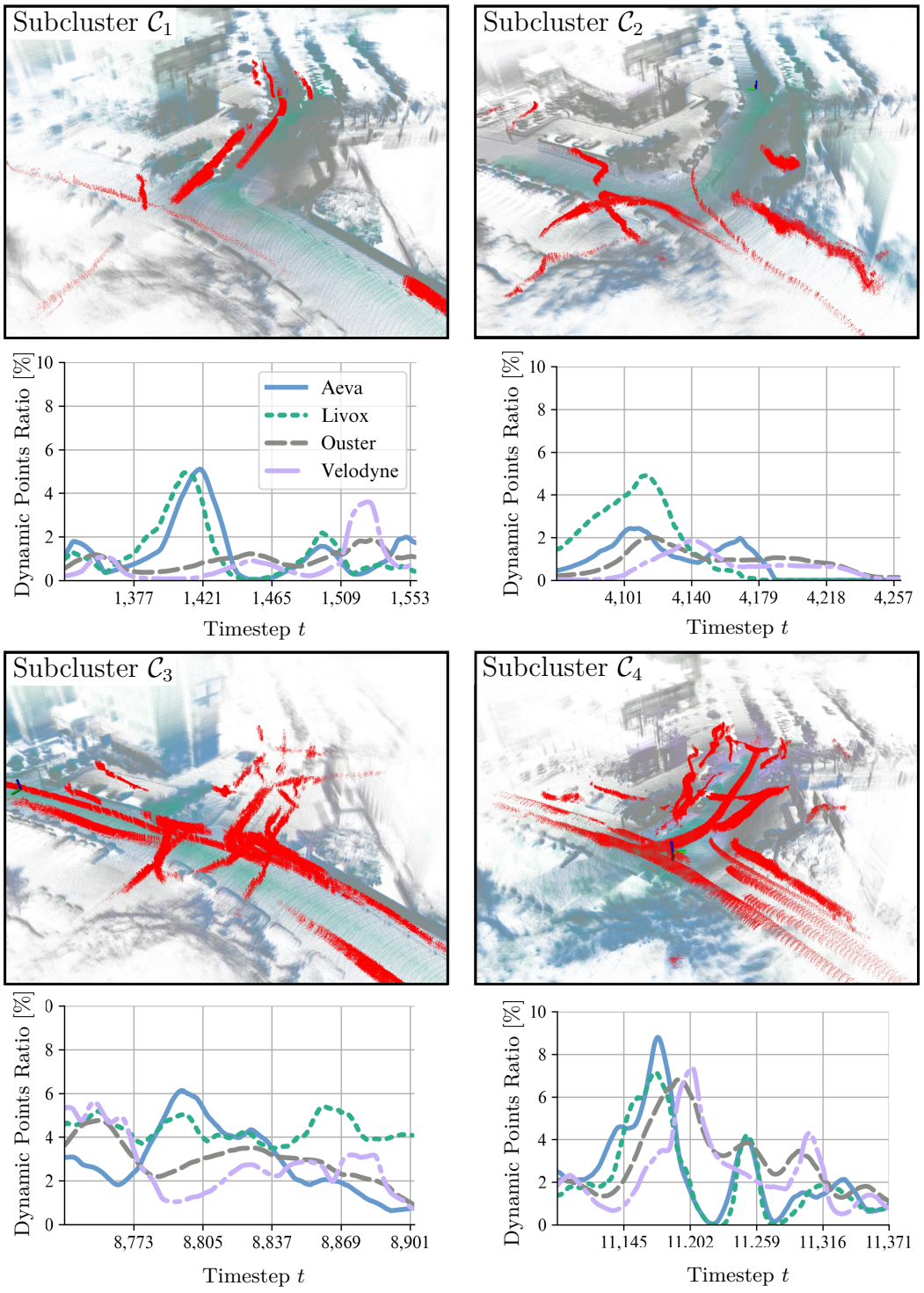



Figure 5.6: Ratios of labeled dynamic points over the total points in the $t$-th scan and labeled subclusters corresponding to $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$ from Fig. 5.4 (d). We visualize the points from all sensors in different colors and show the labeled moving objects in red. All clusters belong to the same physical location, which has been visited multiple times in the sequence and shows different moving objects. Note the varying dynamic point ratio over time, which differs between sensor types and clusters. The figure is best viewed in color.

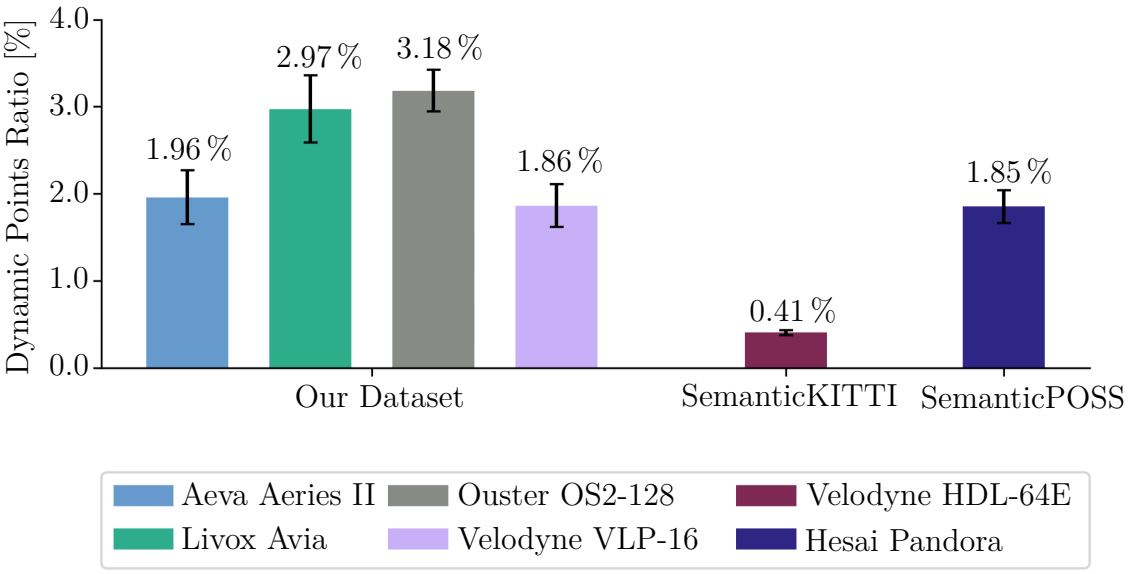



Figure 5.7: Comparison of the ratio of dynamic points with other datasets. The numbers on the bars represent the average ratios, while the black error bars indicate the standard deviations. We counted the points labeled as moving objects to calculate the dynamic points ratio of SemanticKITTI [16]. As shown in Tab. 5.5, the SemanticPOSS [138] wrongly classifies parked vehicles as moving objects. Thus, we filter them out for a fair comparison using our tracking-based filtering and only use the actual moving objects for the dynamic points ratio calculation.

positives and false negatives. Therefore, as a final stage, we perform a human-in-the-loop refinement process to enhance the quality of the MOS labels, as depicted in Fig. 5.5 (c).

### 5.1.5 Data Statistics and File Structure

Our dataset provides a total of 12,188 labeled point clouds. Each MOS label follows the SemanticKITTI MOS benchmark format, so it consists of three classes: *unlabeled*, *static*, and *dynamic*. The point clouds in our dataset are from the HeLiPR dataset, initially designed for place recognition tasks to evaluate whether a robot revisits the same location. For this reason, our HeLiMOS inherits characteristics of the HeLiPR dataset and thus leverages two notable attributes: (a) the inclusion of several revisited scenes and (b) a substantially higher ratio of dynamic points.

As shown in Fig. 5.6, dynamic point ratios and patterns vary substantially over time, even though scans are acquired in the same place. For instance, in the previously mentioned subclusters, $\mathcal{C}_3$ and $\mathcal{C}_4$ contain more moving objects and more complex trajectory patterns, resulting in higher dynamic points ratios compared with $\mathcal{C}_1$ and $\mathcal{C}_2$. In addition, the appearance of static scenes differs due to each sensor's different fields of view, which measure various parts of the

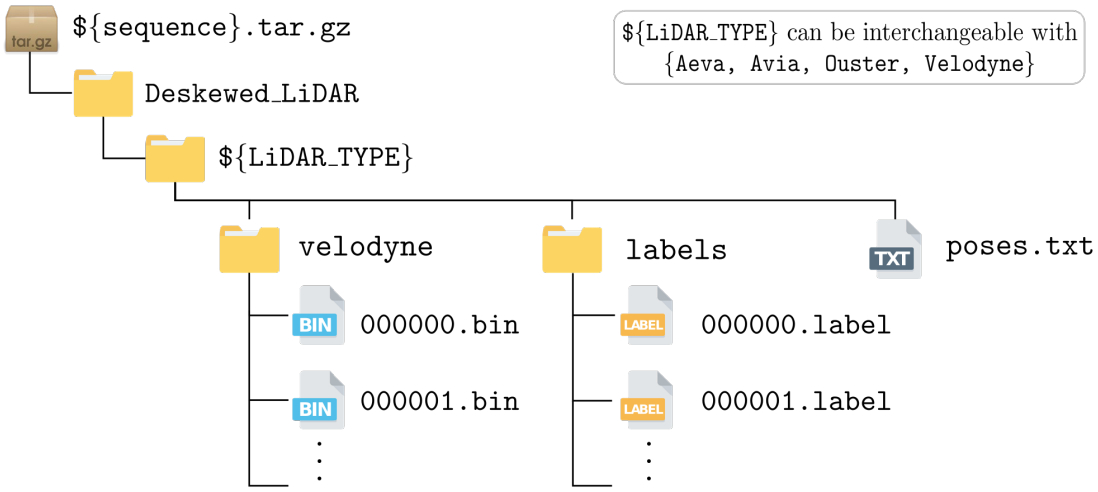



Figure 5.8: File structure of our dataset, which follows the SemanticKITTI format [16]. Despite being misleading, we keep the folder name *velodyne* instead of *scans* to be compatible with other datasets, such as SemanticPOSS [138]. Pose information is from the original dataset [84].

environment depending on the driving direction, which is visible in Fig. 5.6. Because it is crucial for MOS approaches to robustly distinguish moving objects regardless of changes in the dynamic points ratios or moving object trajectory patterns, our dataset can provide an opportunity to evaluate the generalization capabilities of MOS across diverse patterns in the same scene.

Furthermore, note that the most distinctive feature of our dataset is that it not only has higher dynamic points ratios than existing MOS datasets but also has MOS labels of four heterogeneous LiDAR sensors. As presented in Fig. 5.7, our dataset shows consistently higher average dynamic points ratios across all LiDAR sensors compared with the SemanticKITTI [16] and SemanticPOSS [138] datasets.

Therefore, using our dataset, researchers can evaluate the generalization capabilities of MOS approaches in an environment not seen during training and with different LiDAR sensors. As presented in Fig. 5.8, the file structure of our dataset follows the SemanticKITTI format [16] to support compatibility with existing SemanticKITTI dataloaders. We deskewed all the laser scans and then saved them by utilizing the HeLiPR Pointcloud Toolbox available at `https://github.com/minwoo0611/helipr-Pointcloud-Toolbox`.

To provide a fixed setup for training and evaluating MOS approaches, we split the dataset into training, validation, and test sets with ratios of 68%, 16%, and 16%, respectively. Note that we do not randomly sample the frames; instead, we designate specific sequential frames from the revisited scenes, for example, $\mathcal{C}_3$ or $\mathcal{C}_4$ in Fig. 5.6, for the validation and test sets.

## 5.2 Experimental Evaluation

This work mainly focuses on providing point-wise MOS labels for evaluating the generalization capabilities of MOS in heterogeneous LiDAR sensor setups. In addition, we can use our dataset to evaluate the performance of static map building approaches. Thus, we present three experiments utilizing our dataset to support our three essential claims, which are: We (i) provide point-wise annotations for a sequence of the HeLiPR dataset, which are captured by real-world multiple heterogeneous LiDAR sensors, (ii) evaluate state-of-the-art MOS approaches with heterogeneous LiDAR sensor setups as initial benchmarks, and (iii) propose an efficient instance-aware automatic labeling framework by employing an instance-aware static map building approach, ERASOR2 [105], and tracking-based false label filtering [81].

To back up the first and second claim, we will evaluate the MOS performance of the models from Chap. 3 and Chap. 4 which we trained on the SemanticKITTI dataset [16] on our new labeled data in Sec. 5.2.2. To further test their generalization capabilities, we train and evaluate them on different sensor setups across heterogeneous LiDAR sensors in Sec. 5.2.3. Lastly, we back up our last claim in Sec. 5.2.4. We demonstrate our automatic labeling performance to support the rationale behind our choice to use ERASOR2 and the presented tracking-based filtering. We could not evaluate these novel experiments using existing datasets, which shows the necessity of our heterogeneous LiDAR MOS dataset and our automatic labeling framework.

### 5.2.1 Experimental Setup

In the first experiment, we use our MOS models from Chap. 3 and Chap. 4, which we trained on the SemanticKITTI dataset [16]. A 64-channel omnidirectional LiDAR sensor captured the training data, and we quantitatively evaluated their inference results on HeLiMOS using all the labels. In the second experiment, we train our MOS approaches on one type of LiDAR sensors and then test on heterogeneous LiDAR sensors, i.e., training with point clouds from directional LiDAR and testing with those from omnidirectional LiDAR sensors, or vice versa, to examine performance variations across different LiDAR types. We use the mean $\mathrm{IoU}_{\mathrm{MOS}}$ for MOS [121] as a quantitative metric.

For the third experiment, we evaluate the modules of our labeling framework and existing approaches by computing the preservation rate (PR), rejection rate (RR), and $\mathrm{F}_1$ score [104, 105] using the synchronized scans, i.e., $\mathcal{P}_t$, as inputs.

| Method | Directional | | Omnidirectional | | Total |
|---|---|---|---|---|---|
| | *L* | *A* | *O* | *V* | |
| 4DMOS, online | 52.08 | 54.01 | 64.17 | 4.69 | 43.74 |
| 4DMOS, delayed | 58.99 | 58.30 | 70.44 | 5.41 | 48.28 |
| MapMOS, Scan | 58.93 | 63.15 | 81.43 | 4.33 | 51.96 |
| MapMOS, Volume | **62.70** | **66.58** | **82.87** | **5.77** | **54.48** |

Table 5.1: Mean $\text{IoU}_{\text{MOS}}$ of MOS approaches trained on the SemanticKITTI dataset to evaluate generalization capabilities in terms of both environmental changes and LiDAR sensor variations (*L*: Livox Avia, *A*: Aeva Aeries II, *O*: Ouster OS2-128, and *V*: Velodyne VLP-16).

We define the metrics as

$$\text{PR} = \frac{\#\text{ of preserved static voxels}}{\#\text{ of total static voxels}} \tag{5.2}$$

$$\text{RR} = 1 - \frac{\#\text{ of remaining dynamic voxels}}{\#\text{ of total dynamic voxels}} \tag{5.3}$$

$$\text{F}_1 = 2\frac{\text{PR} \cdot \text{RR}}{\text{PR} + \text{RR}}, \tag{5.4}$$

where higher results are better. Note that the terms preservation and rejection rate in Eq. (5.2) and Eq. (5.3), respectively, were originally introduced in [104], and we can interpret them as the recall rate of the voxel-wise static and dynamic classification, respectively. We use the original notation for the sake of completeness. For simplicity, we refer to each sensor type used in our dataset as *L*, *A*, *O*, and *V*, respectively, as described in Sec. 5.1.

### 5.2.2 Environmental Changes and LiDAR Sensor Variations

First, we evaluate the generalization capabilities of MOS approaches in environments not seen during training and to different types of LiDAR sensors. We employ our previously presented 4DMOS from Chap. 3 and MapMOS from Chap. 4, which we can directly apply to other LiDAR setups and which have shown strong generalization capabilities. For 4DMOS, we report the results for an online prediction where we take the prediction for the current scan as the final prediction and the delayed version, which fuses multiple predictions as explained in Sec. 3.1.5. In the case of MapMOS, we report both the result for the current scan's segmentation and the performance using our volumetric belief as explained in Sec. 4.2.2. We use KISS-ICP [188] to estimate the odometry and to align the scans in a standard coordinate frame for both approaches.

| Method | Training Data | Directional | | | Omnidirectional | | | Total |
|---|---|---|---|---|---|---|---|---|
| | | $L$ | $A$ | Avg | $O$ | $V$ | Avg | |
| 4DMOS (Delayed) | $L$+$A$ | 72.82 | 81.58 | 77.20 | 71.50 | 47.65 | 59.58 | 68.39 |
| | $O$+$V$ | 64.74 | 72.73 | 68.74 | 80.50 | 56.09 | 68.30 | 68.52 |
| | All | 73.72 | 84.80 | 79.26 | 82.70 | **57.64** | **70.17** | **74.72** |
| MapMOS (Volume) | $L$+$A$ | 71.84 | 83.65 | 77.75 | 77.06 | 25.62 | 51.34 | 64.54 |
| | $O$+$V$ | 72.16 | 80.71 | 76.44 | **86.74** | 49.13 | 67.94 | 72.19 |
| | All | **73.82** | **85.60** | **79.71** | 84.93 | 32.00 | 58.47 | 69.09 |

Table 5.2: Mean $\text{IoU}_{\text{MOS}}$ of MOS approaches when trained solely on data from specific LiDAR sensors. The bold texts denote the best performance among all trials, and the underlined numbers indicate the best results for each method across different training data scenarios ($L$: Livox Avia, $A$: Aeva Aeries II, $O$: Ouster OS2-128, and $V$: Velodyne VLP-16).

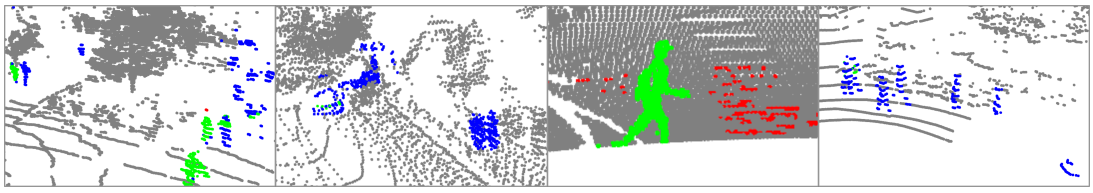
The results in Tab. 5.1 suggest that all approaches are robust to environmental changes due to the excellent performance on the Ouster data $O$, which is the most similar sensor to the 64-channel sensor used to acquire SemanticKITTI and train the approaches. In contrast, we observe a performance degradation for the directional LiDARs $L$ and $A$ compared to the $O$ results. Third, when using sparser point clouds as inputs, the performance of our MOS approaches degrades more, as shown for $V$ in Tab. 5.1, because these MOS approaches heavily depend on the pose estimation of KISS-ICP to use temporal information from LiDAR sequences. Consequently, once the estimated poses are imprecise owing to the sparse point clouds, the MOS performance becomes worse.

### 5.2.3 Performance Across Heterogeneous LiDAR Sensors

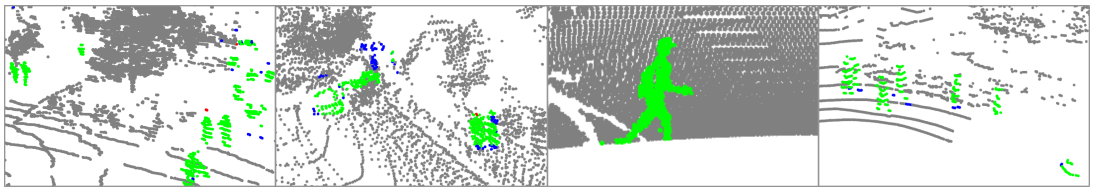
The following experiment evaluates the performance changes caused by domain shifts across different LiDAR sensor types within the same environments. The MOS models showed substantial performance improvements across all sensors after training with our dataset, as presented in Tab. 5.2 and Fig. 5.9.

In general, Fig. 5.9 qualitatively demonstrates how the segmentation results improve when training on target domain data. After training with all sensors, MapMOS successfully predicts previously missed moving objects as dynamic and is more robust against false positives.

Next, we investigate in Tab. 5.2 how well different training setups influence the results on the various sensor setups. Interestingly, unlike 4DMOS, whose performance for each test sensor type improves as we provide more diverse training data, the performance of MapMOS for Ouster OS2-128 and Velodyne VLP-16 is better when training exclusively on data from these sensors. We believe the ap-



Trained on SemanticKITTI



Trained on HeLiMOS

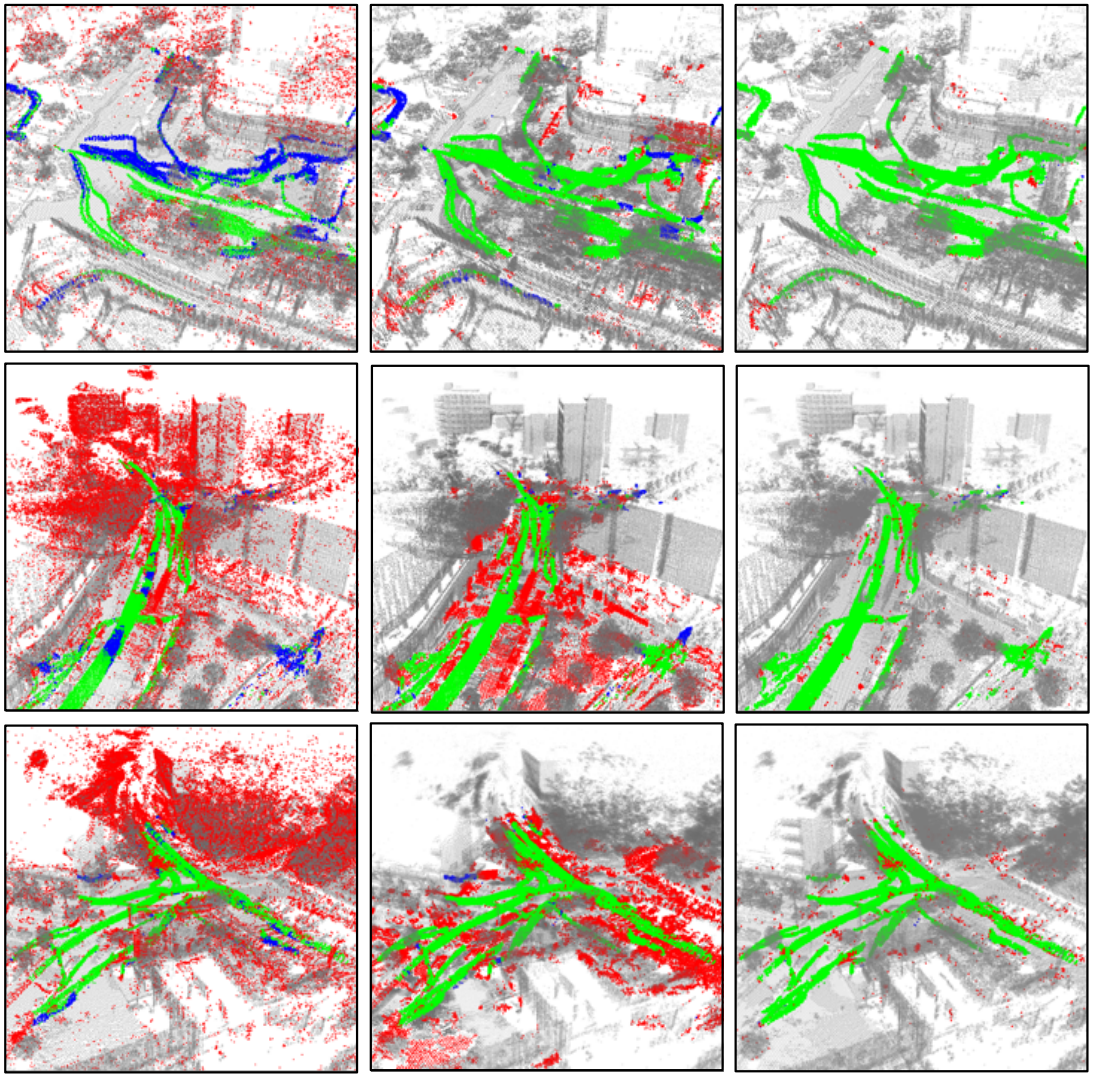
Figure 5.9: Qualitative comparison of MapMOS [123] across all sensors after training with SemanticKITTI data and with our dataset using all sensors. Green, red, and blue points indicate true positives, false positives, and false negatives, respectively. The fewer red and blue points there are, the better. The figure is best viewed in color.

proach slightly overfits the target domain in this case. Overall, MapMOS shows better results due to better generalization capabilities [123] by taking both a local map and a current scan as inputs. In particular, we employ the local map to reduce the geometrical differences between each scan from different sensors by accumulating scans over time. Unfortunately, scans from $V$ are too sparse to estimate the relative poses precisely. Therefore, an error in the pose estimation leads to an inconsistent local map, resulting in performance degradation, whereas 4DMOS only considers a limited time horizon. As a result, MapMOS showed lower $\text{IoU}_{\text{MOS}}$ with $V$ in Tab. 5.2. Nevertheless, MapMOS was on par with 4DMOS regarding total mean $\text{IoU}_{\text{MOS}}$ and showed the highest performance in $L$, $A$, and $O$. These two experiments imply that there is still room for improvement in making existing MOS methods operate sensor-agnostic.

### 5.2.4 Automatic Labeling Performance

Finally, we demonstrate the superiority of our automatic labeling framework. A direct comparison to the baseline AutoMOS [29] is unfair, because AutoMOS relies on ERASOR [104], which ERASOR2 [105] outperforms, see Tab. 5.3. Therefore, we separately evaluate the performance of (a) static map building approaches for initial MOS labeling and (b) tracking-based filtering. We use the final ground truth labels after the human refinement for evaluation.

First, to explicitly show the performance differences in static map building, we conducted experiments on the three most crowded scenes, which we visualize in Fig. 5.10 and for which we report the quantitative results in Tab. 5.3. We can see in Tab. 5.3 that ERASOR2 shows a substantially higher $\text{F}_1$ score compared



(a) Removert [90] (b) ERASOR [104] (c) ERASOR2 [105]

Figure 5.10: Qualitative comparison of static map building results produced by state-of-the-art methods on our dataset using synchronized scans. Green, red, and blue points indicate true positives, false positives, and false negatives, respectively. The fewer red and blue points there are, the better. The figure is best viewed in color.

with Removert [90], a range image-based approach. We can also see this from the large portion of false positives and negatives in Fig. 5.10 (a). The MOS labeling module ERASOR, initially used in AutoMOS, also shows a lower PR and RR than ERASOR2. The reason is that ERASOR directly subtracts the estimated dynamic points from the map cloud without considering instance information, incorrectly estimating static points as dynamic while leaving some dynamic points on the map, as shown in Fig. 5.10 (b). In contrast, by leveraging instance information, ERASOR2 precisely rejects traces of moving objects in the map cloud while preserving most static points, as presented in Fig. 5.10 (c). This implies that ERASOR2 consistently labels the dynamic points within each scan.

| Frame range | Method | PR [%] | RR [%] | $F_1$ score |
|---|---|---|---|---|
| 2,250-2,500 | Removert [90] | 85.072 | 47.170 | 0.607 |
| | ERASOR [104] | 95.325 | 82.490 | 0.884 |
| | ERASOR2 [105] | **99.522** | **95.339** | **0.974** |
| 8,600-8,800 | Removert [90] | 80.581 | 71.965 | 0.760 |
| | ERASOR [104] | 91.610 | 84.290 | 0.878 |
| | ERASOR2 [105] | **99.530** | **93.740** | **0.965** |
| 11,070-11,300 | Removert [90] | 82.852 | 81.301 | 0.821 |
| | ERASOR [104] | 93.969 | 89.955 | 0.919 |
| | ERASOR2 [105] | **99.676** | **97.175** | **0.984** |

Table 5.3: Comparison of static map building approaches for the most crowded frame sequences in our dataset (PR: Preservation Rate, RR: Rejection Rate).

| Method | $IoU_{MOS}$ |
|---|---|
| ERASOR2 [105] | 21.8 |
| ERASOR2 + Tracking-based filtering in AutoMOS [29] | 53.4 |
| ERASOR2 + Our tracking-based filtering | **61.2** |

Table 5.4: Mean $IoU_{MOS}$ before and after applying tracking-based filtering approaches.

