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