Second, as shown in Tab. 5.4, the tracking-based filtering in AutoMOS shows a substantial increase in performance, which indicates that it substantially reduces the number of false positive points. However, as described in Sec. 5.1.4, it cannot reduce the same number of false negatives. In contrast, by introducing augmented bounding boxes, our approach can further suppress the impact of false negative points. By doing so, our developed filtering shows the highest $IoU_{MOS}$.

We conclude that the combination of ERASOR2 and our proposed tracking-based filtering is a suitable automatic labeling framework to help human labelers reduce the time needed for manual labeling.

## 5.3 Related Work

Over the past decade, numerous impactful datasets and benchmarks for autonomous vehicles have been released. A renowned one is the KITTI dataset [58], which provides both odometry and various perception benchmarks. Influenced by this, existing datasets have evolved in two main directions in terms of (a) odometry and place recognition and (b) perception, as we present in Tab. 5.5.

**Odometry Datasets and Benchmarks.** From the viewpoint of odometry and place recognition, the KITTI dataset has a few loop-closing situations

and environmental changes, with only a single omnidirectional LiDAR sensor for a short data collection span. To provide more challenging environments for odometry and place recognition tasks [205], Carlevaris et al. [25], Lu et al. [113] and Jeong et al. [82] propose the NCLT, Apollo-SouthBay, and Complex Urban datasets, respectively, acquired by multiple 2D and 3D omnidirectional LiDAR sensors. Kim et al. [88] release the MulRan dataset focusing on multi-modal long-term mapping and place recognition by employing a 3D LiDAR sensor and an omnidirectional radar sensor.

As a further study, Carballo et al. [24] come up with the LIBRE dataset, consisting of point clouds from ten LiDAR sensors. However, all the deployed sensors are omnidirectional, implying that all the sensors have similar scanning patterns and fields of view due to the same measurement principle. Thus, this dataset is unsuitable for testing whether an algorithm generally works well in a heterogeneous LiDAR sensor setup. To tackle this problem, Qingqing et al. [150] release the TIERS dataset, which consists of three omnidirectional LiDAR sensors and three directional LiDAR sensors. Similar to the TIERS dataset, Jung et al. [84] propose the HeLiPR dataset, which is acquired by two omnidirectional LiDAR sensors and two directional LiDAR sensors, including under-researched channels, i.e., reflectivity, near-infrared, and radial velocity. Unfortunately, these datasets only aim to evaluate odometry and place recognition without any point-wise MOS labels, as summarized in Tab. 5.5. Therefore, we cannot directly use them to evaluate the performance of MOS approaches.

**Perception Datasets and Benchmarks.** Behley et al. [16] release the SemanticKITTI dataset, a pioneering work that first provides point-wise semantic and instance labels for 3D sequential point clouds. Since the semantic labels differentiate between moving and non-moving semantic classes, Chen et al. [28] remap them into moving and non-moving and proposed the SemanticKITTI MOS benchmark. Inspired by SemanticKITTI, Pan et al. [138] come up with the SemanticPOSS dataset, which shares the same labeling protocol with SemanticKITTI to support compatibility with existing SemanticKITTI dataloaders. While these datasets provide abundant point-wise labels, the SemanticKITTI and SemanticPOSS are only captured by a single omnidirectional LiDAR sensor.

Caesar et al. [23] publish the nuScenes dataset, which supports perception tasks in 1,000 short sequences of 20 seconds duration using an omnidirectional 32-beam LiDAR. It is possible to remap the labels of annotated keyframes into moving and non-moving as we did in Sec. 4.2.3, but these are not available for all scans. The same holds for the DOALS [142], which provides moving object labels only for a few keyframes.

Xiao et al. [201] and Chen et al. [27] release the PandaSet and WOMD-LiDAR datasets, respectively, which contain point clouds from multiple heterogeneous

| | Dataset | Year | Multiple LiDARs | Hetero-geneous | Point-wise MOS labels |
|---|---|---|---|---|---|
| Odometry | KITTI [58] | 2012 | ✗ | ✗ | ✗ |
| | NCLT [25] | 2016 | ✗ | ✗ | ✗ |
| | Oxford Robotcar [116] | 2017 | ✓ | ✗ | ✗ |
| | Complex Urban [82] | 2019 | ✓ | ✗ | ✗ |
| | Apollo-SouthBay [113] | 2019 | ✗ | ✗ | ✗ |
| | MulRan [88] | 2020 | ✗ | ✗ | ✗ |
| | LIBRE [24] | 2020 | ✓ | ✗ | ✗ |
| | TIERS [150] | 2022 | ✓ | ✓ | ✗ |
| | HeLiPR [84] | 2023 | ✓ | ✓ | ✗ |
| Perception | KITTI [58] | 2012 | ✗ | ✗ | ✗ |
| | SemanticKITTI [16] | 2019 | ✗ | ✗ | ✓ |
| | SemanticPOSS [138] | 2020 | ✗ | ✗ | △ |
| | nuScenes [23] | 2020 | ✗ | ✗ | ✓ |
| | DOALS [142] | 2021 | ✗ | ✗ | ✓ |
| | PandaSet [201] | 2021 | ✓ | ✓ | ✗ |
| | WOMD-LiDAR [27] | 2023 | ✓ | ✓ | ✗ |

Table 5.5: Comparison between existing 3D point cloud datasets and our proposed dataset. The term "Heterogeneous" indicates whether a dataset comprises mechanically spinning omnidirectional and directional LiDAR sensors. We consider 2D and 3D omnidirectional LiDAR sensors to be homogeneous to each other. The symbol △ indicates that the dataset provides point-wise labels; however, it incorrectly labels parked vehicles as moving objects by naïvely considering all pedestrians and vehicles as in motion.

LiDAR sensors. However, these datasets are also unsuitable for evaluating the performance of MOS in the heterogeneous LiDAR sensor setups because they do not provide point-wise MOS labels. Therefore, to our knowledge, we propose a point-wise MOS dataset for heterogeneous LiDAR sensors, enabling the evaluation of MOS and static map building tasks across diverse LiDAR sensor setups.

**Automatic Labeling Framework.** We also propose an efficient instance-aware automatic labeling framework to substantially lessen a human labeler's annotation burden. It is challenging and time-consuming for human labelers to discern moving objects in the 3D point clouds owing to the sparse characteristics of 3D point clouds [56]. To account for this, Kim and Kim [90] develop Removert, which is a range image-based scan-wise MOS labeling approach. Furthermore, Chen et al. [29] propose an automatic labeling framework called AutoMOS. The main idea of AutoMOS is to identify dynamic candidate points using the static map cleaning approach ERASOR [104]. Similar to our presented an-

notation scheme, these points are clustered, and bounding boxes are extracted. The authors track these bounding boxes using a multi-object method developed by Weng [195].

In contrast to these prior approaches, we take instance information into account to reduce the number of false positives and thus minimize the need for manual corrections by a human labeler. Therefore, we propose instance-aware MOS annotation using ERASOR2 while accounting for the pose uncertainty in the revisited scenes via a topology-based trajectory clustering approach to obtain label information effectively.

## 5.4 Conclusion

In this chapter, we presented a novel moving object segmentation dataset for heterogeneous LiDAR sensors. To successfully depoly MOS approaches in real-world scenarios to answer the question of "What is moving?", they need to generalize to new environments and sensor configurations, or we need to re-train them based on labeled data of the target domain.

Our dataset provides point-wise MOS labels for four different LiDAR sensors, which mainly differ in their field of view and scanning pattern. Furthermore, we proposed a novel instance-aware automatic labeling framework to reduce a human labeler's time, cost, and effort when annotating LiDAR scans in large-scale scenes.

In our experimental evaluation, we trained and evaluated our previously presented approaches 4DMOS from Chap. 3 and MapMOS from Chap. 4 for MOS. More specifically, we investigated the ability of models trained on the SemanticKITTI dataset to generalize to the different sensor types in our proposed HeLiMOS. The experiments suggested that our approaches work well with new sensors with different fields of view or scanning patterns but also depend on the odometry for aligning the scans. Next, we also demonstrated how different training, validation, and test configurations across the heterogeneous sensors affect the MOS performance. Lastly, we provided insight into the performance of our automatic labeling approach with different variations of initial candidate identification methods and tracking modules.

In sum, we demonstrated the necessity of a heterogeneous LiDAR moving object segmentation dataset by suggesting new research directions toward sensor-agnostic segmentation, enabling better evaluations in this field of research. Although our dataset currently covers a single sequence, our proposed framework can be further used in the future to label additional sequences of HeLiPR or additional datasets to push further the availability of labels for evaluating the generalization of MOS approaches.

# Chapter 6

# Segmentation of Stable Points For Localization

Mobile robots increasingly operate in real-world environments that are subject to change over time. Accurate and robust localization in these environments is crucial for effectively operating autonomous mobile systems to perform path planning and obstacle avoidance tasks. Global navigation satellite system-based outdoor localization is the go-to solution that can operate without a prior map. However, sufficient connection to satellites may not always be available in specific environments due to, for example, tree foliage or large buildings. Map-based mobile robot localization utilizes onboard sensors [146, 206] like cameras or LiDARs for vehicle pose estimation by matching the sensory data with the robot's prior belief about the environment in the form of a map.

A standard method for LiDAR-based pose estimation in a given map is to use scan matching algorithms [18, 20]. In contrast to LiDAR odometry approaches like KISS-ICP that estimate a relative odometry from data within the same session, localization in a previously built map is sensitive to additional environmental changes. These changes are not only short-term, like moving objects, but also long-term, like structural changes due to construction sites or changed vegetation. Due to these changes, localization systems may fail since some scan points may not have correspondences in the map, thus leading to failure in the accurate localization of the vehicle.

One possibility to improve scan matching performance in changing environments is to segment the LiDAR scan into stable and unstable points and to utilize only the stable points for localization. Stable points typically represent elements of a stable object, such as walls, poles, light posts, and tree trunks. One approach to isolate these points is to employ handcrafted features for segmentation based on the shape of the object [155, 169]. However, such methods' robustness can be

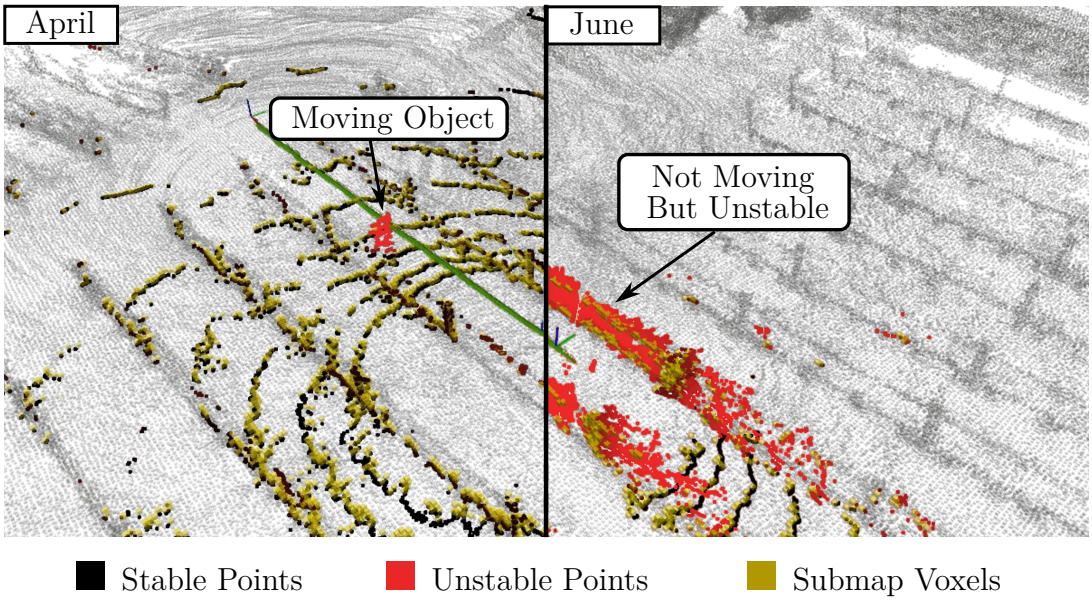



Figure 6.1: Our method segments stable and unstable points in 3D LiDAR scans, exploiting the discrepancy of scan voxels and overlapping map voxels (highlighted as submap voxels). We showcase two LiDAR scans captured during separate localization sessions within an outdoor vineyard. The scan on the left depicts the vineyard state in April, while the scan on the right reveals environmental changes in plant growth in June.

compromised by the varying density of point cloud data, leading to errors in the segmentation of stable points, thus causing a failure in the localization.

An alternative is to employ deep learning approaches [50, 125, 177] to learn to segment stable points from LiDAR scans. Despite the potential of deep learning, these methods often demand substantial amounts of manually annotated data and frequently struggle to generalize effectively to new, unseen data. To address the labeling issue, LTS-NET [77] implicitly learns the inherent stable structure in the environment in a self-supervised fashion and utilizes this structure as a landmark to improve vehicle localization in changing environments. Self-supervised training avoids an expensive manual labeling process. However, the generalization capabilities in novel scenes are poor.

In this chapter, we tackle the challenge of developing a generalizable segmentation of LiDAR data into stable and unstable points for long-term localization in environments not seen during training based on scan-to-map matching. This work goes further than the previous chapters by not only addressing the "What is moving?" question but also integrating the knowledge into a localization pipeline to improve the performance in dynamic environments. In this case, "moving" refers to points that belong to parts of the environment that have moved with respect to a prior map.

Our primary objective is to enhance the reliability of mobile robot localization in dynamic environments. To obtain a strong generalization capability of the

segmentation of stable points, we follow the previous Chap. 4 and exploit the discrepancy between scan and map points by applying sparse 4D convolutions on a joint sparse voxel grid that encompasses both scan voxels and their corresponding map voxels. This allows us to segment scan points into stable and unstable points based on a predicted long-term stability confidence score for each scan point, see Fig. 6.1.

In contrast to the task of moving object segmentation discussed in Chap. 3 and Chap. 4, the segmentation of stable points in 3D LiDAR scan data can segment both present dynamic entities like walking humans and stationary objects that may undergo positional or perceptual alterations in subsequent instances such as plant vegetation. Furthermore, unlike supervised semantic segmentation, our method does not require multiple classes to supervise the learning, and we can train it in a self-supervised manner with no manual annotation by leveraging previous observations.

Our experiments demonstrate that utilizing the stable points for localization improves the performance of scan-matching algorithms, especially in environments where changes in appearance are frequent. In line with the results of our generalization experiments in Sec. 3.2.3 and Sec. 4.2.3, our segmentation of stable points generalizes to new, unseen environments using a similar architecture.

The presented work is a joint collaboration with Ibrahim Hroob, to which we contributed equally. More specifically, I developed the segmentation pipeline and integrated MapMOS from Chap. 4 to segment stable points, whereas Ibrahim Hroob was responsible for the localization part and the experimental evaluation. For completeness, we present the entire work resulting in the joint paper [76] in this thesis.

## 6.1 Generalizable Scan-to-Map Stable Point Segmentation

The main contribution of this chapter is a novel real-time approach for segmenting stable points from a 3D LiDAR scan. Using these stable points for localization, our approach can enhance scan-to-map matching in changing environments. We achieve this by training a 4D sparse convolutional neural network in a self-supervised manner, allowing it to predict spatio-temporal features in the current scan by exploiting discrepancies between the scan and the map data.

In sum, we make two key claims: Our approach can (i) segment scan points into stable and unstable points and utilize the stable points to increase the accuracy of robot long-term localization and (ii) generalize across diverse and unseen environments, including settings not encountered during training, leading to

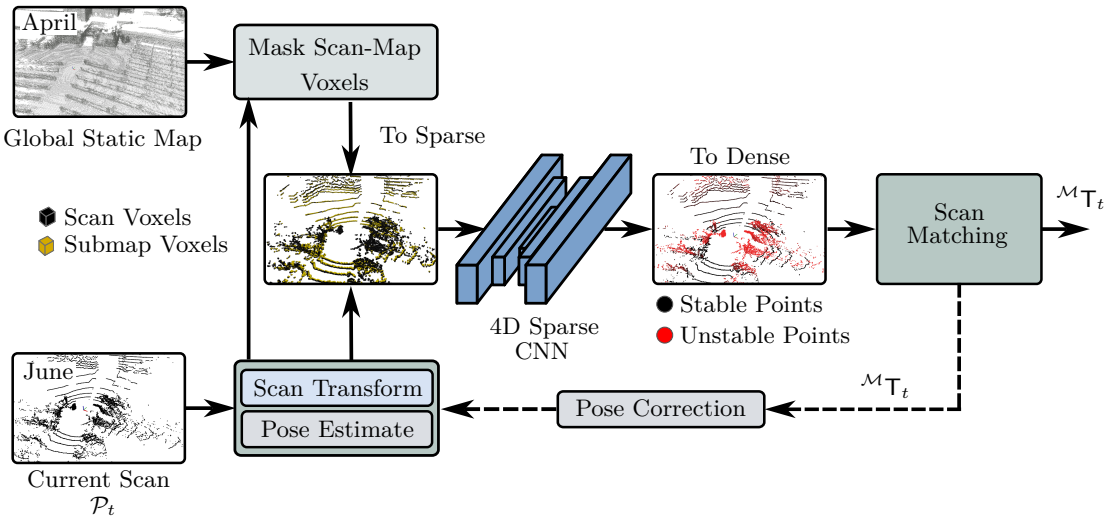



Figure 6.2: Initially, we transform the scan using an initial pose estimate. Next, we voxelize both scan and map points and extract overlapping map voxels, so-called submap voxels. We represent the scan and map voxels as a 4D sparse tensor, with the fourth dimension denoting time $t$. We then apply sparse 4D convolutions on a joint sparse voxel grid encompassing both the scan and submap points, leading to the prediction of long-term stability scores for the scan points.

improved localization performance while being suitable for online mobile robot operation. The chapter and our experimental evaluation back up these claims.

### 6.1.1 Overview

In this chapter, we propose a generalizable segmentation of stable points filter to increase the robustness of pose estimation for scan-matching algorithms in changing environments. We illustrate the pipeline in Fig. 6.2. To this end, we first transform the scan into the global map frame using an initial pose estimate, see Sec. 6.1.2. Then, we employ 4D sparse convolutions across scan and map voxels to exploit their discrepancy and increase the network's generalization capability as outlined in Sec. 6.1.3. In contrast, to Chap. 4, the local map used for localization can be large and of high density. Therefore, we do not use all map voxels, only those that overlap with scan voxels based on the initial guess. We call these map voxels "submap voxels". We train the 4D sparse CNN self-supervised, leveraging prior environmental observations to generate training labels with long-term stability, as described in Sec. 6.1.4.

### 6.1.2 Map-Based 3D LiDAR Localization

When estimating the robot's pose $^{\mathcal{M}}\mathsf{T}_t \in \mathbb{SE}(3)$ in a given map $\mathcal{M}$ and sensor reading $\mathbf{z}_t$ at time $t$, the most used localization algorithm is Monte Carlo localization [40]. It can achieve both local and global localization. Alternatively,

scan-matching algorithms such as iterative closest point [18] or normal distributions transform [20] can achieve accurate robot localization within a known map. However, unlike Monte Carlo localization, they can estimate smoother robot trajectories but require a strong guess of the initial pose and are less robust [4]. In this work, we focus on scan-matching algorithms. We proceed with the assumption of an initial estimate being available. This assumption holds for many robotics applications targeting repeated missions such as data recording and site inspection missions, where the mobile robot typically starts its operations from a fixed initial pose.

The concepts presented in this work are versatile and applicable to both the iterative closest point and normal distributions transform algorithms. However, we cannot directly use KISS-ICP from our previous chapters because it is a LiDAR odometry approach that registers each scan to a local map, which we build incrementally online. In this chapter, we aim to localize a map already built during a previous session. Therefore, we use the normal distributions transform localization framework [93], which performs unscented Kalman filter-based pose estimation [178]. The estimated pose provides a strong initial estimate for the algorithm during scan registration. We define the sensor transformation matrix to estimate at time $t$ as follows:

$$
{}^{\mathcal{M}}\mathsf{T}_t = \begin{bmatrix} R_t & \boldsymbol{t}_t \\ \mathbf{0} & 1 \end{bmatrix}, \tag{6.1}
$$

where $\boldsymbol{t}_t \in \mathbb{R}^3$ is the position and $R_t \in \mathbb{SO}(3)$ is the rotation matrix of the sensor with respect to the origin of the point cloud map $\mathcal{M}$. The normal distributions transform aims to find ${}^{\mathcal{M}}\mathsf{T}_t$ of the current scan $\mathcal{P}_t$ that maximizes the likelihood that $\mathcal{P}_t$ overlaps with the reference map $\mathcal{M}$. Without loss of generality, we omit the superscript $t$ since all the processes occur at the current timestamp $t$. We estimate the transformation matrix ${}^{\mathcal{M}}\mathsf{T}_t^*$ as follows:

$$
{}^{\mathcal{M}}\mathsf{T}_t^* = \underset{{}^{\mathcal{M}}\mathsf{T}_t}{\operatorname{argmax}} \sum_i \exp\left( -\frac{(\boldsymbol{p}_i' - \boldsymbol{\mu}_i)^\mathsf{T} \Sigma_i^{-1} (\boldsymbol{p}_i' - \boldsymbol{\mu}_i)}{2} \right), \tag{6.2}
$$

where we obtain the transformed query points $\boldsymbol{p}_i'$ in homogeneous coordinates $\mathbf{p}_i' = {}^{\mathcal{M}}\mathsf{T}_t \mathbf{p}_i$ by applying the transformation matrix ${}^{\mathcal{M}}\mathsf{T}_t$ on the homogeneous query point $\mathbf{p}_i$. The expressions $\Sigma_i$ and $\boldsymbol{\mu}_i$ are the covariance matrix and the mean of the corresponding normal distributions transform voxel for the point $\mathbf{p}_i'$ looked up in the normal distributions transform voxels of the map $\mathcal{M}$.

In a non-static environment, the LiDAR scan measurements are taken from both stable and unstable objects, expressed as

$$
\mathcal{P} = \mathcal{P}_s \cup \mathcal{P}_u. \tag{6.3}
$$

Here, $\mathcal{P}_s$ denotes the subset of points that we measure from stable objects, while $\mathcal{P}_u$ encompasses the points we associate with unstable objects. Unstable points are characterized by their lack of corresponding points in the map. Filtering out these unstable points enhances the accuracy of scan-matching algorithms by improving the data association between the current scan and the map.

### 6.1.3 Segmentation of Stable Points

Similar to the moving object segmentation in Chap. 4, we aim at exploring the spatio-temporal discrepancy between the LiDAR frame and a point cloud map to decide which points are stable, therefore enabling a generalizable setup that does not rely on additional information such as semantics. The main difference is that we do not build the point cloud map on the fly, but it is a given map our robot should use for localization.

To find the discrepancy between the scan and map point, we first start by transforming the LiDAR scan $\mathcal{P}_t = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N\}$ into the global map frame utilizing the initial pose prediction from the unscented Kalman filter ${}^{\mathcal{M}}\mathsf{T}'_t$, resulting in $\mathcal{P}'$. Then we add a timestamp $t$ to the scan and map points to form a 4D tensor with each point represented as $\boldsymbol{p}_i = [x_i, y_i, z_i, t_i]^\mathsf{T}$, where we use a fixed time $t_m$ for the map points and a fixed time $t_s$ for the transformed scan points $\mathcal{P}'$. The motivation is mainly to distinguish between scan and map points falling in the same voxels at later steps.

Subsequently, we discretize the scan and map 4D tensors into sparse 4D tensors, utilizing a predefined spatial resolution. We denote the scan and map sparse voxel grid coordinates as follows: $\mathsf{C}_S \in \mathbb{R}^{4\times n}$ and $\mathsf{C}_M \in \mathbb{R}^{4\times m}$. Here, $n$ and $m$ are the numbers of scan and map voxels, respectively. We represent each voxel coordinate using its central Cartesian position. It is important to note that we preserve the original point coordinates within their respective voxels to recover a per-point segmentation. We follow the setup from Chap. 3 and use a constant feature of 0.5 for each voxel to extract the spatio-temporal information only from the non-empty voxels represented by the sparse 4D coordinates.

Next, we merge the scan and map 4D tensors into a unified tensor. Given that scan and map voxels share the same coordinate frame, this merging process highlights the discrepancies between the scan and map voxels, revealing three possible scenarios for a voxel. Firstly, if a voxel encompasses both scan and map timestamps, it suggests an association with a stable object. Secondly, a spatial voxel exclusively containing the scan timestamp indicates a potential association with an unstable object. Lastly, if a voxel solely has the map timestamp, it suggests it is beyond the scan range or obscured by another object.

We employ a sparse CNN designed for stability inference through regression to estimate the stability confidence score for each point. This involves applying

sparse convolution to the unified scan and map sparse 4D tensor. We derive our sparse CNN based on the MinkUNet32 [32] from Sec. 2.4.2, which we also used in Chap. 4. We repurpose this network as a regression model with a specific modification to the final layer. In this adaptation, we utilize the sigmoid function to predict confidence scores for stable points, ensuring the values range between 0 and 1 for each point.

In Chap. 4, we estimate moving objects based on the scan and a local map. We use this local map for odometry estimation and clip it based on the sensor's maximum range to reduce the memory and runtime overhead. However, in this chapter, we aim to localize in a given map, which can be at a large-scale level depending on the environment, resulting in a large unified tensor. To reduce the computational cost of the 4D convolutions, we prune the unified sparse tensor by eliminating sparse voxels that exclusively contain the map timestamp. We keep only the voxels that contain at least the scan timestamp as illustrated in Fig. 6.3. This decision stems from our specific interest in inferring stability confidence scores only for the scan points, and we argue that far-away map voxels without corresponding scan voxels do not influence the stability.

Finally, we segment the stable points $\mathcal{P}_s$ from the current scan $\mathcal{P}$ based on the predicted stability confidence scores assigned to $\mathcal{P}$, where we apply a fixed threshold $\epsilon$ for segmenting the stable points. Unlike prior work [77], our approach leverages this scan-map discrepancy effectively in segmenting stable points, contributing to the network's robust generalization performance.

### 6.1.4 Training Labels for Sparse Convolutions

In contrast to moving object labels, which are available, for example, in the SemanticKITTI [16] dataset, stable and unstable points are usually not explicitly labeled. Therefore, we generate the training labels for the sparse CNN in an unsupervised fashion based on prior work [77]. This avoids the manual labeling process as it is time- and resource-consuming. We consider a point stable if we have at least two observations of its environment. We first build point cloud maps of the observations denoted as $\{\mathcal{M}\}_0^k$ using a simultaneous localization and mapping system such as FAST-LIO [203], where $k$ is the number of observations of the environment, along with their associated occupancy grid OctoMaps [75]. We refer to the work by Hroob et al. [77] for further details.

Assuming the point cloud maps are roughly aligned, we fine-align them using the iterative closest point algorithm. The labeling procedure starts with selecting a reference map $\mathcal{M}_i$. For each point $\boldsymbol{p} \in \mathcal{M}_i$, we assign a spatio-temporal stability label based on the maximum spatial distance $d$ to the nearest point in all other maps while accounting for occlusions by querying the occupancy of the point location in the query map.

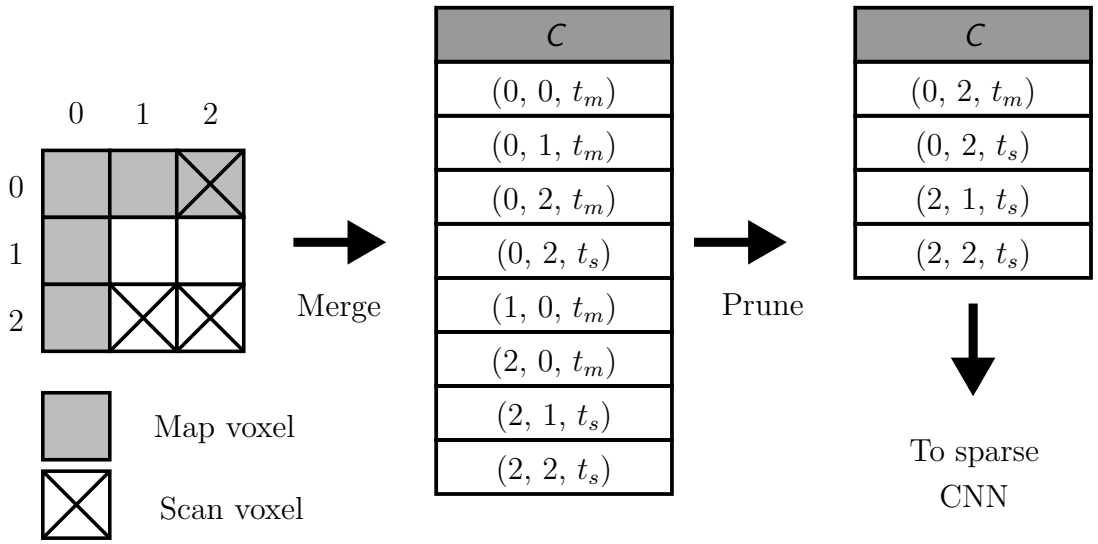



Figure 6.3: Possible occupancy combinations for scan and map voxels. Our method prunes map voxels without scan correspondence, substantially improving performance by eliminating unnecessary voxels. In this simple example, the map depicts a wall corner, and the illustrated scan voxels reveal the occlusion of some map voxels caused by an obstacle. We show the resulting voxel coordinates $C$ before and after pruning and indicate the map and scan timestamps as $t_m$ and $t_s$, respectively.

We transform this distance value into a unitless value using the cumulative distribution function of an exponential function: $F(d) = 1 - \exp(-d)$. This transformation bounds the continuous value between 0 and 1, effectively representing long-term spatio-temporal stability, where a value close to 0 suggests a stable point, whereas a value approaching 1 indicates an unstable point.

### 6.1.5 Implementation Details

We set the quantization size for the sparse voxel grid to 0.1 m so as not to lose details of the features. We train our 4D sparse CNN self-supervised by using the auto-generated stability labels as explained in Sec. 6.1.4 and use the root mean square error (RMSE) loss $\mathcal{L}$ to supervise the training on scan data only.

## 6.2 Experimental Evaluation

This work focuses on segmenting stable points from LiDAR scans and using them to improve the localization performance of scan-matching algorithms in changing environments. We present our experiments to show the capabilities of our method. The results of our experiments also support our key claims, which are: (i) increasing the robustness of robot long-term localization and (ii) generalizing well to different environments without model re-training.

### 6.2.1 Experimental Setup

We demonstrate our method's effectiveness in learning to segment stable points and improving long-term localization using the Bacchus Long-Term (BLT) dataset [143]. This dataset was collected in semi-structured agricultural environments over several months. Additionally, we assess our approach's generalization by employing two more datasets: Riseholme, which is a vineyard at the Riseholme campus and is also part of BLT and a parking lot from the North Campus Long-Term (NCLT) dataset [25], which has diverse objects not found in BLT, thus challenging our model's generalization capabilities.

We compare our approach to different baselines that all rely on the same localization method but differ in the preceding segmentation and filtering of the scan points. For all experiments, we use the normal distributions transform localization framework from Koide et al. [93], because it offers a flexible open-source implementation of 3D LiDAR-based localization in a given map. We use the following baselines: (i) *Raw*, which uses the unfiltered scans for localization, (ii) *Mask*, which takes the scan points that fall into the masked submap voxels, and (iii) LTS-NET [77], which filters stable points based on long-term stability labels but does not leverage the scan-map discrepancy. To assess the influence of moving objects on the localization performance, we also provide a baseline (iv) for which we filter out moving objects using 4DMOS from Chap. 3.

We train the presented approach and LTS-NET on the labels automatically generated from the BLT dataset. We do not train 4DMOS on this dataset for two reasons. First, 4DMOS generalizes well in new unseen environments, and second, the auto-generated labels from the BLT dataset will not work with 4DMOS since the labels do not indicate if the object is currently moving. To generate the training data, we utilize the BLT dataset with sequences from April 20th and June 1st for training and sequences from June 8th for validation. The model undergoes training for 60 epochs, and we save the best-performing model based on its performance on the validation dataset. Additionally, we augment the training batches by applying random flipping, rotating, and scaling transforms from Chap. 3. After predicting long-term stability confidence scores with our method, we use a fixed threshold of $\epsilon = 0.84$ for filtering stable points, which we chose based on the localization performance on the training and validation set.

### 6.2.2 Localization Performance in Agricultural Environments

In this section, we conduct experiments to support our first claim and evaluate the ability of our method to localize in changing environments using the segmented stable points only.

| Method | Raw | | Mask | | 4DMOS [121] | | LTS-NET [77] | | Ours | |
|---|---|---|---|---|---|---|---|---|---|---|
| Seq/Metric | $RMSE$ | $R_{ts}$ | $RMSE$ | $R_{ts}$ | $RMSE$ | $R_{ts}$ | $RMSE$ | $R_{ts}$ | $RMSE$ | $R_{ts}$ |
| April-20th* | 0.133±0 | **100±0** | 0.385±0.03 | 64.8±12 | 0.132±0 | **100±0** | **0.093±0** | **100±0** | 0.128±0 | **100±0** |
| June-1st* | 0.352±0 | 72±0 | 0.777±0.10 | 8.8±2 | 0.546±0 | **100±0** | 0.3±0 | **100±0** | **0.288±0** | **100±0** |
| June-8th** | 0.382±0 | **100±0** | 0.697±0.08 | 8.2±4 | 0.366±0 | **100±0** | **0.272±0** | **100±0** | 0.281±0 | **100±0** |
| June-29th | 0.483±0 | 8.8±0 | 0.610±0.01 | 8.2±1 | 0.451±0 | 8.8±0 | **0.469±0** | **100±0** | 0.528±0.02 | 90.7±0 |
| July-13th | 0.281±0.03 | 75±0 | 0.663±0.02 | 12.2±0.04 | **0.201±0** | 73.7±0 | 0.416±0 | 11.1±0 | 0.350±0.06 | **87±0** |

Table 6.1: Averaged localization performance comparison between baseline methods and our proposed approach (Ours) from five experiments, including standard deviations. We report RMSE results in meters and $R_{ts}$ values in percentage. The * indicates the training sequences of the segmentation of stable points, and the ** indicates the validation sequence.

To evaluate the performance of improving the accuracy of scan-to-map localization in changing environments through stable scan points, we first build a reference map using earlier sequences of the BLT dataset. We mainly use the April 6th sequence of an early growth stage of crops, which contains only stable objects. Subsequently, we utilize the data from the later sessions to perform localization within the reference map.

For the quantitative evaluation, we use the RMSE of the absolute trajectory error (ATE) [65]. However, it is essential to note that the localization algorithm might fail to estimate a reliable pose and provide inaccurate poses. To account for this, we consider the localization failed if the ATE exceeds a specific threshold $\tau$, which is 1.5 m in our use case. We excluded estimated poses beyond this point from the evaluation. We employ the trajectory duration ratio metric [36] to conduct a fair trajectory evaluation. It represents the ratio between the duration of the estimated trajectory $\Delta t_{\text{est}}$ and the total duration of the ground truth trajectory $\Delta t_{\text{gt}}$. Specifically:

$$R_{ts} = \frac{\Delta t_{\text{est}}}{\Delta t_{\text{gt}}}, \tag{6.4}$$

where a value closer to 100 % indicates a more reliable estimation. We summarize the localization performance of all methods in Tab. 6.1.

In the initial vineyard stages on April 20th, when the environment was relatively stable, most methods show a similar localization performance, with a slight advantage for LTS-NET. However, the mask baseline fails when the robot rotates at the end of a row. Subsequent sessions show degradation and eventual failure in the localization performance of both raw scans and 4DMOS-segmented scans, particularly in the June 29th sequence. This is due to 4DMOS' limitation in segmenting only dynamic objects, such as pedestrians, while neglecting unstable objects like overgrown vegetation. As illustrated in Fig. 6.1, these unstable points will not have a map correspondence, thus causing scan matching failure. Conversely, our method and LTS-NET consistently deliver competitive performance across various sequences and metrics.

An interesting observation is the results for the July 13th session, where LTS-NET initially tracks only about 11% of the trajectory, while our method exhibits a more robust performance, tracking 87% of the entire trajectory. In addition to RMSE and $R_{ts}$, we visualize the empirical cumulative distribution function [145] to evaluate the robustness of the system and to assess the registration accuracy between the reference map and the LiDAR scan. We show the cumulative distribution function plots for the test sequences in Fig. 6.4. The closer the curve is towards the upper left corner, the smaller the expected errors and the more robust the system. The results verify that the proposed approach is more robust than the baselines.

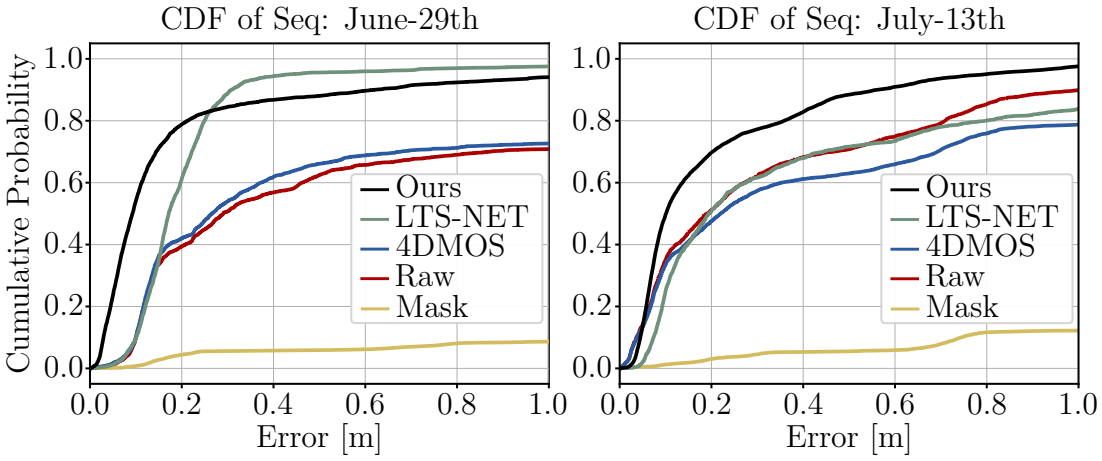



Figure 6.4: Plots of the cumulative distribution function of the translational localization error for the BLT sequences June-29th and July-13th using an off-the-shelf localization method with different point filters.

| | Method | | | |
|---|---|---|---|---|
| Dataset | Raw | 4DMOS | LTS-NET | Ours |
| Riseholme | 0.264 | 0.263 | 0.290 | **0.261** |
| NCLT-115 | 0.165 | 0.166 | 0.165 | **0.157** |
| NCLT-202 | 0.163 | 0.160 | 0.167 | **0.157** |
| NCLT-219 | 0.170 | 0.164 | 0.158 | **0.156** |

Table 6.2: Generalization performance of the proposed method compared to the baselines in new environments. We report the RMSE of the estimated trajectory in meters.

The mask baseline consistently fails in all sessions. The reason behind this failure is that this baseline relies on the accuracy of the initial pose prediction from the unscented Kalman filter to segment the scan points associated with map voxels; thus, a misalignment between the scan and the map larger than the size of the voxelization can fail to correctly segment the scan points, thus causing a failure in the localization.

### 6.2.3 Generalization Capabilities

Next, we assess our second claim about the proposed method's generalization capability to segment the stable points and enhance long-term localization in new and diverse environments. We conducted experiments in two environments. The first environment is a vineyard located at the Riseholme campus of the University of Lincoln, while the second setup is a parking lot from the NCLT dataset. We do not re-train the models or use transfer learning techniques in both cases. Further, we employ a reference map representing the static structure. In the vineyard, we observe changes due to plant growth, while the parking lot poses two distinct

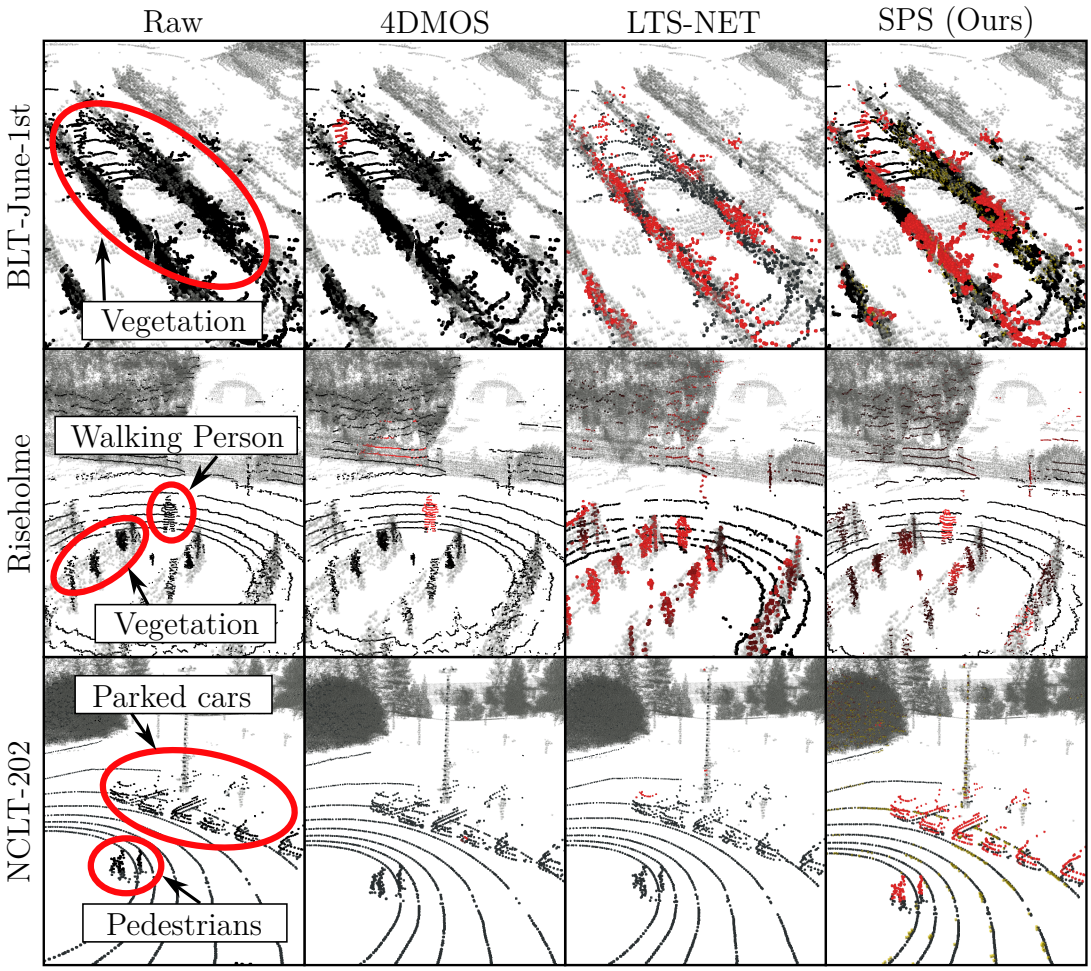



Figure 6.5: Comparison of the generalization ability of our method to multiple baselines. Riseholme is recorded in an agricultural environment with walking persons and changing vegetation, whereas NCLT is a parking lot with parked cars.

challenges: alterations in the parking lot shape based on the number of cars and the presence of plant vegetation, as well as moving objects in the sequence. For the NCLT dataset, we use data from the sequences 2012-01-15, 2012-02-02, and 2012-02-19, denoted as NCLT-115, NCLT-202, and NCLT-219, respectively.

Tab. 6.2 summarizes the method's localization performance compared to the baseline. The reported results are the averages of five runs. The deviation of the runs is not presented since the results were consistent. Additionally, we do not report the $R_{ts}$ metric and only report the RMSE of the ATE since the localizer effectively tracks the robot throughout the entire trajectory for all methods. We exclude the mask baseline from these experiments as it consistently fails in all trials due to initial pose misalignment. The results in Tab. 6.2 support our second claim. Notably, the localization performance of raw and segmented scans from 4DMOS exhibit similarities, suggesting that dynamic objects such as moving pedestrians or cars have minimal impact on the localization performance. To back

| Sequence | Method | IoU | Precision | Recall | F1 |
|---|---|---|---|---|---|
| June-8th** | 4DMOS | 0.039 | 0.554 | 0.039 | 0.072 |
| | LTS-NET | 0.643 | 0.836 | 0.738 | 0.779 |
| | Ours | **0.727** | **0.861** | **0.827** | **0.839** |
| June-29th | 4DMOS | 0.065 | 0.47 | 0.068 | 0.105 |
| | LTS-NET | 0.637 | 0.878 | 0.701 | 0.775 |
| | Ours | **0.784** | **0.924** | **0.836** | **0.877** |
| July-13th | 4DMOS | 0.006 | 0.541 | 0.006 | 0.012 |
| | LTS-NET | 0.611 | 0.846 | 0.687 | 0.755 |
| | Ours | **0.78** | **0.875** | **0.881** | **0.875** |

Table 6.3: Performance of segmenting stable points for the validation and test sequences of the BLT dataset. We report the average over all scans in the corresponding sequence. We compute all metrics for the unstable points. The best results are in bold. The ** indicates data used to validate the segmentation of unstable points.

this up, we manually labeled the dynamic objects and found their proportion to be 0.73 % of all points in the three sequences, and 4.22 % of the points belonging to movable objects like pedestrians and cars. This indicates that the majority of movable and, therefore, unstable points are not dynamic. We hypothesize that the utilization of stable points substantially influences the localization performance.

Furthermore, utilizing both scan and submap voxels as an indication of discrepancy allows us to segment and utilize stable points in new and diverse environments, as shown in Fig. 6.5. The proposed method successfully segments stable elements in the Riseholme dataset, including humans and vegetation. In contrast, the NCLT dataset accurately identifies parking cars and pedestrians as unstable objects, a task where both 4DMOS and LTS-NET fail. Possible reasons are that 4DMOS segments stable points based on their current motion and LTS-NET based on object shapes seen during training, an agricultural environment in this experiment.

### 6.2.4 Filtering of Unstable Points

The previous experiments suggest that our approach successfully localizes in changing environments by segmenting and filtering unstable points. To provide more detailed reasoning as to why our system improves upon existing methods, we provide a quantitative evaluation of the classification performance of unstable points. We report standard metrics such as IoU, precision, recall, and F1 score [49] for unstable points on the validation and test set of BLT and NCLT. We focus on evaluating unstable points since their removal is more critical than, for example, keeping all static points. It is important to note that we use the

| Sequence | Method | IoU | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 115 | 4DMOS | 0.113 | 0.359 | 0.139 | 0.174 |
| | LTS-NET | 0.054 | 0.269 | 0.07 | 0.099 |
| | Ours | **0.262** | **0.382** | **0.483** | **0.391** |
| 202 | 4DMOS | 0.198 | 0.649 | 0.23 | 0.302 |
| | LTS-NET | 0.152 | 0.601 | 0.167 | 0.251 |
| | Ours | **0.585** | **0.684** | **0.785** | **0.721** |
| 219 | 4DMOS | 0.115 | 0.446 | 0.129 | 0.174 |
| | LTS-NET | 0.075 | 0.277 | 0.096 | 0.132 |
| | Ours | **0.502** | **0.616** | **0.717** | **0.638** |

Table 6.4: Performance of segmenting stable points for the NCLT dataset. We average the reported results over all scans and compute all metrics for the unstable points. The best results are in bold.

| Dataset | LiDAR | 4DMOS | LTS-NET | Ours |
|---|---|---|---|---|
| BLT | 16-beams | 0.052 s (19.1 Hz) | 0.095 s (10.5 Hz) | **0.037 s (27.3 Hz)** |
| NCLT | 32-beams | 0.048 s (20.7 Hz) | 0.101 s (9.9 Hz) | **0.036 s (27.8 Hz)** |

Table 6.5: Average inference time for localizing a 3D LiDAR frame using different segmentation approaches.

auto-generated labels from Sec. 6.1.4 for evaluation, as no ground truth labels are available. The results in Tab. 6.3 and Tab. 6.4 illustrate the system's effectiveness in segmenting unstable points compared to the baselines. Our approach achieves the highest recall and removes more unstable points than the baselines, resulting in better localization in Sec. 6.2.2 and Sec. 6.2.3. Additionally, the results in Tab. 6.4 again confirm the ability of our proposed method to segment unstable points across novel and unseen environments. Note that the performance gap for 4DMOS is due to the fact that 4DMOS segments moving objects only, which is only a subset of the moving points.

### 6.2.5 Runtime

We summarize the inference time of our method compared to 4DMOS and LTS-NET in Tab. 6.5 on an NVIDIA GeForce GTX 1080 Ti GPU. The results demonstrate that the proposed approach can run sufficiently fast for mobile robots. Furthermore, our approach shows a smaller GPU memory demand of 1047 MB compared to 1703 MB for 4DMOS, and 9487 MB for LTS-NET.

## 6.3 Related Work

We can distinguish between local and global localization when dealing with localization based on a given map. The latter aims to determine the robot's pose in a map without prior pose information. In local localization, also called pose tracking, the robot starts from a known pose and updates the pose as time progresses. This work addresses local localization using 3D LiDAR data in a changing environment, commonly known as long-term localization.

**LiDAR Map-Based Localization.** Probabilistic and feature-based methods are popular approaches for robot LiDAR localization in a pre-built map. Examples of probabilistic methods include Kalman filters [178] and Monte Carlo localization [40], which are widely used for robot localization [4, 31, 117]. For instance, Chen et al. [31] exploit Monte Carlo localization to estimate the vehicle's local and global pose within a pre-built mesh-based map representation utilizing range images derived from 3D LiDAR data. On the other hand, feature-based methods such as scan matching techniques [18, 20] estimate the robot pose by aligning the current sensor readings (raw laser scans or visual features) with a pre-built map [93, 129, 151]. Some techniques even combine grids and features [200]. In contrast to the probabilistic methods, scan-matching methods need a good guess of the robot's initial pose. However, they often estimate a smoother trajectory. Therefore, we focus on scan-matching systems to improve their performance in dynamic environments.

**Localization in Dynamic Environments.** Regarding long-term localization, the environment may gradually or suddenly change over time, impacting the robot's pose estimation accuracy for scan-matching algorithms [39]. Therefore, researchers explore incorporating new information into the map. If the current observations do not align with the static map, failing pose tracking, a temporary map is generated. This temporary map is later fused with the static map for subsequent localization runs. For instance, Biber and Duckett [19] introduce a dynamic map that represents short and long-term changes. They address the stability-plasticity dilemma, which describes the trade-off in life-long learning of adapting to new observations while remembering old patterns. The proposed map consists of multiple sub-maps, updated based on different timescales like hours, days, or weeks. This allows for tracking the state of the environment at different timescales and then using the timescale for localization that fits the sensor's observations best.

Instead of modeling the temporal changes, Stachniss and Burgard [171] aim at identifying different configurations of the dynamic environment represented by multiple sub-maps. The robot is then localized by estimating the current configuration from previously clustered grid maps. Walcott et al. [190] propose a

dynamic pose graph for SLAM that incorporates low-dynamic changes by removing inactive scans and adding new scans. To deal with a more extensive variety of dynamics, Tipaldi et al. [180] develop a hidden Markov model to represent the dynamics of the environment and estimate both the robot's pose and the environment's state.

However, all these approaches operate on 2D laser data in mostly indoor environments. Using them with large-scale and outdoor 3D point clouds is not straightforward. For 3D data, cleaning the map in a post-processing step is common by removing points from dynamic objects. For example, Removert [90] removes map points based on their visibility in a query scan. We already discussed the generation of static 3D maps and refer the reader to Sec. 4.3 for further details on related work.

In contrast to the abovementioned methods, our approach does not require a complex map update process since it uses the initial environment map. We do not remove map points measured from moving objects but instead, filter out points that are generally inconsistent with the map. This also covers measurements from vegetation that changed between runs, for example.

**Deep Learning-based Localization.** Several methods exploit deep neural networks and semantic information for long-term localization. For example, Tinchev et al. [179] propose a learning-based method for segment-matching trees and localizing in diverse environments. The input point cloud is first segmented into individual objects based on the point distance, then a CNN computes a descriptor for each segment. This descriptor is matched against the existing segments in the map to estimate the final pose with PROSAC [33], a robust matching method based on random sample consensus [54].

At the same time, Kim et al. [87] present a long-term localization method based on a point cloud descriptor called Scan Context Image based on the Scan Context [89] descriptor developed for loop closure detection. The authors train a convolutional neural network to localize on a grid map by classifying the corresponding grid cell.

Zimmerman et al. [218] aim to overcome the discrepancy between the sensory data and the static map by leveraging human-readable cues. The cues stem from spotted text, such as room numbers, and help to guide the localization filter to areas where the text is expected to be seen. This increases robustness to low dynamic changes.

Recently, Wiesmann et al. [197] propose to learn a neural distance field from sensor data and demonstrate how to localize in such an implicit representation of the environment. The authors do not handle dynamic changes explicitly but rely on a robust kernel to weigh down the influence of moved objects in the environment during the scan registration.

While those methods avoid map updates and work on the sensory data, they are bound to indoor environments, rely on handcrafted features, or need supervised training. In contrast, our method learns the features in a self-supervised fashion. We develop a novel self-supervised segmentation of stable points method by exploiting 4D sparse convolutions. Our method can improve long-term localization performance by using stable points for localization. Using sparse representations allows us to achieve a segmentation in real-time.

## 6.4 Conclusion

In this chapter, we presented a novel approach to increase the accuracy of scan-to-map-based localization in dynamic environments and demonstrated how we can further use the knowledge about dynamics after answering the "What is moving?" question. Our approach segments the scan points into stable and unstable points based on their long-term stability, and we demonstrated how to use only the stable points for localization. The backbone of our method is a modified MinkUNet32 [32], a sparse 4D sparse CNN, that we trained in a self-supervised fashion.

In our experimental evaluation, we initially trained and evaluated our method using the BLT dataset and then assessed its generalization capabilities on two additional datasets. The outcome indicated improved localization performance, successful generalization to unseen data, and a runtime suitable for mobile robots, which supported all claims made in this chapter.

Despite the effectiveness of our proposed approach, we rely on an accurate relative pose estimate for the initial alignment of the scan with the map to accurately determine discrepancies between scan and map data. This relative alignment should be a reasonable initial guess to avoid wrong segmentation leading to a localization failure.

# Part II

# Spatio-Temporal Prediction

# Chapter 7

# Supervised Trajectory Prediction

The ability to identify moving objects in dynamic environments as presented in the previous chapters and predict their future movement is a subconscious and effortless skill for humans and the key to safe autonomous navigation. Humans anticipate possible maneuvers of other agents and react accordingly in advance. Especially with varying numbers and types of moving objects in dynamic and complex environments, predicting the intentions of others can become a challenging task. Humans refined this skill over time, typically outperforming technical systems. Interdependencies between agent behaviors and the multimodal nature of future intentions in a dynamic and complex environment render trajectory prediction a challenging problem. Intelligent systems require a reliable perception of the driving environment over time and space to make medium- or even long-term predictions. They have to answer this thesis's second main research question about an observed object, which is "Where is an object moving to?".

Since predicting future behavior is not straightforward, we will first focus on trajectory prediction for autonomous vehicles. In contrast to pedestrians and bicycles, these vehicles are more restricted in motion due to larger inertia, traffic rules, and road geometries. Inertia makes vehicle movements better predictable, especially in structured driving scenarios like highway driving.

Despite these favorable properties, challenges arise due to the dynamic interactions among vehicles. Particularly, lane change maneuvers require special attention for surrounding objects. Given a target vehicle for which one wants to predict the future trajectory, neighboring vehicles influence but also restrict the possible maneuvers, such as accelerating or changing a lane. The situation depicted in Fig. 7.1 illustrates such a scenario. If the black vehicle (T) is faster than the truck in front (F), it must slow down or change lanes. We show possible scenarios as colorized trajectories. Their likelihood strongly depends on the positions, velocities, and accelerations of the neighboring vehicles F, L, R, RL,

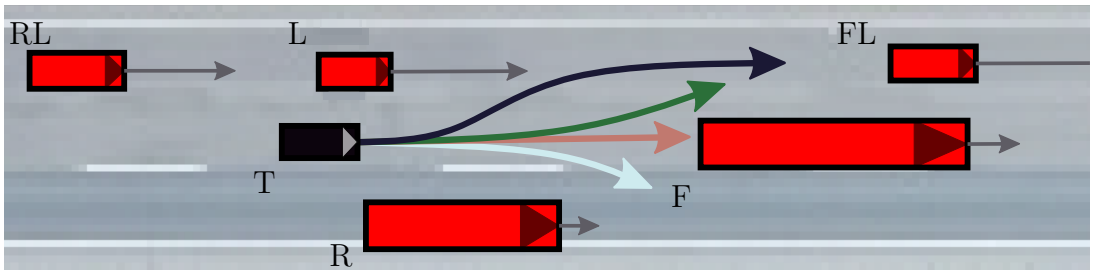



Figure 7.1: The goal is to predict the future maneuver and trajectory of the black target vehicle (T). We estimate the different future motions depending on the neighbors' states, such as position, velocity, and acceleration. Colors indicate possible maneuvers with corresponding trajectories. Gray arrows denote velocity vectors, and their lengths indicate speed.

and FL. The example illustrates the inherent interdependencies among vehicle behaviors for a time horizon of several seconds. This chapter proposes a method to tackle these challenges by paying attention to the other road users and their dynamics and explicitly estimating the driver's maneuver intention in advance.

In this chapter, we propose a new, data-driven approach for predicting the motion of vehicles in a road environment. The model allows for inferring future intentions from the past interaction among vehicles in highway driving scenarios. Using our neighborhood-based data representation, the proposed system jointly exploits correlations in the spatial and temporal domain using a CNN. Our system considers multiple possible maneuver intentions and their corresponding motion and predicts the trajectory for five seconds into the future. We implemented our approach and evaluated it on two highway datasets taken in different countries and can achieve a competitive prediction performance.

This chapter is a joint work with Thomas Höllen, and the initial idea is based on his master thesis. My contribution was refining, developing, and implementing the approach and conducting the experimental evaluation for the published paper [122].

## 7.1 Maneuver-Based Trajectory Prediction Using 2D Convolutions

The main contributions of this chapter towards vehicle motion prediction are twofold. First, we present a novel semantic neighborhood representation of the scene around a target vehicle for joint aggregation of higher-level features in prediction tasks. The memory-efficient and dense 3D tensor encodes the time, neighbor positions, and past vehicle states as dynamic context. Second, we propose using two 2D CNNs for joint spatio-temporal feature extraction from the proposed input representation. Our approach explicitly uses convolutions across time and the

space of neighboring vehicles. To this end, we classify a target vehicle's future lane change intention with respect to a lateral motion and then predict a trajectory based on the classified lane change intention. Using a spatio-temporal CNN for sequence prediction leads to a simpler and more compact architecture than recurrent network approaches, resulting in fewer parameters to train but competitive prediction performance.

This yields an approach that can (i) successfully classify lane change maneuvers and predict a corresponding trajectory for real-world scenarios by (ii) performing joint spatio-temporal feature aggregation with 2D CNNs, (iii) and outperform state-of-the-art methods. The chapter, our experimental evaluation, and an ablation study support these three main claims.

### 7.1.1 Overview

The key idea of our approach is to first classify the lane change maneuver of a target vehicle for each step into the future and then predict the corresponding trajectory with the classification result as an additional input.

At the current timestamp $t=0$, the goal is to predict the sequence of $P$ future positions $\{(x_1, y_1), \ldots, (x_P, y_P)\}$ from $H$ past states $\mathcal{S} = \{s_0, \ldots, s_{H-1}\}$. Each past state $s_t$ consists of multiple channels like position, acceleration, and velocity of the target vehicle and its neighbors, which we transform into a 3D tensor as described in more detail in Sec. 7.1.2. We pre-define the lane change maneuvers, which cover sequences of straight driving, left, and right lane changes, see Sec. 7.1.6. The classification and regression parts rely on exploiting past state sequences with spatio-temporal 2D convolutions, which we explain in Sec. 7.1.3. Based on the CNN features, the classification module predicts a discrete maneuver (straight driving, left lane change, or right lane change) for each prediction step $1, \ldots, P$ into the future, see Sec. 7.1.4. We pass the classified sequence to the regression module as an additional input to predict a trajectory based on the maneuver as outlined in Sec. 7.1.5. Both modules share the same convolution-based architecture. When training on common, publicly available highway datasets, lane changes are usually underrepresented since the vehicles drive straight most of the time. We found that training a single network with two heads and a joint multi-task loss for classification and regression is hard to tune due to the imbalance of lane change labels. Therefore, we propose to use partitioned models for each task, which makes it easier to train them separately without shared parameters.

### 7.1.2 Neighborhood-Based Input Representation

We consider the target vehicle's direct neighborhood to predict a trajectory from past data. Close neighbors influence the possible actions a vehicle can take. For example, a leading vehicle with a lower velocity requires deceleration or a lane change of the target vehicle. Still, the corresponding neighboring lane must be free for a safe lane change maneuver. Whereas the short-time prediction of, for example, the next state is mainly restricted by the inertia of the vehicle, we assume that longer prediction horizons, including the neighborhood, lead to better predictions, which we also back up by our experiment in Sec. 7.2.4. We define a semantic neighborhood consisting of up to seven vehicles around the target inspired by the work of Hu et al. [78]. As mentioned, the vehicles in front (F) and on the left (L) and right (R) play an important role. We further discretize the remaining neighborhood in front left (FL) and right (FR) as well as rear left (RL) and right (RR) vehicles. This results in a maximum number of eight considered vehicles, including the target. We illustrate the semantic neighborhood in Fig. 7.2.

For each vehicle $i$, we represent its state $\boldsymbol{s}_t^i$ at time $t$ with the $x_t^i$ and $y_t^i$ position, velocity $v_t^i$, and acceleration $a_t^i$,

$$\boldsymbol{s}_t^i = \left[x_t^i, y_t^i, v_t^i, a_t^i\right]^\mathsf{T}. \tag{7.1}$$

Note that the publicly available highway datasets provide these quantities for this work. Otherwise, a previous detection and tracking module must estimate the positional, velocity, and acceleration. We normalize the data beforehand to guarantee a consistent input scaling. Note that our approach is not restricted to these channels, and our multi-channel design makes it easy to add new information channels useful for prediction.

Next, we concatenate these states for all neighboring vehicles, resulting in a state $s_t$ at time $t$. We store the neighbors and channels per time $t$ along the third, temporal dimension, see Fig. 7.2. This results in an image-like 3D tensor with the spatial neighbor information as the height and the time as the width of an image with the given input channels. We use three seconds of input data to compare with existing methods and predict the maneuvers and positions for the following five seconds. This results in an input tensor of dimension $4 \times 8 \times 30$ with four channels, eight neighbors, and 30 frames of history sampled at 10 Hz.

If a neighbor is not present, we set the entries to zero, which is a state that is not naturally present in the input data due to the choice of the reference location. The concatenation order of neighbors can be chosen randomly but needs to be consistent for training and evaluation. We experienced that keeping neighbor positions close to each other in the data representation is advantageous. Based on our experiments in Sec. 7.2.4, we hypothesize that the convolutional network

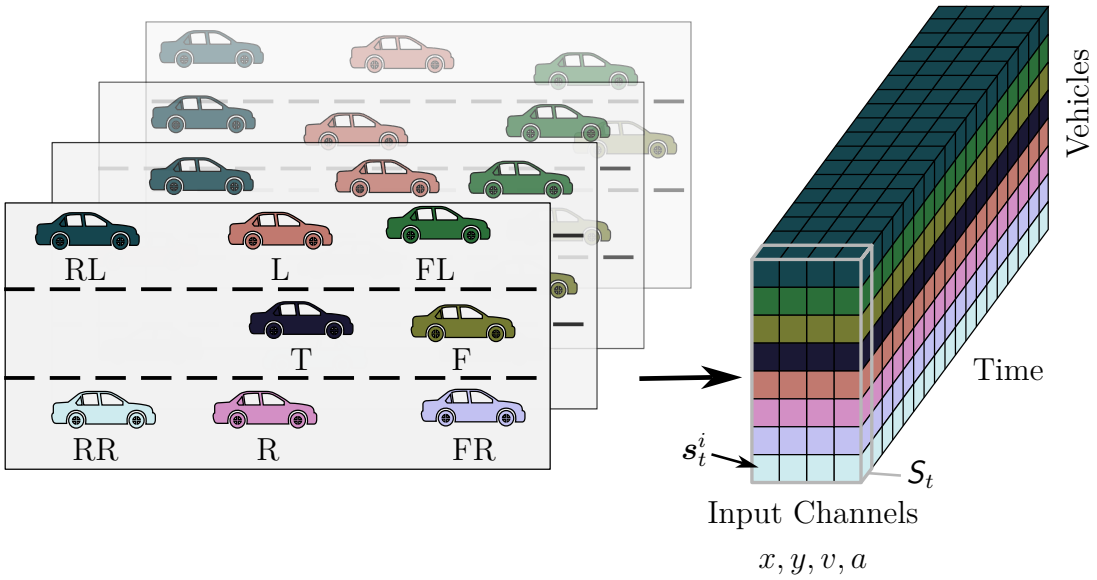



Figure 7.2: *Left*: At each timestamp, we discretize the neighborhood around the target vehicle into seven neighbor positions. *Right*: For each neighbor, we extract the input channels $x$ and $y$ positions, velocity $v$, and acceleration $a$ for each timestamp $t$ resulting in 2D tensors $S_t$ and concatenate them to a 3D input tensor.

can learn a notion of absolute position and discriminates between neighbors. We discuss this further in Sec. 7.1.3. In contrast to the previous chapters and other approaches in Chap. 3 to Chap. 6 that encode the whole scene into a grid structure [43], our neighborhood representation is more dense and memory-efficient, since we do not take all vehicles into account.

### 7.1.3 Spatio-Temporal Convolutional Neural Network

In the previous works presented in Part I, we used a modified off-the-shelf 4D sparse CNN for the segmentation task. In this chapter, existing convolutional architectures are unavailable, and we therefore design our own. With our efficient dense representation of the data proposed in Sec. 7.1.2, we can process the data with 2D convolutions to predict a future maneuver and corresponding trajectory.

As done in the previous chapters, we apply convolutions along the time dimension to detect local correlations between states in the history sequence. In addition to that, we extend the temporal convolution along the spatial neighbor dimension, resulting in 2D convolutions. We use four convolutional layers for feature aggregation. The hierarchical stacking of multiple layers ensures that the low-level features from the input data can be combined in the subsequent layers for higher-level representations. We conclude that higher layers learn to interpret the convolution along the neighborhood dimension and, therefore, account for the absolute position of the neighbor. This assumption is backed up by Kayhan et

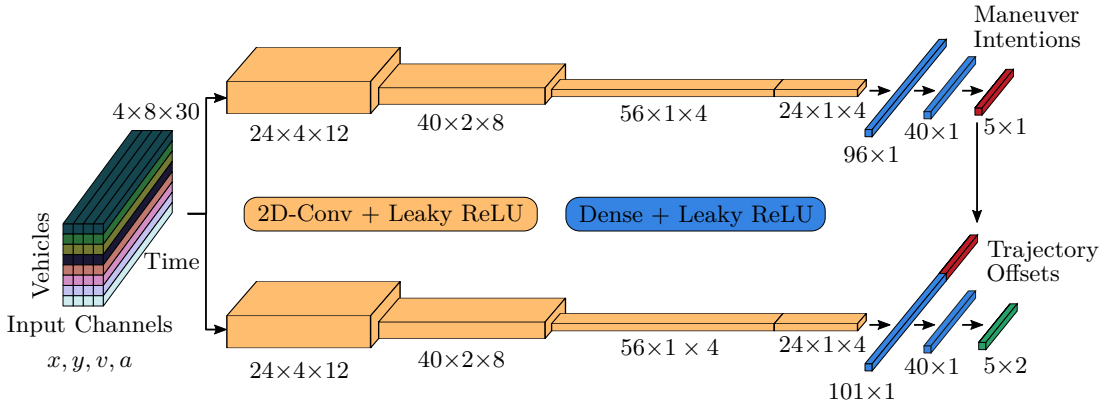



Figure 7.3: Two spatio-temporal convolutional networks aggregate features parallelly from the neighborhood-based 3D input tensor. The tensor from Fig. 7.2 contains the $x$ and $y$ positions, velocity $v$, and acceleration $a$ for all eight vehicles at the past timestamps. The upper module classifies a maneuver for each prediction step and passes the result to the lower regression module to predict the offsets between the future trajectory and the current position. For each layer, we provide the output dimensions.

al. [85], who claim that CNNs can encode absolute spatial locations from boundary effects. Since we use zero padding and our spatial dimension is only eight, we can expect to use the boundary effects during training.

It is worth pointing out that the learning process of all vehicles' temporal and spatial relations happens jointly. This makes the prediction fast since, in contrast to recurrent models commonly used for temporal prediction, we do not need to compute intermediate hidden states. We briefly review recurrent models in Sec. 7.3 for more details. Analogously to vision-based 2D convolutions with multiple image channels, the filter depth matches the number of feature channels.

Stacking multiple convolutional layers increases the receptive field of the CNN. For temporal convolutional networks, the receptive field determines the number of past timestamps that influence a single output. This is an important factor for prediction performance, especially for longer input sequences. Besides deepening the architecture by adding layers, increasing the size of the convolution kernels is possible. However, since all techniques increase the number of training parameters and the model complexity, we use dilated convolutions. A dilated convolution expands the receptive field without reducing the resolution or coverage, as pointed out by Yu and Koltun [209]. We implement dilations along the temporal dimension by adding zeros between kernel entries depending on the dilation rate. We provide an experiment on the improvement by using dilated convolutions in Sec. 7.2.4.

We illustrate the complete network architecture in Fig. 7.3. Each convolutional layer uses a leaky rectified linear unit activation function [115]. At the first layer, we convolve the input tensor of size $4 \times 8 \times 30$ with 24 filters with a kernel size of $5 \times 10$ and dilation of one. Each output in the resulting feature map of

size $24 \times 4 \times 12$ has a receptive field of $4 \times 5 \times 19$. Furthermore, we carry out a dilated convolution with 40 filters of size $3 \times 3$ resulting in a size of $40 \times 2 \times 8$. The third layer applies 56 filters of size $2 \times 3$ with dilation, which leads to a map with $56 \times 1 \times 4$ features. Finally, the last convolutional layer reduces the channel dimension with 24 $1 \times 1$ kernels to 24 feature channels and a final receptive field of $4 \times 8 \times 27$.

### 7.1.4 Maneuver Classification

When only using a regression module to predict the trajectory, the model must capture multiple possible maneuvers as motivated in Fig. 7.1. Mozaffari et al. [127] point out that this can lead to predictions that are averaged over all possible modes. We first classify the target vehicle's future maneuver and predict a trajectory that depends on the estimated maneuver.

To classify the future maneuver of the vehicle with our spatio-temporal CNN defined in Sec. 7.1.3, we flatten the output of the last convolutional layer with a size of $24 \cdot 4 = 96$ and pass it to a fully connected layer with a leaky rectified linear unit activation function and a hidden dimension of 40. A final fully connected layer processes the hidden feature vector of length 40. The output layer predicts the class logits, which we can normalize to a probability for each of the three maneuvers "straight", "left", and "right" at each prediction step, resulting in an output vector of size $5 \times 3$. Selecting the indices of the maneuvers with the highest predicted probability results in an estimated maneuver intention vector of size $5 \times 1$ containing values between 0 and 2. Based on the three possible maneuvers at each of the $P = 5$ prediction steps, there are in total $3^5 = 243$ maneuver sequences the classification network can model. During training, we optimize the network parameters by minimizing the sum of negative log-likelihoods at each prediction step reading

$$\mathcal{L}_{\text{class}} = \sum_{t=1}^{P} -\log p(c_t), \tag{7.2}$$

with $p(c_t)$ denoting the predicted probability for the ground truth class $c_t$ at timestamp $t$. We analyze the improvements in the prediction performance achieved by our classification module in Sec. 7.2.4.

### 7.1.5 Trajectory Regression

We feed the classified maneuver sequence directly to the regression module to make maneuver-based predictions. As discussed above, there are 243 possible combinations of maneuver predictions for a given input tensor. We concatenate the resulting vector of size $5 \times 1$ containing the indices of the five most likely maneuvers at each timestamp with the flattened CNN output vector, resulting

in 101 features. We pass this vector to a fully connected layer with the same architecture as in the classification module. The corresponding hidden vector of size 40 is the input for the final output layer with a linear activation function, which then predicts a vector of size $5 \times 2$ containing the $x$ and $y$ positions at each of the five prediction steps. During training, we use the ground truth maneuvers as input for the regression module, whereas during evaluation, we use the maneuvers predicted by the classification module. The training aims at minimizing the batch-wise RMSE reading

$$\mathcal{L}_{\text{reg}} = \sqrt{\frac{1}{P}\sum_{t=1}^{P}\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\|^2}, \tag{7.3}$$

with predicted locations $\hat{\boldsymbol{x}}_t$ and ground truth locations $\boldsymbol{x}_t$ for each timestamp $t$, where each location consists of the $x$ and $y$ positions.

With the proposed architecture, the temporal and spatial information of the highway scene around a target vehicle is jointly encoded by applying 2D convolutions. As Bai et al. [12] point out, the temporal convolution avoids a long backpropagation path, resulting in a more compact model, which is easier to train compared to RNNs. Also, we can carry out the prediction in parallel due to the temporal convolutional structure resulting in faster inference. The total amount of trainable parameters for the design depicted in Fig. 7.3 is 65,721. For comparison, the convolutional social pooling approach [43] has 194,954, and the multiple futures prediction approach [175] has 1,073,644 trainable parameters.

### 7.1.6 Output Parameterization and Ground Truth Generation

The number of predicted future positions $P$ depends on the output size of the last dense layer in the classification and regression heads in Fig. 7.3 and is only limited by the resolution of the dataset used for training. We choose to predict and evaluate the future for the next 5 s represented via 5 positions being 1 s apart in time. This sampling rate is commonly used for assessing the performance of trajectory prediction methods [43, 166, 215]. Note that we still process the input data at 10 Hz, unaffected by the output resolution. Furthermore, a higher output rate can be easily added.

A ground truth maneuver output with three classes (straight, left lane change, right lane change) is of size $5 \times 3$, and we compute it from the lane IDs. The lane ID is provided by the datasets and not part of the input data. If the lane ID of a vehicle changes between consecutive frames, we assume a lane change with a duration of 2 s before and after the corresponding frames and infer the maneuver direction from the lane IDs. The regression layer outputs trajectory offsets that

describe the difference between the predicted position and the initial position at $t = 0$ of the target vehicle. This improves the training since we can achieve the identity mapping from input to output by predicting zero offsets. We denormalize the model's output at inference time and add them to the target's current position.

### 7.1.7 Implementation Details

We train the classification and regression modules with the Adam optimizer [92] and a learning rate of $7 \cdot 10^{-5}$. Additionally, we augment the training set by overlapping the input data by 20 frames, resulting in more training examples. This leads to better coverage of different lane change maneuvers in the training data.

## 7.2 Experimental Evaluation

This chapter mainly focuses on predicting a vehicle's lane change maneuver and corresponding future trajectory based on past information about states, including the neighborhood. We present our experiments to show the capabilities of our method and to support our key claims made about our work, which are: (i) Successfully classifying lane change maneuvers and predicting a corresponding trajectory for real-world scenarios by (ii) performing joint spatio-temporal feature aggregation with 2D CNNs, (iii) and outperforming state-of-the-art methods.

### 7.2.1 Experimental Setup

We consider two datasets to train and evaluate our proposed approach in real-world scenarios. The highD dataset [94] contains 110,000 vehicle trajectories from 60 highway recordings. The highways are located in Germany and have 2 to 3 lanes. To show that our approach works well with different driving behaviors and highway structures, we also use the NGSIM dataset [69] with 9,206 vehicles driving on two US highways with 5 to 6 lanes. Both datasets provide access to position, velocity, and acceleration. For each dataset, we use 70% of all vehicle trajectories for training, 10% for validation, and 20% for testing. We use the same vehicle trajectories for testing as Song et al. [166] for a better comparison.

The total training time for each module of our approach is 17.5 h with 300 epochs on an Intel Xeon W-2133 CPU and an Nvidia Quadro RTX 5000 GPU. It takes around 0.3 ms on average to classify the future maneuver sequence and to predict the corresponding trajectory.

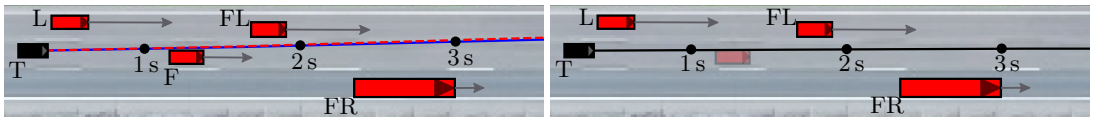



Figure 7.4: Two scenarios with predictions for the black target vehicle (T). *Left*: The original scene with a dashed red line as ground truth. The solid line represents the first seconds of the predicted future trajectory. The length of the gray velocity vectors depicts the velocity of neighboring vehicles. Our approach successfully classifies a left lane change (blue) since the vehicle in front (F) is slower. *Right*: If we modify the neighborhood by removing the vehicle in front, the prediction changes to a straight driving maneuver (black).

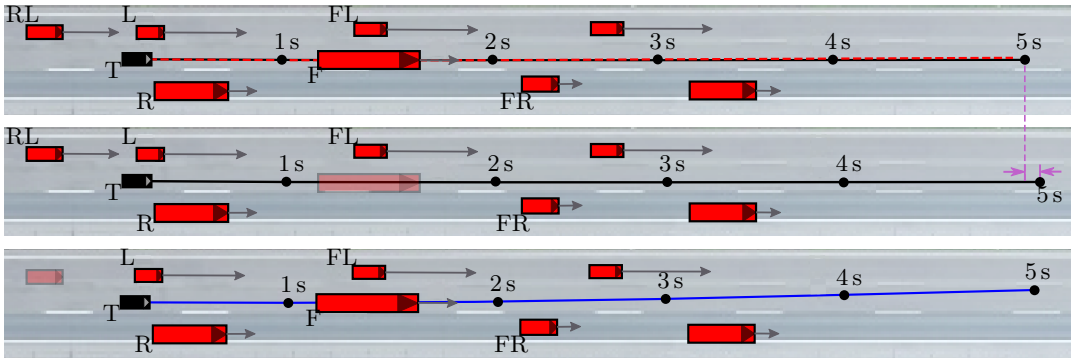



Figure 7.5: Three scenarios with modified environments. *Top*: In the original highD scene, the target cannot change the lane and needs to decelerate. *Middle*: Removed leading vehicle; the target does not need to decelerate, resulting in a longer trajectory. *Bottom*: Removed left rear neighbor; a left lane change (blue) is now possible and, therefore, predicted. Note that the length of the gray velocity vectors depicts the velocity of neighboring vehicles.

### 7.2.2 Qualitative Analysis

The qualitative experiments evaluate the prediction results on real-world data and support the claim that our modular approach of classification and regression results in reasonable predicted maneuver-based trajectories. We show that the prediction relies on the spatio-temporal encoding of the target vehicle's surroundings by modifying the neighborhood, which leads to different results. We take the original scenes from recording 56 of the highD dataset.

In Fig. 7.4, we consider two possible scenarios for the same highway scene and depict the original neighborhood on the left. A slower vehicle (F) is in front of the black target vehicle (T), forcing it to change lanes or decelerate. In this case, the target will let the left center (L) vehicle pass and change lanes. Our approach successfully classifies a left lane change maneuver indicated by the blue color and predicts a trajectory that matches the ground truth depicted in red. To demonstrate the influence of the semantic neighborhood on the prediction, we remove the leading vehicle (F) by replacing the corresponding entries in the input data with zeros. Based on the new neighborhood, our model predicts a straight driving maneuver since there is no longer a need to change lanes.

The second example in Fig. 7.5 demonstrates a more advanced highway scene. The black target vehicle (T) faces a slower vehicle in front (F) again. Additionally, two vehicles at the rear and center of the left lane (RL and L) hinder the target from changing lanes. Our approach predicts that the vehicle will stay on the lane and slow down. We show the ground truth in dashed red, which confirms our prediction. In the middle part of Fig. 7.5, we removed the truck in front (F) from the spatio-temporal input representation. One can see that the predicted trajectory is still straight but longer compared to the previous case. We can reason that there is no longer a slower leading vehicle that forces the target vehicle to decelerate. In a third scenario, we keep the truck in front (F) and remove the vehicle on the rear left (RL). The classification module now outputs a lane change maneuver to the left (blue), and the corresponding trajectory moves to the left lane. All three examples show that our approach can predict reasonable maneuver intentions and trajectories from the neighborhood of the target vehicle.

### 7.2.3 Quantitative Analysis

We present a second experiment to support the claim that our approach outperforms existing state-of-the-art methods for trajectory prediction. For a fair comparison, we evaluate our approach on the same test sets for each dataset as done by Song et al. [166]. We report the RMSE for each prediction step, a common trajectory prediction metric. We compare the results with the baseline methods S-LSTM [5], CS-LSTM [43], S-GAN [67], MATF [215] and PiP-noPlan [166]. In the case of S-GAN and MATF, which are stochastic models, we take the best root mean squared error among three sampled trajectories. We refer to the PiP-noPlan implementation since the planning coupled module in PiP uses the ground truth future trajectory of the target vehicle to predict the future trajectory of neighboring vehicles, which would result in an unfair comparison.

We show the final results in Tab. 7.1. Our method generally shows better results for both datasets at the prediction steps. Especially for the highD dataset, our proposed approach outperforms other state-of-the-art methods at larger prediction horizons. All methods show larger RMSE values for the NGSIM dataset, which we can explain by larger noise in the data as pointed out by Krajewski et al. [94]. This noise leads to smaller performance improvements between the baselines and explains why our method achieves a smaller margin than the evaluation on the highD dataset. We conclude that noise, for example, in the vehicle speed, heavily influences the prediction due to the high average speed on highways.

| Dataset | Time | S-LSTM [5] | CS-LSTM [43] | S-GAN [67] | MATF [215] | PiP-noPlan [166] | Ours |
|---|---|---|---|---|---|---|---|
| NGSIM | 1 s | 0.60 | 0.58 | 0.57 | 0.66 | 0.55 | **0.53** |
| | 2 s | 1.28 | 1.26 | 1.32 | 1.34 | 1.20 | **1.17** |
| | 3 s | 2.09 | 2.07 | 2.22 | 2.08 | 2.00 | **1.93** |
| | 4 s | 3.10 | 3.09 | 3.26 | 2.97 | 3.01 | **2.88** |
| | 5 s | 4.37 | 4.37 | 4.40 | 4.13 | 4.27 | **4.05** |
| highD | 1 s | 0.19 | 0.19 | 0.30 | - | 0.18 | **0.10** |
| | 2 s | 0.57 | 0.57 | 0.78 | - | 0.53 | **0.21** |
| | 3 s | 1.18 | 1.16 | 1.46 | - | 1.09 | **0.41** |
| | 4 s | 2.00 | 1.96 | 2.34 | - | 1.86 | **0.78** |
| | 5 s | 3.02 | 2.96 | 3.41 | - | 2.81 | **1.34** |

Table 7.1: Comparison of the root mean squared error at each prediction step evaluated on the NGSIM [69] and highD [94] datasets. Bold numbers indicate the best results. The baseline results are reported by Song et al. [166]. For the stochastic models S-GAN and MATF, we report the best root mean squared error among three sampled trajectories. Note that for PiP, we refer to the PiP-noPlan implementation for a fair comparison since the planning coupled module in PiP partially uses ground truth information.

| Time | Only Input $x$ and $y$ | Without Neighborhood | Shuffled Neighborhood | Without Dilation | Without Maneuver | Random Maneuver | Full Approach |
|---|---|---|---|---|---|---|---|
| 1 s | 0.24 | **0.08** | 0.15 | 0.15 | 0.14 | 0.14 | 0.10 |
| 2 s | 0.64 | **0.18** | 0.25 | 0.30 | 0.26 | 0.33 | 0.21 |
| 3 s | 1.24 | **0.39** | 0.43 | 0.52 | 0.47 | 0.62 | 0.41 |
| 4 s | 2.02 | 0.83 | 0.82 | 0.90 | 0.87 | 1.04 | **0.78** |
| 5 s | 3.00 | 1.52 | 1.40 | 1.45 | 1.48 | 1.60 | **1.34** |

Table 7.2: Ablation study with the root mean squared error at each prediction step evaluated using the highD dataset [94]. Bold numbers indicate the best results.

### 7.2.4 Ablation Study

Finally, we conduct an ablation study to evaluate how much each proposed component of our method contributes to the performance reported in Tab. 7.1. Since the data is more accurate, we re-train and evaluate our approach with several modifications on the highD [94] dataset. We show the results in Tab. 7.2.

In a first ablation experiment, we re-train our method only using $x$- and $y$-position as input channels. The result shown in Tab. 7.2 indicates that adding velocity and acceleration as inputs leads to a major performance improvement, which justifies the multi-channel design of the proposed 3D input tensor.

Suppose we ignore the local neighborhood defined in Sec. 7.1.2 for prediction as done by Nikhil and Morris [131]. In that case, the prediction performance is slightly better for short-term predictions but worse for larger prediction horizons. This indicates that the behavior of other traffic participants mainly influences the future trajectory at larger horizons, and we should not ignore it. Furthermore, we re-train our approach with a shuffled neighborhood order, and the competitive result for prediction steps larger than 1 s indicates that the CNN is able to learn the absolute position across the vehicle dimension independent of the ordering.

We conduct an additional ablation study to investigate the effect of dilated convolutions to increase the receptive field and reduce the number of parameters. If we do not use dilated convolutions, we need to increase the size of the dense layers in Fig. 7.3 to account for the resulting larger feature maps at the output stage of the CNN. This results in a model with 90,681 parameters. Our experiment shows that using dilated convolutions leads to better performance while having fewer parameters to train.

The last ablation study investigates the effect of our maneuver-based prediction. First, we do not use maneuvers, and the performance degrades. We hypothesize that the regression module alone cannot capture all maneuvers and that a prior classification improves distinguishing between these modes. Finally, we show that our classification module can predict different maneuvers based on the previous input by comparing it to a model randomly sampling a maneuver sequence from the training distribution. One can see that the sampled maneuvers lead to worse performance than our trained classification module.

## 7.3 Related Work

Brown et al. [21] break down the task of estimating and predicting human driver behavior into state estimation, intention estimation, trait estimation, and motion prediction. In this chapter, we focus on predicting the future motion of a vehicle based on the estimated driver intention and organize the related work according

to their level of interaction awareness, intention estimation, and finally, different strategies to process spatio-temporal data for prediction.

**Interaction-aware Prediction.** Inferring the future motion of a vehicle from past data has been approached from different perspectives. Closed-loop approaches roll out a control policy in a forward simulation for all target vehicles up to the prediction time, which results in interaction-aware trajectories. Schulz et al. [160] model the evolution of a traffic scene as a Markov process. Using a particle filter, they jointly estimate a hierarchical set of variables such as kinematic state, route intentions, maneuver intentions, and actions for all traffic agents.

Interaction awareness between different traffic participants is deepened with game-theoretic approaches [173], which condition an agent's future motion on the predicted motion of others. Peters et al. [140] infer the parameters of cost functions of other agents from partial state observations and use these objectives in a dynamic game theoretic framework for trajectory prediction. Since this dependency increases the computational complexity and can become intractable, other approaches model the future behaviors of vehicles to be independent of each other. Among these, physics-based models solely use kinematic and dynamic properties and apply filter- or sampling-based methods [99].

Other approaches aim to learn interactions from data. Mozaffari et al. [127] differentiate data-based prediction approaches based on input representation, output type, and prediction method. If we only use the target's trajectory history for prediction, we cannot consider interdependencies between surrounding vehicles [139, 202]. In contrast, adding information about neighboring agents [38, 44], stacking different sources of spatial information in bird's eye view [37, 73] or using raw sensor data of the target's surroundings makes interaction-awareness between past states possible [26, 101].

Deo and Trivedi [43] combine these ideas and develop a so-called "social tensor". This tensor consists of a bird's eye view image of the scene and is augmented by pre-processed temporal features. They use a combination of LSTMs for temporal feature extraction for each vehicle trajectory and CNNs for processing the resulting grid representation. An LSTM decoder generates the final trajectory for each maneuver.

This chapter extended this idea by defining a spatio-temporal representation in advance and then jointly aggregating spatial and temporal features for prediction instead of processing them separately [43].

**Intention Estimation.** A common goal of motion prediction is the estimation of maneuver intentions. These methods provide a high-level understanding of future behavior, usually defined for specific scenarios like intersections [219] or highway lane changes [47]. To predict low-level future motion, we can use an occupancy grid map containing the occupancy probabilities at future timestamps [73].

This output representation also allows for predicting multiple future modes but lacks accuracy for large grid cells, resulting in less consistent trajectories.

Another possibility is to predict trajectories directly, either in a uni- or multimodal fashion. Unimodal predictions come with a lower computational cost but tend to converge to a mean of different behavior modes [38, 114]. Maneuver-based predictions fix this problem by outputting multiple motion hypotheses incorporating different maneuvers. We can formulate these models in a probabilistic framework, meaning they model or sample from a multimodal distribution conditioned on the input data [67, 175, 215].

Other approaches estimate intention modes in advance and use them to predict a trajectory [166, 202] based on the maneuver. Song et al. [166] propose PiP to inform the prediction pipeline with the planned trajectory of the ego-vehicle. The paper suggests that conditioning the prediction module with planned trajectories improves the forecasting. However, the experimental evaluation is not trivial because a plan without ground truth information is usually unavailable.

Casas et al. [26] perform a multi-class classification with eight intention classes using CNNs on a voxelized LiDAR scan and a dynamic map containing road structures and traffic lights. The generated intention scores are then further processed to condition the trajectory estimation.

In contrast to these approaches, we define an intention space with three lateral motions and classify a maneuver at each step. Combining these motions at each step results in more maneuvers during the prediction.

**Spatio-Temporal Prediction.** Different methods have evolved for processing the spatial and temporal dependencies to predict future behavior. RNNs can predict time series by maintaining a hidden state while processing temporal information. However, they can be difficult to train for larger time series due to vanishing or exploding gradients caused by their recurrent structure. Therefore, more advanced recurrent models like gated recurrent units [34] or LSTMs [72] have been proposed and applied for trajectory prediction [5, 7, 67, 86].

Since only extracting temporal information does not account for dependencies between traffic participants, Alahi et al. [5] develop Social LSTM, which processes the past states of each agent with a separate LSTM and predicts future trajectories for all agents in the scene. To also encode social interaction, the authors propose a social pooling layer, which shares the hidden state of the encoders of spatially close agents.

CNNs have also been widely implemented for spatial information aggregation [26, 37, 73, 74]. Other methods are fully connected [78] or graph neural networks [46]. Generative adversarial networks proposed by Goodfellow et al. [62] have also been considered for trajectory prediction [67, 96, 215]. For example, Gupta et al. [67] encode the trajectories of agents with LSTMs and then employs a

pooling module to share information between agents, similar to Social LSTM [5]. The approach called SGAN [67] uses a generative adversarial network to output multi-model trajectories that look socially acceptable. Zhao et al. [215] propose to fuse the information from multiple single-agent LSTM encoders. The resulting MATF architecture fuses the information from multiple agents and encodes the scene context. It allows the prediction of all agents' future trajectories in a single forward pass. Other combinations of the presented approaches can be found for trajectory prediction [13, 154].

Our main difference is that we solely use a convolutional architecture for jointly encoding temporal and spatial information and decoding the resulting features for prediction without recurrent structures. This results in a simpler architecture that is easier to train and faster to run.

The use of CNNs instead of RNNs for sequence modeling has been proposed by Bai et al. [12]. They argue that such temporal convolutional neural networks can outperform recurrent models for specific tasks while being more straightforward to train. In addition, temporal convolutional neural networks can process the data in parallel, making them faster at inference. Nikhil and Morris [131] propose to use CNNs for pedestrian trajectory prediction and show superior performance and speed compared to LSTM-based models. However, their method does not consider neighboring agents since they only use 1D convolutions along the time dimension. Temporal convolutions inspire the approach developed in this work and provide, in contrast to Nikhil and Morris [131], a joint aggregation of spatial and temporal features from the proposed novel tensor input representation.

## 7.4 Conclusion

This chapter presented a novel approach to classify a target vehicle's future lane change maneuver and predict the corresponding trajectory. Our model operates on a 3D spatio-temporal input representation encoding the neighborhood information around the target. We exploit the local correlations in neighboring state sequences with spatio-temporal 2D convolutions, resulting in a simple, memory-efficient architecture with fast inference. This allowed us to successfully account for different possible driving maneuvers that we can use to make more informed predictions for all agents in the scene.

We implemented and evaluated our approach on two datasets taken at different locations with respect to the RMSE between the predicted and ground truth trajectories. Further, we provided comparisons to other existing techniques, and the experiments suggested that our approach can successfully predict maneuver-based trajectories and estimate trajectories closer to the ground truth compared to state-of-the-art methods. Finally, we carried out an ablation study to investigate

the impact of different parts of the architecture on the prediction performance. Our experimental evaluation backed up all claims made in this chapter.

The presented approach provided an answer to the question "Where is an object moving to?", but only for structured environments like highways and given previously detected and tracked trajectories of other vehicles. In the next chapter, we aim to omit these restrictions to obtain a method that predicts a future state of the surroundings independent of the environment and prior perception outputs.

# Chapter 8

# Self-Supervised Point Cloud Prediction

Exploiting past 3D LiDAR scans to predict future point clouds is a promising method for autonomous mobile systems to realize foresighted state estimation, collision avoidance, and planning. Estimating the future scene on the sensor level requires no preceding steps as in localization or tracking systems. We can train our point cloud prediction self-supervised without the need for expensive labeling and evaluate its performance online in unknown environments since the following incoming LiDAR scans always give the ground truth data.

In contrast to most approaches, which predict, for example, future locations of traffic agents as demonstrated in Chap. 7, point cloud prediction does not need any preceding inference steps such as localization, detection, or tracking to predict a future scene. Running an off-the-shelf detection and tracking system on the predicted point clouds can yield future 3D object bounding boxes as demonstrated by recent approaches for point cloud forecasting [111, 196]. Besides that, our approach presented in Chap. 7 uses an explicit neighborhood encoding for structured environments like highways, which does not work for general scenarios.

This chapter addresses the problem of predicting large and unordered future point clouds from a given sequence of past scans, as shown in Fig. 8.1. This idea provides a more general answer to the "Where is an object moving to?" question by reasoning about the future evolution of a scene solely based on the geometric information from a point cloud sequence. This idea aligns with our initial works on MOS presented in Part I, which have shown to be robust to the dynamic environment and agnostic to the semantic class of moving objects.

High-dimensional and sparse 3D point cloud data render point cloud prediction a challenging problem not yet fully explored in the literature. We can estimate a future point cloud by applying a predicted future scene flow to the

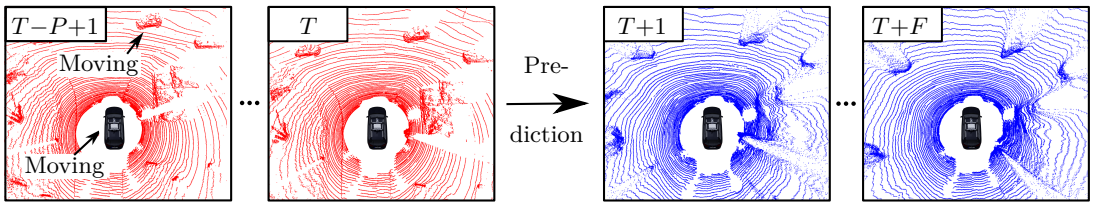



Figure 8.1: Given a sequence of $N$ past point clouds (left in red) at time $T$, the goal is to predict the $F$ future scans (right in blue). Note that all point clouds are in the sensor's coordinate system. The figure is best viewed in color.

last received scan or by generating a new set of future points. This chapter focuses on generating new point clouds to predict the future scene. In contrast to existing approaches [111, 196], which exploit recurrent neural networks for modeling temporal correspondences, we take inspiration from video prediction approaches [3] and use spatio-temporal convolutions to encode the spatial and temporal information jointly. We propose an end-to-end approach that exploits a 2D range image representation of each 3D LiDAR scan and concatenates a sequence of range images to obtain a 3D tensor. Based on such tensors, we develop an encoder-decoder architecture using dense 3D convolutions to jointly aggregate spatial and temporal information of the scene and estimate a future range image and per-point scores for being a valid or an invalid point for multiple future time steps. This allows us to predict detailed future point clouds of varying sizes with fewer parameters to optimize, resulting in faster training and inference times.

Furthermore, our approach is also fully self-supervised and does not require manual data labeling. We evaluate our method on multiple datasets, and the experimental results suggest that our method outperforms existing point cloud prediction architectures and generalizes well to new, unseen environments without additional fine-tuning. Our method operates online faster than the common LiDAR frame rate of 10 Hz using a consumer-grade GPU.

## 8.1 3D Spatio-Temporal Convolutions For Point Cloud Prediction

The main contribution of this chapter is a novel range image-based encoder-decoder neural network to process spatio-temporal information from points clouds using 3D convolutions jointly. Our method can obtain structural details of the environment using skip connections and horizontal consistency using circular padding. Ultimately, it provides more accurate predictions than other state-of-the-art approaches for point cloud prediction. We make three main key claims: Our approach can (i) predict a sequence of detailed future 3D point clouds from a given input sequence by a fast joint spatio-temporal point cloud processing

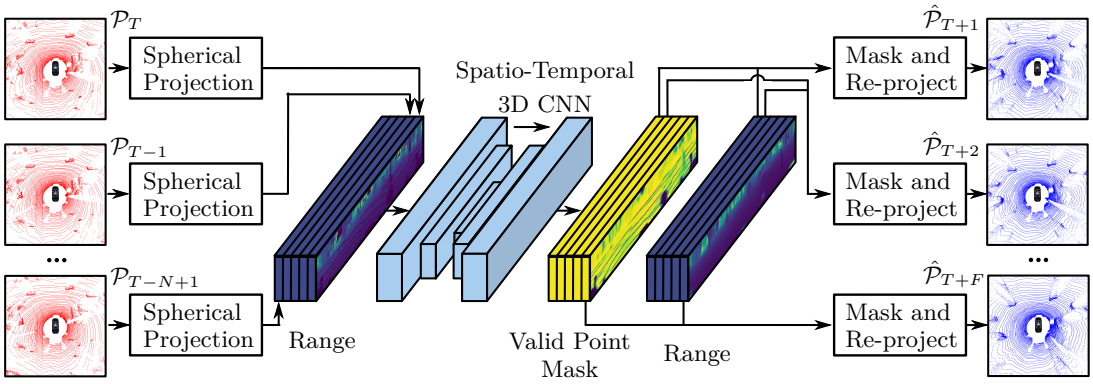



Figure 8.2: Overview of our approach. At time step $T$, the past point clouds are first projected into a 2D range image representation and then concatenated. After passing through our proposed spatio-temporal 3D CNN network, the combined predicted mask and range are re-projected to obtain future 3D point cloud predictions.

using temporal 3D CNNs, (ii) outperform state-of-the-art point cloud prediction approaches, and (iii) generalize well to unseen environments and operate online faster than a typical rotating 3D LiDAR sensor frame rate of 10 Hz on a consumer-grade GPU. This chapter and our experimental evaluation back up these claims. The open-source code and the trained models are available at `https://github.com/PRBonn/point-cloud-prediction`.

### 8.1.1 Overview

Our approach aims to predict a future sequence of $F$ full-scale LiDAR points $\{\mathcal{P}_{T+1}, \dots, \mathcal{P}_{T+F}\}$ given a sequence of $N$ past scans $\{\mathcal{P}_T, \dots, \mathcal{P}_{T-N+1}\}$ with $\mathcal{P}_t = \{\boldsymbol{p}_i \in \mathbb{R}^3\}$. As illustrated in the overview of our method in Fig. 8.2, we first project each past 3D point cloud into a range image representation and concatenate the 2D images to obtain a 3D spatio-temporal tensor, see Sec. 8.1.2 for details. Second, we pass this tensor to our encoder-decoder architecture to extract the spatial and temporal correlations with 3D convolutional kernels as described in Sec. 8.1.3. We use skip connections and circular padding to maintain details and horizontal consistency of the predicted range images, see Sec. 8.1.4. We provide details on the training procedure in Sec. 8.1.5.

### 8.1.2 Point Cloud Sequence Representation

This section provides details on our range image representation. We convert each 3D LiDAR point $\boldsymbol{p} = [x, y, z]^\mathsf{T}$ to spherical coordinates and map them to image coordinates $[u, v]^\mathsf{T}$ resulting in a projection $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\left(1 - \arctan(y, x)\, \pi^{-1}\right) W \\ \left(1 - \left(\arcsin(z\, r^{-1}) + \mathrm{f}_{\mathrm{up}}\right) \mathrm{f}^{-1}\right) H \end{bmatrix}, \tag{8.1}$$

with $(H, W)$ as the height and width of the range image and $\mathrm{f} = \mathrm{f}_{\mathrm{up}} + \mathrm{f}_{\mathrm{down}} \in \mathbb{R}$ is the vertical field of view of the sensor. Each range image pixel stores the range value $r = ||\boldsymbol{p}||_2$ of the projected point or zero if no projected point exists in the corresponding pixel. If multiple points project to the same pixel due to rounding, we keep the closer one since closer points are more important than far-away points. We re-project a range image pixel with coordinates $u$ and $v$ and range $r$ solving Eq. (8.1) using pitch $\theta = \arctan(y, x)$, yaw $\gamma = \arcsin(z\, r^{-1})$, and computing

$$[x,\, y,\, z]^{\mathsf{T}} = [r\cos(\theta)\cos(\gamma),\, r\cos(\theta)\sin(\gamma),\, r\sin(\theta)]^{\mathsf{T}}\,. \tag{8.2}$$

In contrast to the existing work using range images [30, 125], we found that additionally using $x, y, z$, and intensity values did not improve the performance of point cloud prediction. We concatenate the range images along the temporal dimension to obtain the input 3D tensor with a size of $N \times H \times W$, which is further processed by our encoder-decoder network.

### 8.1.3 Spatio-Temporal Encoder-Decoder Architecture

Our approach first converts the 3D LiDAR points into spherical coordinates and maps them to image coordinates. This results in a dense 2D range image from which we can recover the 3D information, see Sec. 8.1.2. The main task of the encoder-decoder architecture illustrated in Fig. 8.3 is to jointly extract spatio-temporal features from the input sequence and output future range image predictions. Compared to a multilayer perceptron architecture that disregards spatial and temporal dimensions, convolutions impose an inductive bias of having local correlations along these dimensions. Convolutional networks usually require less trainable parameters and are less prone to overfitting. Note that we could also use a sparse 4D representation introduced for 4DMOS in Chap. 3 for this task. However, the prediction task is more complex than segmentation because we need to generate a sequence of new point clouds, which is not straightforward for such an architecture. We, therefore, take inspiration from video prediction and use 2D projections over time, resulting in a dense 3D tensor.

When using range images for point cloud prediction with 3D convolutions, we require sufficient receptive fields along the spatial and temporal dimensions between the encoder and decoder to capture the motion of points in the past frames and to propagate their locations into the future range images. The encoder takes the 3D input tensor $N \times H \times W$, containing $N$ range images with height $H$ and width $W$, and first standardizes the range values based on mean and standard deviation computed from the training data. We pass the standardized tensor through a 3D convolutional input layer with $C$ kernels to get a $C$-channel feature representation of size $C \times N \times H \times W$. Inspired by FutureGAN [3], the encoder

block receives a tensor of size $C_l \times N_l \times H_l \times W_l$ at each encoder stage $l$, applies a combination of 3D convolution, 3D batch normalization [80], and the leaky rectified linear unit activation function [115] while maintaining the size of the tensor. A strided 3D convolution follows resulting in a downsampled tensor of size $C_{l+1} \times (N_l - p_l) \times \frac{H_l}{h_l} \times \frac{W_l}{w_l}$ where $h_l$ and $w_l$ are predefined downsampling factors for the spatial feature size and $p_l$ redces the temporal feature dimension. We batch normalize the resulting tensor and apply a leaky rectified linear unit activation function. The kernel size is $p_l \times h_l \times w_l$ and the stride $(1, h_l, w_l)$ to achieve the desired downsampling. The downsampling compresses the sequential point cloud feature representation and forces the network to learn meaningful intermediate spatio-temporal features.

To predict future range images for $F$ future time steps, the decoder subsequently upsamples the feature tensor to the final output size $2 \times F \times H \times W$. Note that we fix the number of future range images in the architecture. Still, we can achieve longer prediction horizons by re-training with a larger output size or in an auto-regressive manner by sequentially feeding back the predicted range images as the input tensor. This work will only focus on predicting a fixed number of future point clouds. The decoder architecture is a mirrored version of the encoder. First, a transposed 3D convolutional layer with kernel size $p_l \times h_l \times w_l$ and stride $(1, h_l, w_l)$ increases the feature map. We insert a 3D batch normalization and leaky rectified linear unit activation layer before a second 3D CNN, 3D batch normalization, and leaky rectified linear unit follows. Finally, we pass the output tensor $C \times F \times H \times W$ through a 3D CNN output layer with two kernels of size $1 \times 1 \times 1$ and stride $(1, 1, 1)$ and apply the final sigmoid function to get normalized values between 0 and 1. The first output channel maps to a predefined range interval, resulting in future range predictions. As done by Weng et al. [196], the second channel contains a probability for each range image point to be valid for re-projection. The re-projection mask keeps all points with a probability above 0.5. This makes it possible to mask out, for example, the sky for which no ground truth points are available.

### 8.1.4 Skip Connections and Horizontal Circular Padding

We use strided 3D convolutions and downsample the range images as described in Sec. 8.1.3. The reduced feature space size causes a loss of details in the predicted range of images. We address this problem by adding skip connections [152] between the encoder and decoder to maintain details from the input scene. As shown in Fig. 8.3, the feature maps bypass the remaining encoding steps, and the mirrored decoder stage concatenates them with the previously upsampled feature volume along the channel dimension. Concatenation enables the network to account for the temporal offset between encoder and decoder feature maps.

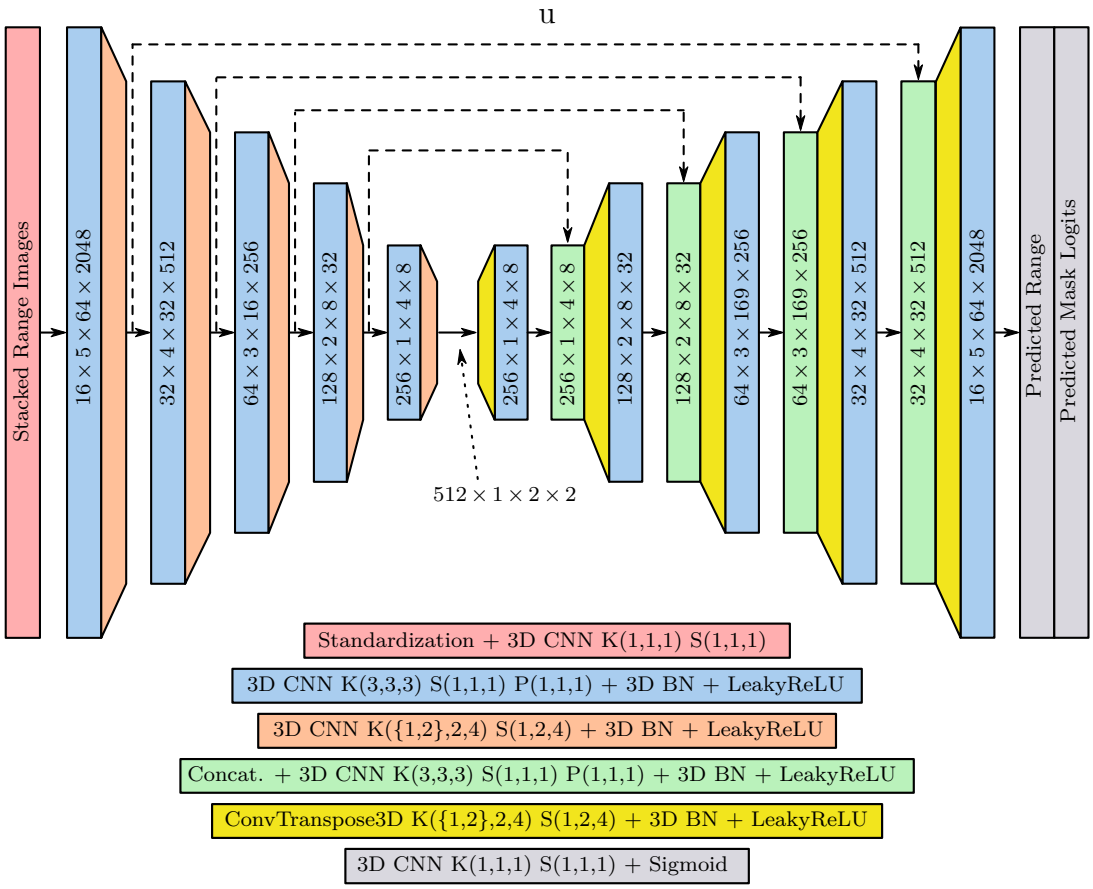



Figure 8.3: Our spatio-temporal architecture using a 3D CNN with feature map sizes $C_i \times N_i \times H_i \times W_i$. Solid and dashed arrows indicate information flow and skip connections, respectively. Details of each block are in the colorized boxes with kernel sizes K, stride S, and padding P (if used). The figure is best viewed in color.

A combination of 3D convolutions, 3D batch normalization, and leaky rectified linear unit follows to merge the features to the original number of channels while maintaining the temporal and spatial dimensions. We investigate the effect of skip connections in Sec. 8.2.2 and Sec. 8.2.5 and show that they maintain details in the predicted point clouds.

Another challenge when using range images for 3D point cloud prediction is maintaining spatial consistency on the horizontal borders of the range images. Range images obtained by rotating LiDAR sensors, such as a Velodyne or an Ouster sensor, are panoramic images with strong horizontal correlations between the image borders. For example, if the ego vehicle is rotating along the vertical $z$ axis, an object passing the left border of the range image will appear on the right border. To consider this property, we introduce circular padding for the width dimension. We pad the left side of each feature map with its right side and vice versa. We pad the vertical dimension with zeros. We provide an experiment on the effectiveness of this padding in Sec. 8.2.2.

### 8.1.5 Training

When training the network, we project the ground truth point clouds into the range images $H \times W$, which allows for the computation of 2D image-based losses. We slice the data into samples of sequences consisting of $N$ past and $F$ future frames, where subsequent samples are one frame apart. We train our architecture with a combination of multiple losses. We define the average range loss $\mathcal{L}_{\mathrm{R},t}$ at time step $t$ between a predicted range image $\{\hat{r}_{i,j}\}_t \in \mathbb{R}^{H \times W}$ and a ground truth range image $\{r_{i,j}\}_t \in \mathbb{R}^{H \times W}$ as

$$\mathcal{L}_{\mathrm{R},t} = \frac{1}{HW} \sum_{i,j} \Delta r_{i,j}, \quad \text{with} \quad \Delta r_{i,j} = \begin{cases} \|\hat{r}_{i,j} - r_{i,j}\|_1, & \text{if } r_{i,j} > 0 \\ 0, & \text{otherwise} \end{cases}, \tag{8.3}$$

such that we only compute the range loss for valid ground truth points. We train the mask output at time step $t$ with predicted probabilities $\{\hat{m}_{i,j}\}_t \in \mathbb{R}^{H \times W}$ with a binary cross-entropy loss

$$\mathcal{L}_{\mathrm{M},t} = \frac{1}{HW} \sum_{i,j} -y_{i,j} \log(\hat{m}_{i,j}) - (1 - y_{i,j}) \log(1 - \hat{m}_{i,j}), \tag{8.4}$$

where $y_{i,j}$ is 1 if the ground truth point is valid and 0 otherwise. Both losses only consider the predicted 2D range images and not the re-projected point clouds, which makes them fast to compute. However, we also need to ensure 3D consistency of the predicted point clouds with the corresponding ground truth ones. A common metric for comparing 3D point clouds is the Chamfer distance [51]. There are different definitions of the Chamfer distance, and we will use the squared version from Fan et al. [51]. At time step $t$, we re-project the masked range image into a 3D point cloud $\hat{\mathcal{P}}_t = \{\hat{\boldsymbol{p}} \in \mathbb{R}^3\}, |\hat{\mathcal{P}}_t| = N$, and compare it with the ground truth point cloud $\mathcal{P}_t = \{\boldsymbol{p} \in \mathbb{R}^3\}, |\mathcal{P}_t| = M$, by computing

$$\mathcal{L}_{\mathrm{CD},t} = \frac{1}{N} \sum_{\hat{\boldsymbol{p}} \in \hat{\mathcal{P}}_t} \min_{\boldsymbol{p} \in \mathcal{P}_t} \|\hat{\boldsymbol{p}} - \boldsymbol{p}\|_2^2 + \frac{1}{M} \sum_{\boldsymbol{p} \in \mathcal{P}_t} \min_{\hat{\boldsymbol{p}} \in \hat{\mathcal{P}}_t} \|\hat{\boldsymbol{p}} - \boldsymbol{p}\|_2^2. \tag{8.5}$$

The computation of the Chamfer distance is generally slow due to the search for nearest neighbors. A fast-to-compute image-based loss using $\mathcal{L}_{\mathrm{R},t}$ and $\mathcal{L}_{\mathrm{M},t}$ is about 2.5 times faster compared to a loss including the 3D Chamfer distance. We propose a training scheme with only range and mask loss for pre-training. This results in a good initialization for fine-tuning, including a Chamfer distance loss. We provide an experiment on the training scheme in Sec. 8.2.5. Given a current time step $t = T$, our total loss function for $F$ future time steps is given by

$$\mathcal{L} = \sum_{t=T+1}^{T+F} \mathcal{L}_{\mathrm{R,t}} + \alpha_{\mathrm{M}} \mathcal{L}_{\mathrm{M,t}} + \alpha_{\mathrm{CD}} \mathcal{L}_{\mathrm{CD,t}}, \tag{8.6}$$

with tunable weighting parameters $\alpha_{\mathrm{M}}$ and $\alpha_{\mathrm{CD}}$.

### 8.1.6 Implementation Details

For the loss weights, we experienced that setting $\alpha_{\mathrm{M}} = 1.0$ and $\alpha_{\mathrm{CD}} = 0.0$ for pre-training and $\alpha_{\mathrm{M}} = 1.0$ and $\alpha_{\mathrm{CD}} = 1.0$ for fine-tuning worked best. We train our network parameters with the Adam optimizer [92], a learning rate of $10^{-3}$, and accumulate the gradients for 16 samples. We use an exponential decaying learning rate schedule. All models require less than 50 epochs to converge.

## 8.2 Experimental Evaluation

This work presents a self-supervised point cloud prediction approach using spatio-temporal 3D convolutional neural networks. We present our experiments to show the capabilities of our method and to support our key claims, which are: (i) predicting a sequence of detailed future 3D point clouds from a given input sequence by a fast joint spatio-temporal point cloud processing using temporal 3D convolutional networks, (ii) outperforming state-of-the-art point cloud prediction approaches, and (iii) generalizing well to unseen environments and operating online faster than a typical rotating 3D LiDAR sensor frame rate of 10 Hz on a consumer-grade GPU. We follow the experimental setting of MoNet [111] with the KITTI Odometry dataset [58] and predict the next five frames from the past five LiDAR scans captured at 10 Hz.

### 8.2.1 Experimental Setup

We follow the experimental setup of MoNet [111] and train the models on sequences 00 to 05 of the KITTI Odometry dataset [58], validate on sequences 06 and 07, and test on sequences 08 to 10. We project the point clouds from the Velodyne HDL-64E laser scanner into range images $H \times W = 64 \times 2048$ according to Eq. (8.1). As explained in Sec. 8.1.3, we map the normalized output values of the network to a predicted range interval defined before training. Based on the minimum and maximum range values of the training data, we set the minimum predicted range to 1 m and the maximum range to 85 m for KITTI Odometry [58] and to 110 m for the Apollo dataset [113] used in the generalization experiment in Sec. 8.2.6.

### 8.2.2 Qualitative Results

The first experiment supports our claim that our model can predict a sequence of detailed future 3D point clouds using a temporal convolutional network. We show the re-projected point clouds for all steps of a scene from the KITTI Odometry sequence 08, which was not used during training in Fig. 8.4. Our method can

estimate the ego motion of the vehicle, resulting in an accurate prediction of the environment. The moving bicyclist in front of the ego vehicle is consistently predicted in all five future frames, demonstrating that our predicted range images preserve details in the scene. For a more detailed inspection, we provide a cutout of the predicted and ground truth range and mask images and the combined masked prediction for the last prediction step in Fig. 8.5. One can see that our approach successfully predicts the correct range for close and far-away points as well as the moving bicyclist in front of us. Additionally, our mask prediction manages to mask out the sky at the top and the shadow from the ego vehicle at the bottom.

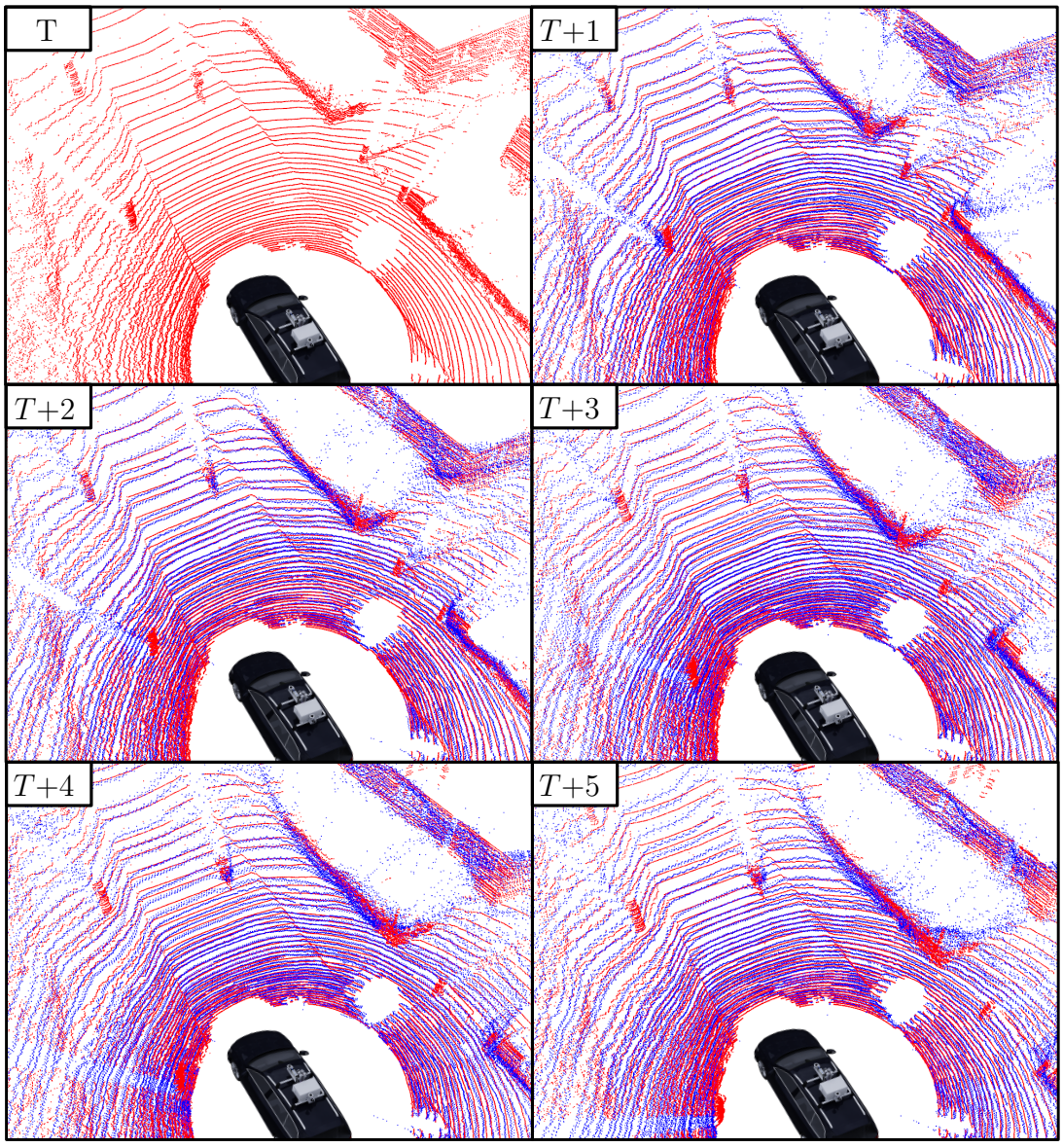



Figure 8.4: Last received point cloud at time $T$ and the predicted next five future point clouds. We show ground truth points in red and predicted points in blue. The scene is from the KITTI Odometry sequence 08, which we did not use during training.

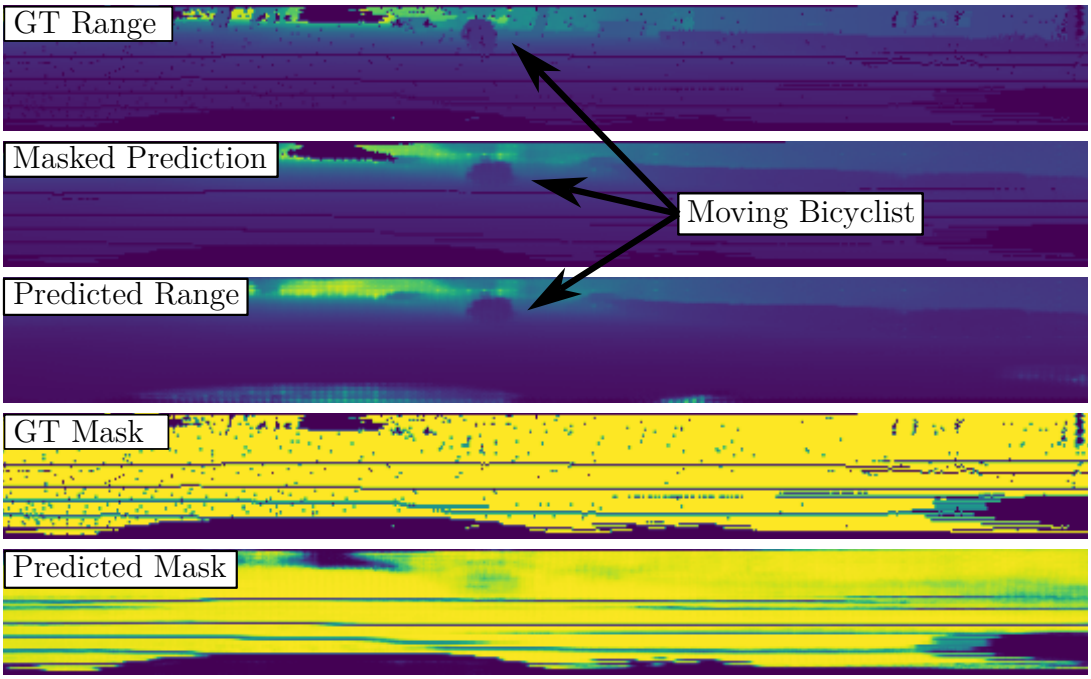



Figure 8.5: Ground truth and predictions of range image and re-projection mask at the last prediction step. The range is color-coded with increasing range from blue to yellow. We show invalid points in dark blue. Yellow indicates a valid point for the mask, whereas dark blue represents invalid points. The masked prediction is the combination of the predicted range and re-projection mask.

### 8.2.3 Quantitative Results

We compare the quantitative results to multiple baselines to support the claim that our method outperforms state-of-the-art point cloud prediction approaches. Lu et al. [111] operate and evaluate on sub-sampled point clouds, whereas our approach can predict full-size point clouds. To account for this, Tab. 8.1 (*left*) shows the performance of our method on sub-sampled point clouds with two different sampling rates used in MoNet [111]. Our method achieves a better result for larger prediction steps for both samplings. However, sampling points from the original point clouds for evaluation strongly affects the Chamfer distance of our full-sized prediction. Thus, we provide an evaluation on full-scale point clouds to investigate the performance of our approach in Tab. 8.1 (*right*). We compare our results to an *Identity* baseline that takes the last received scan as a constant prediction for all future time steps. The *Constant Velocity* and *Ray Tracing* baselines use the transformation between the previous two received scans from a SLAM approach [17] for a constant velocity prediction of the sensor. The *Constant Velocity* baseline transforms the last received point cloud according to the predicted motion at each prediction step, and the *Ray Tracing* baseline aggregates all transformed past points to render a denser range image. Since many points in the scene are static, these baselines work well for constant ego motion.

| | Sampled Point Clouds | | | | | Full-scale Point Clouds | | | |
|---|---|---|---|---|---|---|---|---|---|
| Prediction Step | PointLSTM [53] | Scene Flow [111] | MoNet [111] | Ours 32,768 pts | Ours 65,536 pts | Identity Baseline | Const. Vel. Baseline | Ray Tracing Baseline | Ours |
| 1 | 0.332 | 0.503 | **0.278** | 0.367 | 0.288 | 0.271 (0.144) | **0.105** (0.067) | 0.158 (0.095) | 0.254 (0.124) |
| 2 | 0.561 | 0.854 | 0.409 | 0.446 | **0.352** | 0.719 (0.448) | **0.217** (0.221) | 0.253 (0.246) | 0.310 (0.188) |
| 3 | 0.810 | 1.323 | 0.549 | 0.546 | **0.428** | 1.216 (0.798) | 0.385 (0.443) | 0.388 (0.460) | **0.378** (0.262) |
| 4 | 1.054 | 1.913 | 0.692 | 0.638 | **0.509** | 1.727 (1.169) | 0.604 (0.711) | 0.556 (0.719) | **0.448** (0.370) |
| 5 | 1.299 | 2.610 | 0.842 | 0.763 | **0.615** | 2.240 (1.156) | 0.852 (0.968) | 0.751 (0.971) | **0.547** (0.487) |
| Mean (Std) | 0.811 | 1.440 | 0.554 | 0.552 | **0.439** | 1.235 (1.192) | 0.433 (0.641) | 0.421 (0.627) | **0.387** (0.331) |

Table 8.1: Chamfer distance results on KITTI Odometry test sequences 08 to 10 in $\left[\mathrm{m}^2\right]$ with sampled (*left*) and full-scale (*right*) points clouds. We use the Chamfer distance defined in Eq. (8.5). We first train our model on range and mask loss and then fine-tune, including a Chamfer distance loss for 10 epochs as described in Sec. 8.1.5. The best mean results are in bold for sampled and full-scale point clouds. For full-scale point clouds, we report the standard deviation of the Chamfer distance in parentheses. See Sec. 8.2.4 for a more detailed statistical analysis.

The results in Tab. 8.1 show that our method outperforms all baselines at larger prediction steps. Our approach can reason about the sensor's ego motion and the motion of moving points, while all three baselines do not consider points belonging to moving objects. The standard deviation of the Chamfer distance indicates that the results of our approach are more consistent throughout the test set. In contrast, the baseline results suffer from outliers in the case of moving objects or non-linear ego motion.

### 8.2.4 Statistical Analysis

For a more detailed quantitative analysis for full-scale point cloud prediction, we compare box plots for all prediction steps in Fig. 8.6. Each plot shows the median (orange line), minimum (lower bar), maximum (upper bar), first quartile (lower colorized rectangle), and third quartile (upper colorized rectangle) of the Chamfer distances computed between the predicted and ground truth point clouds for the test set. As previously discussed, the constant velocity baseline achieves the lowest median Chamfer distances for short prediction horizons. Since all baselines ignore moving objects for the prediction, our method outperforms them for larger time steps, resulting in lower median and interquartile ranges. The statistical results support our claims made in this chapter.

### 8.2.5 Ablation Study

In this experiment, we provide a more detailed analysis of the effect of the model architecture and the proposed training scheme. We average the metrics over all validation samples and time steps as shown in Tab. 8.2. We always report the Chamfer distance, but it is only used during training if specified. First, we modify the size of the network by decreasing (*Small*) or increasing (*Large*) the number of channels of our model by a factor of two. The larger model predicts slightly better-masked range images but requires four times more parameters.

All losses increase without skip connections (*Skip Con.*), and many objects are not predicted due to the large feature compression. The effect of circular padding (*Circular Padding*) is not visible quantitatively because of the small number of affected points. Finally, we show that our fine-tuned model (*Ours*) achieves similar performance on the validation set compared to a model optimized with the Chamfer distance loss from the beginning (*CD Loss*). Our proposed training scheme with pre-training on fast-to-compute image-based losses gives a good pre-trained model for fine-tuning with the 3D Chamfer distance loss while reducing training time.

The situation in Fig. 8.7 from test sequence 08 shows an ego-vehicle finishing a left turn while passing a vehicle. This scenario is challenging because of the dif-

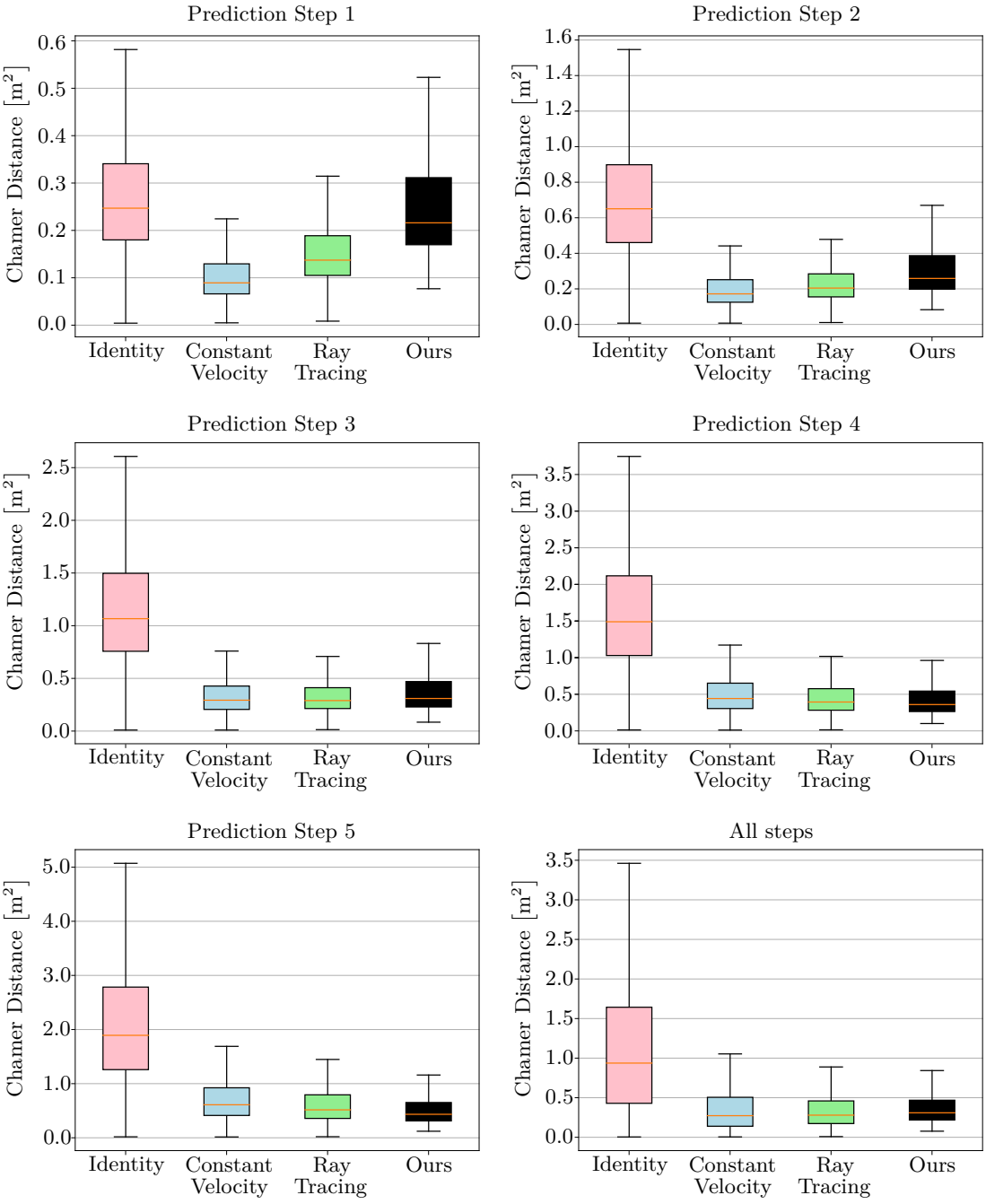



Figure 8.6: Comparison of Chamfer distance results for each prediction step.

ferent scales of close and far objects and the ego motion of the car. Our approach, pre-trained on range and mask loss only, predicts the future point clouds across five time steps and maintains details in the scene. Small objects are lost without skip connections due to the compression (green circle). Without circular padding, the points belonging to the car in the back of the ego vehicle (black circle) are not consistently predicted since they span across the range image borders.

| Property | Small | Large | Skip Con. | Circular Padding | CD Loss | Pre-trained | Ours |
|---|---|---|---|---|---|---|---|
| 4 M Parameters | ✗ | | | | | | |
| 17 M Parameters | | | ✗ | ✗ | ✗ | ✗ | ✗ |
| 68 M Parameters | | ✗ | | | | | |
| Skip Connections | ✗ | ✗ | | ✗ | ✗ | ✗ | ✗ |
| Circular Padding | ✗ | ✗ | ✗ | | ✗ | ✗ | ✗ |
| Loss Weight $\alpha_{\text{CD}}$ | 0 | 0 | 0 | 0 | 1 | 0 | $0 \rightarrow 1$ |
| $\mathcal{L}_{\text{R}}$ [m] | 0.867 | **0.741** | 1.258 | 0.801 | 0.805 | 0.798 | 0.858 |
| $\mathcal{L}_{\text{Mask}}$ | 0.309 | **0.288** | 0.337 | 0.297 | 0.300 | 0.299 | 0.301 |
| $\mathcal{L}_{\text{CD}}$ $\left[\text{m}^2\right]$ | 1.110 | 0.714 | 2.759 | 0.885 | 0.487 | 0.985 | **0.480** |

Table 8.2: Evaluation of models with varying architectures, padding, and training schemes on the KITTI Odometry validation sequences 06 and 07. Each variation (columns) is given by a combination of different design decisions (rows) indicated by crosses (✗), for example, variation *Large* is a model with 68 M parameters using skip connections and circular padding. The arrow ($\rightarrow$) indicates a pre-training and fine-tuning with two different loss weights $\alpha_{\text{CD}}$ as discussed in Sec. 8.1.5. The best results are in bold.

### 8.2.6 Generalization

Finally, we aim to support our claim that our method generalizes well to new, unseen environments. We first evaluate a model fine-tuned on KITTI Odometry on the Apollo-SouthBay ColumbiaPark [113] test set. Note that this data is from an unseen, *different* environment (collected in the U.S.) with a *different* type of car setup, compared to the training data (collected in Germany). Our model achieves a mean Chamfer distance of $0.934\,\text{m}^2$ on the Apollo test set. The performance gap compared to the results in Tab. 8.1 is because the training data has a maximum range of 85 m, but the Apollo test data contains points up to a range of 110 m. Thus, we fine-tune the model on the Apollo-SouthBay Columbia-Park [113] training set for a single epoch. This yields a mean Chamfer distance of $0.426\,\text{m}^2$ and demonstrates the generalization capability of our approach and the advantage of self-supervised training.

We provide an additional experiment on the effect of using more training data in Sec. 8.2.7. Fig. 8.8 shows that the fine-tuned model reduces the Chamfer distance by predicting more accurate shapes and more distant points. Compared to Fig. 8.7, the different sensor mount causes a changed pattern of invalid points around the car. Our approach can correctly predict the re-projection mask.

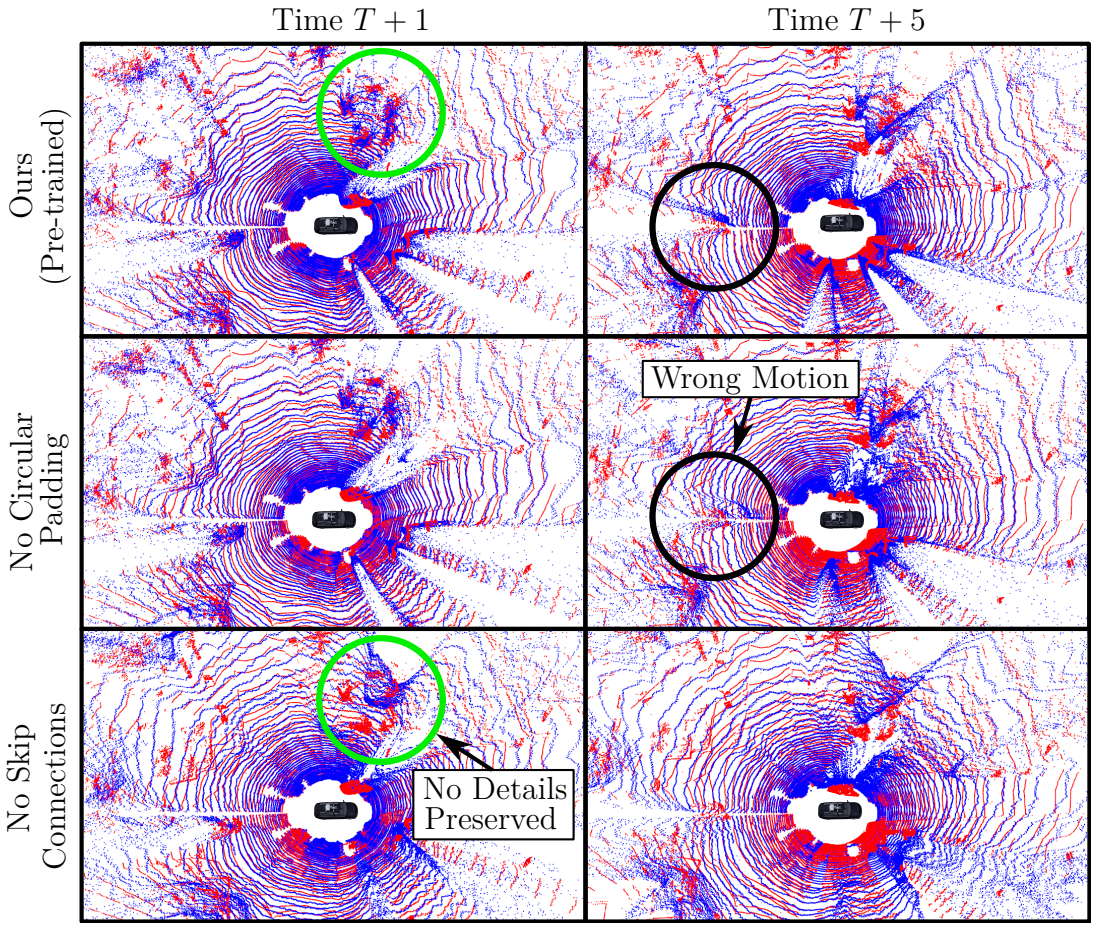



Figure 8.7: Qualitative comparison of our pre-trained method and two ablated models with predicted (blue) and ground truth (red) point clouds from test sequence 08. Given a past sequence of five point clouds at time $T$, the upper row shows the first predicted frame at $T+1$, and the lower row shows the last predicted frame at $T+5$. The green circle encloses an area where details are lost without skip connections, and the points in the black circle demonstrate that circular padding maintains spatial consistency at the range image border. The figure is best viewed in color.

### 8.2.7 Amount of Training Data

This experiment demonstrates the advantage of a large amount of data for training our point cloud prediction method. Since the approach is self-supervised, data obtained from the LiDAR sensor can be directly used for training without expensive labeling. Fig. 8.9 shows the total validation loss on sequences 06 and 07 after 50 epochs with respect to the number of training iterations. Note that the final number of training iterations within 50 epochs depends on the total amount of samples in the training data.

We show the validation loss with the previously explained experimental setting in Fig. 8.9 on the left (*yellow*). If we train a model on a larger dataset containing the KITTI sequences 00 to 05 plus 11 to 15, the validation loss decreases faster and

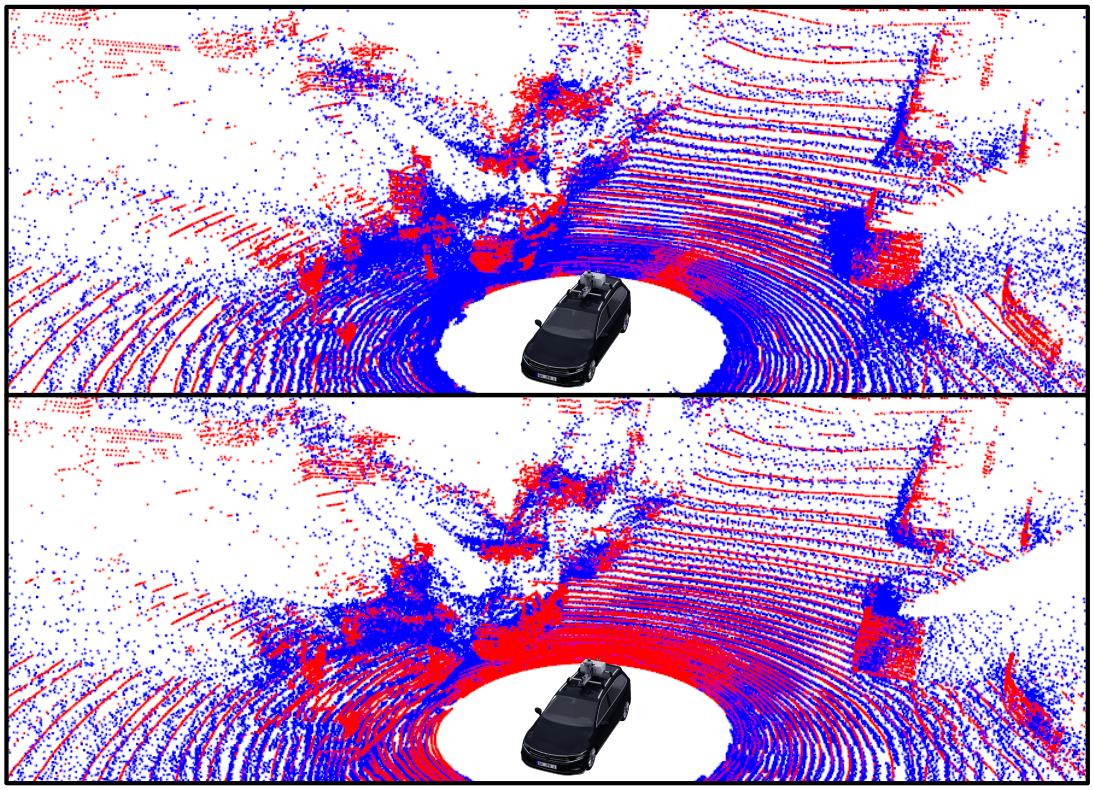


Figure 8.8: Predicted (blue) and ground truth (red) future point clouds on the Apollo-SouthBay ColumbiaPark [113] test set at prediction step 5. *Top*: Model only trained on KITTI odometry data. *Bottom*: After training one epoch on the Apollo training set.

converges to a lower minimum (*blue*). The validation loss is further reduced for the largest dataset containing sequences 00 to 05 plus 11 to 21 (*green*), indicating that more data improves the model's performance on the unseen validation set. The individual loss components in Tab. 8.3 show that we can improve all losses by using more data. This emphasizes that self-supervised point cloud prediction profits from a large amount of sensor data without expensive labeling.

### 8.2.8 Runtime

For the runtime, our approach takes on average 11 ms (90 Hz) for predicting five future point clouds with up to 131,072 points each on a system with an Intel i7-6850K CPU and an Nvidia GeForce RTX 2080 Ti GPU, which is faster than the sensor frame rate of 10 Hz of a typical rotating 3D LiDAR sensor and faster than existing approaches like MoNet [111] using the same GPU. Compared to a runtime of 280 ms reported by Lu et al. [111] for MoNet for predicting five future point clouds with 65,536 points each, our approach is 25 times faster for twice as many points.

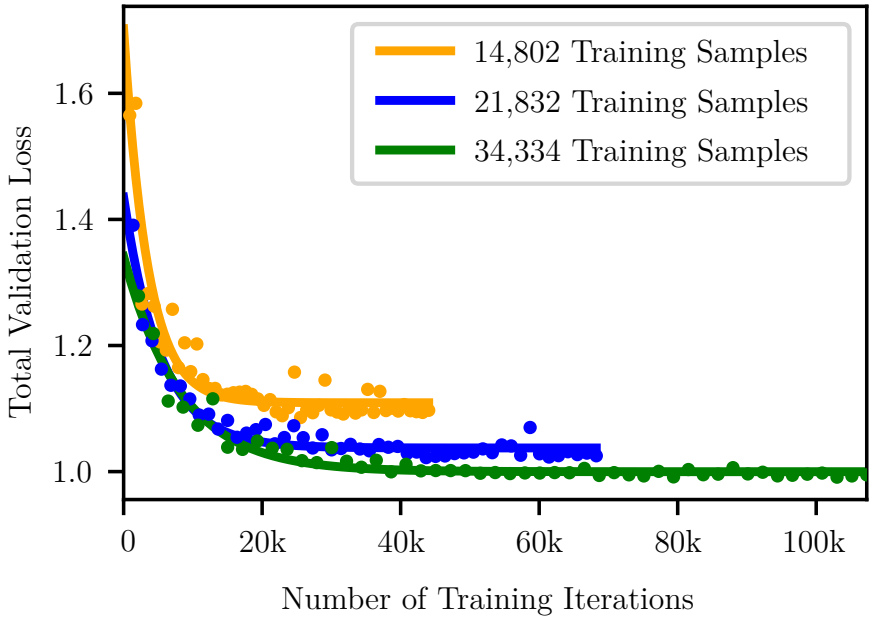



Figure 8.9: Total validation loss for a given number of training iterations with different training data sizes. The solid line is a fitted exponential curve.

| Loss | 14,082 Samples | 21,832 Samples | 34,334 Samples |
|---|---|---|---|
| Range $\mathcal{L}_{\text{R}}$ [m] | 0.798 | 0.735 | **0.710** |
| Mask $\mathcal{L}_{\text{Mask}}$ | 0.299 | 0.290 | **0.285** |
| Chamfer Distance $\mathcal{L}_{\text{CD}}$ [$\text{m}^2$] | 0.985 | 0.795 | **0.768** |

Table 8.3: Final validation loss components for different numbers of training dataset sizes. We train all methods for 50 epochs. The best results are in bold.

## 8.3 Related Work

Extracting spatio-temporal information from point cloud sequences has been exploited in the literature for different tasks related to point cloud prediction. We now review related approaches, such as scene flow estimation, vision, and point-based prediction.

**Scene Flow Estimation.** Liu et al. [109] extend PointNet++ [149], which extracts spatial features from a single point cloud, and propose FlowNet3D for estimating the 3D motion field between two point clouds. Lu et al. [112] use FlowNet3D to interpolate intermediate frames between two LiDAR scans. Other deep learning-based scene flow estimation methods can be found in the literature [183, 213].

In contrast to computing the scene flow between two past frames, point cloud prediction estimates multiple future point clouds. Scene flow requires labels like point-wise correspondences or segmentation masks, which makes learning costly.

Instead, our approach focuses on the self-supervised generation of new, future point clouds based on the observed sensor data.

**Future State Prediction.** Predicting future trajectories of traffic participants based on their past motion is one possible application of LiDAR-based future scene prediction. Given LiDAR scans, these methods rely on preceding detection and tracking modules to infer past trajectories.

For example, Luo et al. [114] develop a 3D convolutional architecture based on a bird's eye view of voxelized LiDAR scans transformed by the sensor's ego motion. Inspired by this, Casas et al. [26] increase the prediction horizon with IntentNet and estimate high-level driver behavior from HD maps. Liang et al. [102] integrate a tracking module into the end-to-end pipeline to improve the temporal consistency of the predictions while Meyer et al. [124] and Laddha et al. [98] explore the range image representation to predict future 3D bounding boxes. A second method to estimate a future scene representation without the need for prior object detections is prediction performed at the sensor level. Hoermann et al. [73] directly predict a future occupancy grid map from a sequence of past 3D LiDAR scans and Song et al. [168] predict a motion flow for indoor 2D LiDAR maps. Wu et al. [198] use a spatio-temporal pyramid network to estimate voxelized bird's eye view classification and future motion maps. Recently, Toyungyernsub et al. [182] differentiate between dynamic and static cells for grid map prediction but require tracking labels to segment moving cells.

In contrast to the approaches above, our method operates on full-size point clouds without voxelization. Weng et al. [196] show a reversed trajectory prediction pipeline that can take the point cloud predictions and detect and track objects in future scans. This makes it possible first to train the prediction of point clouds self-supervised and then use off-the-shelf detection and tracking modules to get future trajectories.

**Vision-based Prediction.** There has been a large interest in vision-based prediction. Srivastava et al. [170] propose an LSTM [72] encoder-decoder model that takes an input sequence of image patches or feature vectors and outputs a future sequence of the same kind. Shi et al. [165] propose ConvLSTMs that employ convolutional structures in the temporal transitions imposed by the LSTM. This makes it possible to encode temporal and spatial correlations in image sequences jointly. The integration of 3D convolutions into LSTMs has been researched by Wang et al. [193]. Aigner et al. [3] omit the use of recurrent structures and develop a video prediction approach called FutureGAN based on 3D convolutions combined with a progressively growing generative adversarial network.

Using the range image representation of LiDAR scans, we can transfer the techniques from vision-based methods for solving point cloud prediction. However, there are differences between video and range image-based point cloud pre-

dictions. The main difference is that range images contain 3D point information rather than RGB values. In addition to that, range images have a 360-degree field of view, which causes a strong correlation between the left and right image borders because of the principle of a typical rotating LiDAR sensor, for example, a Velodyne or an Ouster scanner.

**Point Cloud Prediction.** Though vision-based and voxel-based prediction methods have been well studied, predicting full-scale future point clouds is still challenging and has not yet been fully explored in existing works. Fan et al. [53] propose a point recurrent neural network (PointRNN) to predict a future sequence of unordered and unstructured point clouds. Their method models spatio-temporal local correlations of point features and states. Lu et al. [111] introduce MoNet, which integrates a point-based context and motion encoder network and processes these features in a novel recurrent neural network. A 3D scene flow-based point cloud prediction approach has been recently developed by Deng and Zakhor [42]. All three methods are 3D point-based but only consider a limited number of points.

Weng et al. [196] propose an encoder-decoder network that takes a highly compressed feature representation of a full-scale LiDAR scan and models the temporal correlations with an auto-encoder based on a recurrent neural network. The feature vectors are obtained by a point-based architecture or 2D convolutional neural networks processing 2D range image representations. Deep learning-based generation of LiDAR scans from a 2D range image has also been investigated by Caccia et al. [22].

In contrast to existing approaches, we propose a technique that concatenates the projected 2D range images to a spatio-temporal 3D representation, which makes it possible to use 3D convolutions without the need for recurrent architectures for temporal modeling. As discussed by Bai et al. [12] for general sequence modeling, using convolutional networks results in an architecture that is easier to train and faster at inference since the input range images are no longer processed sequentially. Compared to MoNet [111], this allows us to predict more future point clouds in less time.

## 8.4 Conclusion

This chapter presented a novel approach to self-supervised point cloud prediction with 3D convolutional neural networks. Our method exploits the range image representation to extract spatio-temporal features from the input point cloud sequence jointly. This allows us to successfully predict detailed, full-scale future 3D point clouds during online operation. We can train and evaluate our approach online without needing labeled data.

We implemented and evaluated our approach on different datasets, provided comparisons to other existing techniques, and supported all claims made in this work. The experiments suggested that our method outperforms existing baselines and generalizes well to different sensor mounts in unseen environments.

Our proposed approach addressed the question of "Where is an object moving to?" from a different perspective compared to our previous prediction model presented in Chap. 7 by focusing on the prediction of raw sensor data. This field of research has not yet been fully explored, and it is a promising direction that requires further studies.

Since we also use the range image representation to reduce the computational complexity, we also suffer from label bleeding [125], previously mentioned for segmentation. We observed that the predicted range images do not have sharp edges, leading to intermediate points between two objects after re-projection. We believe that using a sparse 4D architecture as used in Part I of the thesis can mitigate this.

# Chapter 9

# Conclusion

Spatio-temporal perception is a fundamental capability that mobile robots must have to operate safely in unknown and dynamic environments. Equipped with LiDAR sensors, mobile robots can process spatial data over time to accurately assess their surroundings' dynamics. In this thesis, we explored how we can employ CNNs to extract both spatial and temporal information from sequences of LiDAR scans.

We addressed two core questions critical to deploying such systems in real-world environments: The first question, "What is moving?", focuses on identifying moving objects. Distinguishing moving objects from non-moving ones is an important capability to avoid collisions and safely plan trajectories in crowded scenarios. Our work introduced two approaches for segmenting moving objects from either a limited sequence of LiDAR scans or a local map generated from all previous measurements. A key strength of these methods is their robustness to new, unseen objects, as they rely primarily on movement patterns of points rather than object semantics. We evaluated the generalization capabilities of these approaches across unseen environments and proposed a dataset with novel moving object labels from various LiDAR sensors. Additionally, we developed a method to segment long-term dynamic objects that have moved since a prior map was generated, such as parked cars or grown vegetation. Next, we filtered out such unstable points for localizing in the map to increase localization accuracy and robustness.

The second question, "Where is an object moving to?", pertains to predicting the future behavior of other agents to facilitate successful path planning. Initially, we focused on structured environments, such as highways, where we identified a set of high-level maneuvers typical in human driving. Based on past states of nearby vehicles, we developed a system that predicts both the future maneuvers and trajectories of these vehicles. We proposed a self-supervised method that predicts a sequence of future point clouds to mitigate the reliance on labeled training

data and the requirement for prior detection and tracking. This approach offers the advantage of being trainable and evaluable in real-time, enabling continuous adaptation and learning.

## 9.1 Summary of the Key Contributions

In Part I, we introduced 4DMOS in Sec. 3, a system that segments moving objects in real-time from a sequence of LiDAR scans. Utilizing the sparse 4D CNN discussed in Sec. 2.4.2, we developed a strategy to merge multiple predictions for the same scan in a static state binary Bayes filter. We experimentally showed that this reduces the number of false positives and negatives in a probabilistic fashion. Experimental results demonstrated that our method surpasses existing approaches and generalizes effectively to unseen environments and a modified sensor configuration. However, this approach assumes that moving objects remain visible throughout the sequence of past measurements, which often does not hold for sensors with sparse sampling patterns. Additionally, we did not leverage prior knowledge of whether we previously observed a moving object at a specific location, which could indicate that the space is generally free or typically occupied by moving objects.

Our second contribution in Sec. 4 is MapMOS and addressed these challenges by utilizing all available past information. We proposed an architecture that segments moving objects based on a local map of the environment and develop a volumetric belief to model which part of the environment can be occupied by moving objects. Mapping experiments demonstrated that we can effectively use this belief to obtain a static map of the environment for further planning purposes.

For both 4DMOS and MapMOS, we conducted extensive evaluations using data unseen during training to highlight the advantages of our proposed architectures. To bolster our findings, we introduced a new dataset, HeLiMOS, in Sec. 5, which includes moving object labels for all sensors within one sequence of the original HeLiPR dataset. This dataset enabled us to train and evaluate both systems across various sensor types and measurement patterns, showcasing their generalization capabilities.

In addition, we demonstrated in Sec. 6 how we can adapt our MapMOS approach to segment not only moving objects but also unstable ones. We contributed a new system to filter out points that are unstable for localization within a given map. In this case, we aimed not only to segment currently moving points but also to identify generally unstable points, such as those from parked cars or vegetation. This chapter illustrated how the methodologies established in this thesis can facilitate a broader segmentation of spatio-temporal data beyond currently moving objects.

Part II of the thesis contributed to the spatio-temporal prediction of future states for autonomous vehicles. In Chap. 7, we introduced a novel neighborhood representation that captures the past states of neighboring vehicles, accompanied by an architecture based on spatio-temporal 2D convolutions for processing this data. Our architecture allowed us to predict the future maneuvers of vehicles based on their local neighborhoods and to regress future trajectories. We experimentally validated that the neighborhood influences the prediction and showed that the predicted trajectories are closer to the recorded ground truth compared to existing approaches. However, this approach still relied on labeled data for training, requiring prior steps such as detection and tracking.

We tackled this limitation of supervised prediction with our last contribution in Chap. 6, where we introduced a novel architecture capable of predicting a future sequence of point clouds based on a past sequence. We achieved a self-supervised prediction without the need for labels. We again utilized 3D convolutions over time to process our input tensor of stacked range images and proposed two modifications to deal with details in the images and their circular nature.

The methods presented in this thesis make it possible to segment moving objects in a sequence of LiDAR scans or with respect to a local map, regardless of the sensor setup or the environment – a challenge where most MOS systems fall short. It is essential for spatio-temporal perception systems to generalize well to unseen environments and successfully deploy them in the real world. Both work on spatio-temporal prediction demonstrate how we can use convolutions over space and time to predict where an object is moving to, making prediction models more efficient and fast compared to using recurrent networks. Additionally, our proposed point cloud prediction architecture enables us to predict a future state of the environment without the need for object trajectories, which require prior detection and tracking steps. This new paradigm allows us to train and evaluate our generalizable method reliably in unknown environments.

## 9.2 Future Work

While developing the various approaches presented in this thesis, several potential future directions have emerged that could build upon the topics discussed. We have identified four main areas that appear promising for further exploration:

First, we focused primarily on the sparse 4D CNN in Part I due to the complexities involved in the prediction task. While the MOS task necessitates only binary classification, spatio-temporal prediction requires generating new states, such as trajectories or 3D points. Despite this, the sparse 4D architectures have demonstrated strong performance in segmentation and possess the capability to effectively tackle the challenges identified in the second part of the thesis. They

directly operate in the voxelized 3D space over time without projecting to lower dimensional representations like range images. This could potentially enhance the quality of the predicted point clouds discussed in Chap. 8. Using a 4D CNN to predict velocity or flow vectors may lead to more accurate future point cloud sequences. However, our investigation into this area is still in its primary stages and requires further development.

Second, we recognize that we could integrate the two core questions of the theses, "What is moving?" and "Where is an object moving to?", into a single approach that simultaneously segments currently moving points and predicts their future movements. This joint prediction would address the challenge of predicting the positions of non-moving objects within the local sensor frame, which becomes unnecessary if we already estimate the ego motion with an odometry system like KISS-ICP. In a joint estimation, both tasks are highly coupled and can support each other to improve the prediction quality. There are research works [101, 102] that propose an end-to-end prediction that circumvents the need for preceding perception modules. However, such approaches still need training data, and we cannot evaluate the prediction quality online. By comparing a predicted future point cloud with its reference in a self-supervised manner, we hypothesize that this could serve as a supervisory signal for the moving object segmentation, keeping the entire pipeline self-supervised. There is ongoing research in this direction for scene flow and motion [14] as well as occupancy fields [1]. Self-supervision is essential for tasks like prediction, as it minimizes reliance on previously estimated intermediate representations, such as tracked trajectories.

Third, advancing new sensor technologies can further improve the segmentation performance. For example, the Aeva Aeries II sensor from Chap. 5 is a frequency-modulated continuous wave LiDAR providing a radial velocity measurement for the points. Such sensors can drastically help segment moving objects by providing additional motion cues. While we still believe that only relying on a radial velocity measurement is insufficient for most downstream tasks, integrating these measurements into existing approaches is a promising research direction, which has already been explored for radar sensors [211].

Lastly, while sparse CNNs are primarily used in research, they are not yet optimized for deployment on computationally constrained mobile robots. The approaches introduced in this thesis operate at sensor frame rate but depend on GPU availability for rapid computation. We believe substantial research is still needed on these architectures to reduce their size and enhance their computational efficiency.

# Bibliography


[1] B. Agro, Q. Sykora, S. Casas, T. Gilles, and R. Urtasun. UnO: Unsupervised Occupancy Fields for Perception and Forecasting. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[2] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, and B. Yoon. Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN). *Sensors*, 20(12):3344, 2020.

[3] S. Aigner and M. Korner. FutureGAN: Anticipating the Future Frames of Video Sequences using Spatio-Temporal 3d Convolutions in Progressively Growing GANs. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] N. Akai. Efficient Solution to 3D-LiDAR-based Monte Carlo Localization with Fusion of Measurement Model Optimization via Importance Sampling. *arXiv preprint*, arXiv:2303.00216, 2023.

[5] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] I. Alonso, L. Riazuelo, L. Montesano, and A. Murillo. 3D-MiniNet Learning a 2D Representation from Point Clouds for Fast and Efficient 3D LIDAR Semantic Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 5(4):5432–5439, 2020.

[7] J. Amirian, J. Hayet, and J. Pettre. Social Ways: Learning Multi-Modal Distributions of Pedestrian Trajectories With GANs. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*, 2019.

[8] Y. Aoki, H. Goforth, A.S. Rangaprasad, and S. Lucey. PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet. In *Proc. of the*

*IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] 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.

[10] 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)*, 159:104287, 2023.

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

[12] S. Bai, J. Kolter, and V. Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. In *Proc. of the Int. Conf. on Learning Representations (ICLR) Workshop*, 2018.

[13] M. Bansal, A. Krizhevsky, and A. Ogale. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. In *Proc. of Robotics: Science and Systems (RSS)*, 2019.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] P. Biber and T. Duckett. Dynamic Maps for Long-Term Operation of Mobile Service Robots. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[20] P. Biber and W. Straßer. The Normal Distributions Transform: A New Approach to Laser Scan Matching. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[21] K. Brown, K. Driggs-Campbell, and M. Kochenderfer. A Taxonomy and Review of Algorithms for Modeling and Predicting Human Driver Behavior. *arXiv preprint*, arXiv:2006.08832, 2020.

[22] L. Caccia, H. Van Hoof, A. Courville, and J. Pineau. Deep Generative Modeling of Lidar Data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[23] 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.

[24] A. Carballo, J. Lambert, A. Monrroy, D. Wong, P. Narksri, Y. Kitsukawa, E. Takeuchi, S. Kato, and K. Takeda. LIBRE: The Multiple 3D LiDAR dataset. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2020.

[25] 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.

[26] S. Casas, W. Luo, and R. Urtasun. IntentNet: Learning to Predict Intention from Raw Sensor Data. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2018.

[27] K. Chen, R. Ge, H. Qiu, R. Ai-Rfou, C.R. Qi, X. Zhou, Z. Yang, S. Ettinger, P. Sun, Z. Leng, et al. Womd-Lidar: Raw Sensor Dataset Benchmark for Motion Forecasting. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[28] 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(4):6529–6536, 2021.

[29] 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.

[30] 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.

[31] 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.

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

[33] O. Chum and J. Matas. Matching with PROSAC - Progressive Sample Consensus. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[34] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *Proc. of the Advances in Neural Information Processing Systems Workshops*, 2014.

[35] 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.

[36] F. Crocetti, E. Bellocchio, A. Dionigi, S. Felicioni, G. Costante, M.L. Fravolini, and P. Valigi. ARD-VO: Agricultural Robot Data Set of Vineyards and Olive Groves. *Journal of Field Robotics (JFR)*, 40(6):1678–1696, 2023.

[37] H. Cui, V. Radosavljevic, F. Chou, T. Lin, T. Nguyen, T. Huang, J. Schneider, and N. Djuric. Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[38] S. Dai, L. Li, and Z. Li. Modeling Vehicle Interactions via Modified LSTM Models for Trajectory Prediction. *IEEE Access*, 7:38287–38296, 2019.

[39] 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.

[40] 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.

[41] P. Dellenbach, J. Deschaud, B. Jacquet, and F. Goulette. CT-ICP Real-Time Elastic LiDAR Odometry with Loop Closure. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[42] D. Deng and A. Zakhor. Temporal LiDAR Frame Prediction for Autonomous Driving. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2020.

[43] N. Deo and M. Trivedi. Convolutional Social Pooling for Vehicle Trajectory Prediction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*, 2018.

[44] N. Deo and M. Trivedi. Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2018.

[45] 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.

[46] F. Diehl, T. Brunner, M. Truong-Le, and A. Knoll. Graph Neural Networks for Modelling Traffic Participant Interaction. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2019.

[47] W. Ding, J. Chen, and S. Shen. Predicting Vehicle Behaviors Over An Extended Horizon Using Behavior Interaction Network. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[48] J.L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990.

[49] 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.

[50] H.W. F. Duerr, M. Pfaller and J. Beyerer. LiDAR-based Recurrent 3D Semantic Segmentation with Temporal Memory Alignment. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2020.

[51] H. Fan, H. Su, and L. Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[52] H. Fan, X. Yu, Y. Ding, Y. Yang, and M. Kankanhalli. PSTNet: Point Spatio-Temporal Convolution on Point Cloud Sequences. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2021.

[53] H. Fan and Y. Yang. PointRNN: Point Recurrent Neural Network for Moving Point Cloud Processing. *arXiv preprint*, arXiv:1910.08287, 2019.

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

[55] W. Förstner and B. Wrobel. *Photogrammetric Computer Vision – Statistics, Geometry, Orientation and Reconstruction.* Springer Verlag, 2016.

[56] A.H. Gebrehiwot, P. Vacek, D. Hurych, K. Zimmermann, P. Pérez, and T. Svoboda. Teachers in Concordance for Pseudo-Labeling of 3D Sequential Data. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):536–543, 2022.

[57] 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*, IV-1/W1:107–114, 2017.

[58] 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.

[59] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[60] 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.

[61] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

[62] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *Communications of the ACM*, 63(11):139–144, 2020.

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

[64] A. Graves and J. Schmidhuber. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2008.

[65] M. Grupp. evo: Python package for the evaluation of odometry and SLAM. `https://github.com/MichaelGrupp/evo`, 2017.

[66] T. Guadagnino, B. Mersch, I. Vizzo, S. Gupta, M. Malladi, L. Lobefaro, G. Doisy, and C. Stachniss. Kinematic-ICP: Enhancing LiDAR Odometry with Kinematic Constraints for Wheeled Mobile Robots Moving on Planar Surfaces. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2025.

[67] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[68] S. Gupta, T. Guadagnino, B. Mersch, I. Vizzo, and C. Stachniss. Effectively Detecting Loop Closures using Point Cloud Density Maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[69] J. Halkias and J. Colyar. Next Generation SIMulation Fact Sheet. Technical report, Federal Highway Administration, 2006.

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

[71] M. Henein, J. Zhang, R. Mahony, and V. Ila. Dynamic SLAM: The Need For Speed. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[72] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[73] 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.

[74] J. Hong, B. Sapp, and J. Philbin. Rules of the Road: Predicting Driving Behavior with a Convolutional Model of Semantic Interactions. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[75] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[76] I. Hroob, B. Mersch, C. Stachniss, and M. Hanheide. Generalizable Stable Points Segmentation for 3D LiDAR Scan-to-Map Long-Term Localization. *IEEE Robotics and Automation Letters (RA-L)*, 9(4):3546–3553, 2024.

[77] I. Hroob, S. Molina, R. Polvara, G. Cielniak, and M. Hanheide. Learned Long-Term Stability Scan Filtering for Robust Robot Localisation in Continuously Changing Environments. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2023.

[78] Y. Hu, W. Zhan, and M. Tomizuka. Probabilistic Prediction of Vehicle Semantic Intention and Motion. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2018.

[79] S. Huang, Z. Gojcic, J. Huang, A. Wieser, and K. Schindler. Dynamic 3D Scene Analysis by Point Cloud Accumulation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2022.

[80] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2015.

[81] S. Jang, M. Oh, B. Yu, I. Nahrendra, S. Lee, H. Lim, and H. Myung. TOSS: Real-time tracking and moving object segmentation for static scene mapping. In *Proc. of the Intl. Conf. on Robot Intelligence Technology and Applications (RiTA)*, 2023.

[82] J. Jeong, Y. Cho, Y.S. Shin, H. Roh, and A. Kim. Complex Urban Dataset with Multi-Level Sensors from Highly Diverse Urban Environments. *Intl. Journal of Robotics Research (IJRR)*, 38(6):642–657, 2019.

[83] M.I. Jordan. Serial Order: A Parallel Distributed Processing Approach. In *Neural-Network Models of Cognition*, volume 121, pages 471–495. North-Holland, 1997.

[84] M. Jung, W. Yang, D. Lee, H. Gil, G. Kim, and A. Kim. HeLiPR: Heterogeneous LiDAR Dataset for inter-LiDAR Place Recognition under Spatiotemporal Variations. *Intl. Journal of Robotics Research (IJRR)*, 12(43):1867––1883, 2024.

[85] O. Kayhan and J. van Gemert. On Translation Invariance in CNNs: Convolutional Layers can Exploit Absolute Spatial Location. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[86] B. Kim, C.M. Kan, J. Kim, S.H. Lee, C.C. Chung, and J.W. Choi. Probabilistic Vehicle Trajectory Prediction over Occupancy Grid Map via Recurrent Neural Network. In *Proc. of the IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, 2017.

[87] 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.

[88] 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.

[89] 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.

[90] 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.

[91] J. Kim, J. Woo, and Sunghoon. RVMOS: Range-View Moving Object Segmentation Leveraged by Semantic and Motion Features. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):8044–8051, 2022.

[92] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2015.

[93] K. Koide, J. Miura, and E. Menegatti. A Portable Three-dimensional LIDAR-based System for Long-term and Wide-area People Behavior Measurement. *Intl. Journal of Advanced Robotic Systems*, 16(2), 2019.

[94] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways

for Validation of Highly Automated Driving Systems. In *Proc. of the IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, 2018.

[95] L. Kreuzberg, I. Zulfikar, S. Mahadevan, F. Engelmann, and B. Leibe. 4D-StOP: Panoptic Segmentation of 4D LiDAR using Spatio-temporal Object Proposal Generation and Aggregation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2022.

[96] A. Kuefler, J. Morton, T.A. Wheeler, and M. Kochenderfer. Imitating Driver Behavior with Generative Adversarial Networks. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2017.

[97] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. A Navigation System for Robots Operating in Crowded Urban Environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[98] A. Laddha, S. Gautam, G.P. Meyer, C. Vallespi-Gonzalez, and C.K. Wellington. RV-FuseNet: Range View Based Fusion of Time-Series LiDAR Data for Joint 3D Object Detection and Motion Forecasting. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[99] S. Lefèvre, D. Vasquez, and C. Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *Journal of Robotics and Mechanical Engineering Research*, 1:1–14, 2014.

[100] E. Li, S. Casas, and R. Urtasun. MemorySeg: Online LiDAR Semantic Segmentation with a Latent Memory. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023.

[101] L. Li, B. Yang, M. Liang, W. Zeng, and M. Ren. End-to-end Contextual Perception and Prediction with Interaction Transformer. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[102] M. Liang, B. Yang, W. Zeng, Y. Chen, R. Hu, S. Casas, and R. Urtasun. PnPNet: End-to-End Perception and Prediction With Tracking in the Loop. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[103] H. Lim, S. Jang, B. Mersch, J. Behley, H. Myung, and C. Stachniss. HeLiMOS: A Dataset for Moving Object Segmentation in 3D Point Clouds From Heterogeneous LiDAR Sensors. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

[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] H. Lim, L. Nunes, B. Mersch, X. Chen, J. Behley, H. Myung, and C. Stachniss. ERASOR2: Instance-Aware Robust 3D Mapping of the Static World in Dynamic Scenes. In *Proc. of Robotics: Science and Systems (RSS)*, 2023.

[106] J. Lin and F. Zhang. Loam_livox: A Fast, Robust, High-Precision LiDAR Odometry and Mapping Package for LiDARs of Small FoV. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[107] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse Convolutional Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[108] J. Liu, C. Chang, J. Liu, X. Wu, L. Ma, and X. Qi. MarS3D: A Plug-and-Play Motion-Aware Model for Semantic Segmentation on Multi-Scan 3D Point Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[109] 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.

[110] X. Liu, M. Yan, and J. Bohg. MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[111] F. Lu, G. Chen, Z. Li, L. Zhang, Y. Liu, S. Qu, and A. Knoll. MoNet: Motion-based Point Cloud Prediction Network. *IEEE Trans. on Intelligent Transportation Systems (T-ITS)*, 23:13794–13804, 2022.

[112] F. Lu, G. Chen, S. Qu, Z. Li, Y. Liu, and A. Knoll. PointINet: Point Cloud Frame Interpolation Network. In *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, 2021.

[113] 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.

[114] W. Luo, B. Yang, and R. Urtasun. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[115] A. Maas, A. Hannun, and A. Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2013.

[116] 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.

[117] D. Maggio, M. Abate, J. Shi, C. Mario, and L. Carlone. Loc-NeRF: Monte Carlo Localization using Neural Radiance Fields. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[118] R. Marcuzzi, L. Nunes, L. Wiesmann, J. Behley, and C. Stachniss. Mask-Based Panoptic LiDAR Segmentation for Autonomous Driving. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1141–1148, 2023.

[119] R. Marcuzzi, L. Nunes, L. Wiesmann, I. Vizzo, J. Behley, and C. Stachniss. Contrastive Instance Association for 4D Panoptic Segmentation for Sequences of 3D LiDAR Scans. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[120] 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.

[121] 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.

[122] B. Mersch, T. Höllen, K. Zhao, C. Stachniss, and R. Roscher. Maneuver-based Trajectory Prediction for Self-driving Cars Using Spatio-temporal Convolutional Networks. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[123] B. Mersch, T. Guadagnino, X. Chen, Tiziano, I. Vizzo, J. Behley, and C. Stachniss. Building Volumetric Beliefs for Dynamic Environments Exploiting Map-Based Moving Object Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 8(8):5180–5187, 2023.

[124] G.P. Meyer, J. Charland, S. Pandey, A. Laddha, S. Gautam, C. Vallespi-Gonzalez, and C.K. Wellington. LaserFlow: Efficient and Probabilistic Object Detection and Motion Forecasting. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

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

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

[127] S. Mozaffari, O. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis. Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review. *IEEE Trans. on Intelligent Transportation Systems (T-ITS)*, 23:33–47, 2022.

[128] K. Museth, J. Lait, J. Johanson, J. Budsberg, R. Henderson, M. Alden, P. Cucka, D. Hill, and A. Pearce. OpenVDB: An Open-source Data Structure and Toolkit for High-resolution Volumes. In *ACM SIGGRAPH 2013 courses*, 2013.

[129] N. Naikal, J. Kua, G. Chen, and A. Zakhor. Image Augmented Laser Scan Matching for Indoor Dead Reckoning. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[130] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. In *Proc. of the SIGGRAPH Asia*, 2013.

[131] N. Nikhil and B. Morris. Convolutional Neural Network for Trajectory Prediction. In *Proc. of the Europ. Conf. on Computer Vision Workshops*, 2018.

[132] 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.

[133] L. Nunes, R. Marcuzzi, B. Mersch, J. Behley, and C. Stachniss. Scaling Diffusion Models to Real-World 3D LiDAR Scene Completion. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[134] D. Nuss, S. Reuter, M. Thom, T. Yuan, G. Krehl, M. Maile, A. Gern, and K. Dietmayer. A Random Finite Set Approach for Dynamic Occupancy Grid Maps with Real-Time Application. *Intl. Journal of Robotics Research (IJRR)*, 37(8):841–866, 2018.

[135] M. Oh, E. Jung, H. Lim, W. Song, S. Hu, E. Lee, J. Park, J. Kim, J. Lee, and H. Myung. TRAVEL: Traversable Ground and Above-Ground Object Segmentation Using Graph Representation of 3D LiDAR Scans. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7255–7262, 2022.

[136] 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.

[137] E. Palazzolo and C. Stachniss. Fast Image-Based Geometric Change Detection Given a 3D Model. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[138] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, and H. Zhao. Semanticposs: A Point Cloud Dataset with Large Quantity of Dynamic Instances. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2020.

[139] S. Park, B. Kim, C. Kang, C. Chung, and J. Choi. Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2018.

[140] 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.

[141] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep Contextualized Word Representations. In *Proc. of the Conf. of the North American Chapter of the Association for Computational Linguistics*, 2018.

[142] 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.

[143] R. Polvara, S. Molina, I. Hroob, A. Papadimitriou, K. Tsiolis, D. Giakoumis, S. Likothanassis, D. Tzovaras, G. Cielniak, and M. Hanheide. Bacchus Long-Term (BLT) Data Set: Acquisition of the Agricultural Multimodal BLT Data Set with Automated Robot Deployment. *Journal of Field Robotics (JFR)*, 40(8):2280–2298, 2023.

[144] 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.

[145] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing Icp Variants on Real-World Data Sets: Open-Source Library and Experimental Protocol. *Autonomous Robots*, 34:133–148, 2013.

[146] H. Porav, W. Maddern, and P. Newman. Adversarial Training for Adverse Conditions: Robust Metric Localisation Using Appearance Transfer. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[147] S.J.D. Prince. *Understanding Deep Learning*. MIT Press, 2023.

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

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

[150] L. Qingqing, Y. Xianjia, J.P. Queralta, and T. Westerlund. Multi-Modal LiDAR Dataset for Benchmarking General-Purpose Localization and Mapping Algorithms. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[151] J. Roewekaemper, C. Sprunk, G. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard. On the Position Accuracy of Mobile Robot Localization based on Particle Filters combined with Scan Matching. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[152] O. Ronneberger, P.Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proc. of the Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[153] 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.

[154] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, S.H. Rezatofighi, and S. Savarese. SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[155] 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.

[156] 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.

[157] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart. Dynablox: Real-time Detection of Diverse Dynamic Objects in Complex Environments. *IEEE Robotics and Automation Letters (RA-L)*, 8(10):6259–6266, 2023.

[158] L. Schmid, M. Abate, Y. Chang, and L. Carlone. Khronos: A Unified Approach for Spatio-Temporal Metric-Semantic SLAM in Dynamic Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2024.

[159] L. Schmid, J. Delmerico, J. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena. Panoptic Multi-Tsdfs: a Flexible Representation for Online Multi-Resolution Volumetric Mapping and Long-Term Dynamic Scene Consistency. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[160] J. Schulz, C. Hubmann, J. Lochner, and D. Burschka. Interaction-Aware Probabilistic Behavior Prediction in Urban Environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[161] P. Schütt, R. Rosu, and S. Behnke. Abstract Flow for Temporal Semantic Segmentation on the Permutohedral Lattice. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[162] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[163] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[164] 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.

[165] X. Shi, Z. Chen, H. Wang, D.Y. Yeung, W.K. Wong, and W.C. Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2015.

[166] H. Song, W. Ding, Y. Chen, S. Shen, M. Wang, and Q. Chen. PiP: Planning-informed Trajectory Prediction for Autonomous Driving. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[167] S. Song, H. Lim, A.J. Lee, and H. Myung. DynaVINS: A Visual-Inertial SLAM for Dynamic Environments. *IEEE Robotics and Automation Letters (RA-L)*, 7(4):11523–11530, 2022.

[168] Y. Song, Y. Tian, G. Wang, and M. Li. 2D LiDAR Map Prediction via Estimating Motion Flow with GRU. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[169] 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.

[170] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised Learning of Video Representations using LSTMs. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2015.

[171] C. Stachniss and W. Burgard. Mobile Robot Mapping and Localization in Non-Static Environments. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2005.

[172] J. Sun, Y. Dai, X. Zhang, J. Xu, R. Ai, W. Gu, 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.

[173] L. Sun, W. Zhan, C. Chan, and M. Tomizuka. Behavior Planning of Autonomous Cars with Social Perception. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2019.

[174] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Verlag, 2010.

[175] Y. Tang and R. Salakhutdinov. Multiple Futures Prediction. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2019.

[176] 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.

[177] 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.

[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] G.D. Tipaldi, D. Meyer-Delius, and W. Burgard. Lifelong localization in changing environments. *Intl. Journal of Robotics Research (IJRR)*, 32(14):1662–1678, 2013.

[181] I. Tishchenko, S. Lombardi, M.R. Oswald, and M. Pollefeys. Self-Supervised Learning of Non-Rigid Residual Flow and Ego-Motion. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2020.

[182] M. Toyungyernsub, M. Itkina, R. Senanayake, and M.J. Kochenderfer. Double-Prong ConvLSTM for Spatiotemporal Occupancy Prediction in Dynamic Environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[183] A.K. Ushani and R.M. Eustice. Feature Learning for Scene Flow Estimation from LIDAR. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2018.

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

[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] I. Vizzo, B. Mersch, R. Marcuzzi, L. Wiesmann, J. Behley, and C. Stachniss. Make It Dense: Self-Supervised Geometric Scan Completion of Sparse 3D Lidar Scans in Large Outdoor Environments. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):8534–8541, 2022.

[187] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss. VDBFusion: Flexible and Efficient TSDF Integration of Range Sensor Data. *Sensors*, 22(3):1296, 2022.

[188] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023.

[189] I. Vizzo, B. Mersch, L. Nunes, L. Wiesmann, T. Guadagnino, and C. Stachniss. Toward Reproducible Version-Controlled Perception Platforms: Embracing Simplicity in Autonomous Vehicle Dataset Acquisition. In *Worshop on Building Reliable Ratasets for Autonomous Vehicles, IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, 2023.

[190] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J.J. Leonard. Dynamic Pose Graph SLAM: Long-term Mapping in Low Dynamic Environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[191] D.Z. Wang, I. Posner, and P. Newman. What Could Move? Finding Cars, Pedestrians and Bicyclists in 3D Laser Data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012.

[192] S. Wang, J. Zhu, and R. Zhang. Meta-RangeSeg: LiDAR Sequence Semantic Segmentation Using Multiple Feature Aggregation. *IEEE Robotics and Automation Letters (RA-L)*, 7(4):9739–9746, 2022.

[193] Y. Wang, L. Jiang, M.H. Yang, L.J. Li, M. Long, and L. Fei-Fei. Eidetic 3D LSTM: A Model for Video Prediction and Beyond. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2019.

[194] L. Wellhausen, R. Dubé, A. Gawel, R. Siegwart, and C. Cadena. Reliable Real-time Change Detection and Mapping for 3D LiDARs. In *Proc. of the IEEE Intl. Sym. on Safety, Security, and Rescue Robotics (SSRR)*, 2017.

[195] 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.

[196] X. Weng, J. Wang, S. Levine, K. Kitani, and N. Rhinehart. Inverting the Pose Forecasting Pipeline with SPF2: Sequential Pointcloud Forecasting for Sequential Pose Forecasting. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2020.

[197] L. Wiesmann, T. Guadagnino, I. Vizzo, N. Zimmerman, Y. Pan, H. Kuang, J. Behley, and C. Stachniss. LocNDF: Neural Distance Field Mapping for Robot Localization. *IEEE Robotics and Automation Letters (RA-L)*, 8(8):4999–5006, 2023.

[198] P. Wu, S. Chen, and D.N. Metaxas. MotionNet: Joint Perception and Motion Prediction for Autonomous Driving Based on Bird's Eye View Maps. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[199] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In *Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[200] K. Wurm, C. Stachniss, and G. Grisetti. Bridging the Gap Between Feature- and Grid-based SLAM. *Journal on Robotics and Autonomous Systems (RAS)*, 58(2):140–148, 2010.

[201] P. Xiao, Z. Shao, S. Hao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang, et al. Pandaset: Advanced Sensor Suite Dataset for Autonomous Driving. In *Proc. of the IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, 2021.

[202] L. Xin, P. Wang, C. Chan, J. Chen, S. Li, and B. Cheng. Intention-aware Long Horizon Trajectory Prediction of Surrounding Vehicles using Dual LSTM Networks. In *Proc. of the IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, 2018.

[203] W. Xu and F. Zhang. FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):3317–3324, 2021.

[204] K. Yilmaz, J. Schult, A. Nekrasov, and B. Leibe. Mask4Former: Mask Transformer for 4D Panoptic Segmentation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[205] H. Yin, X. Xu, S. Lu, X. Chen, R. Xiong, S. Shen, C. Stachniss, and Y. Wang. A Survey on Global LiDAR Localization: Challenges, Advances and Open Problems. *Intl. Journal of Computer Vision (IJCV)*, 132:1–33, 2024.

[206] H. Yin, Y. Wang, L. Tang, X. Ding, S. Huang, and R. Xiong. 3D LiDAR Map Compression for Efficient Localization on Resource Constrained Vehicles. *IEEE Trans. on Intelligent Transportation Systems (T-ITS)*, 22(2):837–852, 2020.

[207] 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.

[208] 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.

[209] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2016.

[210] P. Yu, C. Guo, Y. Liu, and H. Zhang. Fusing Semantic Segmentation and Object Detection for Visual SLAM in Dynamic Scenes. In *Proc. of the 27th ACM Symposium on Virtual Reality Software and Technology*, 2021.

[211] M. Zeller, V. Sandhu, B. Mersch, J. Behley, M. Heidingsfeld, and C. Stachniss. Radar Velocity Transformer: Single-scan Moving Object Segmentation in Noisy Radar Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[212] M. Zeller, V. Sandhu, B. Mersch, J. Behley, M. Heidingsfeld, and C. Stachniss. Radar Instance Transformer: Reliable Moving Instance Segmentation in Sparse Radar Point Clouds. *IEEE Trans. on Robotics (TRO)*, 40:2357–2372, 2024.

[213] M. Zhai, X. Xuezhi, L. Ning, and X. Kong. Optical Flow and Scene Flow Estimation: A Survey. *Pattern Recognition*, 114:107861, 2021.

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

[215] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. Wu. Multi-Agent Tensor Fusion for Contextual Trajectory Prediction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[216] B. Zhou, J. Xie, Y. Pan, J. Wu, and C. Lu. MotionBEV: Attention-Aware Online LiDAR Moving Object Segmentation With Bird's Eye View Based Appearance and Motion Features. *IEEE Robotics and Automation Letters (RA-L)*, 8(12):8074–8081, 2023.

[217] M. Zhu, S. Han, H. Cai, S. Borse, M.G. Jadidi, and F.M. Porikli. 4D Panoptic Segmentation as Invariant and Equivariant Field Prediction. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023.

[218] N. Zimmerman, L. Wiesmann, T. Guadagnino, T. Läbe, J. Behley, and C. Stachniss. Robust Onboard Localization in Changing Environments Exploiting Text Spotting. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[219] A. Zyner, S. Worrall, and E. Nebot. Naturalistic Driver Intention and Path Prediction Using Recurrent Neural Networks. *IEEE Trans. on Intelligent Transportation Systems (T-ITS)*, 21(4):1584–1594, 2018.

# List of Figures

# List of Tables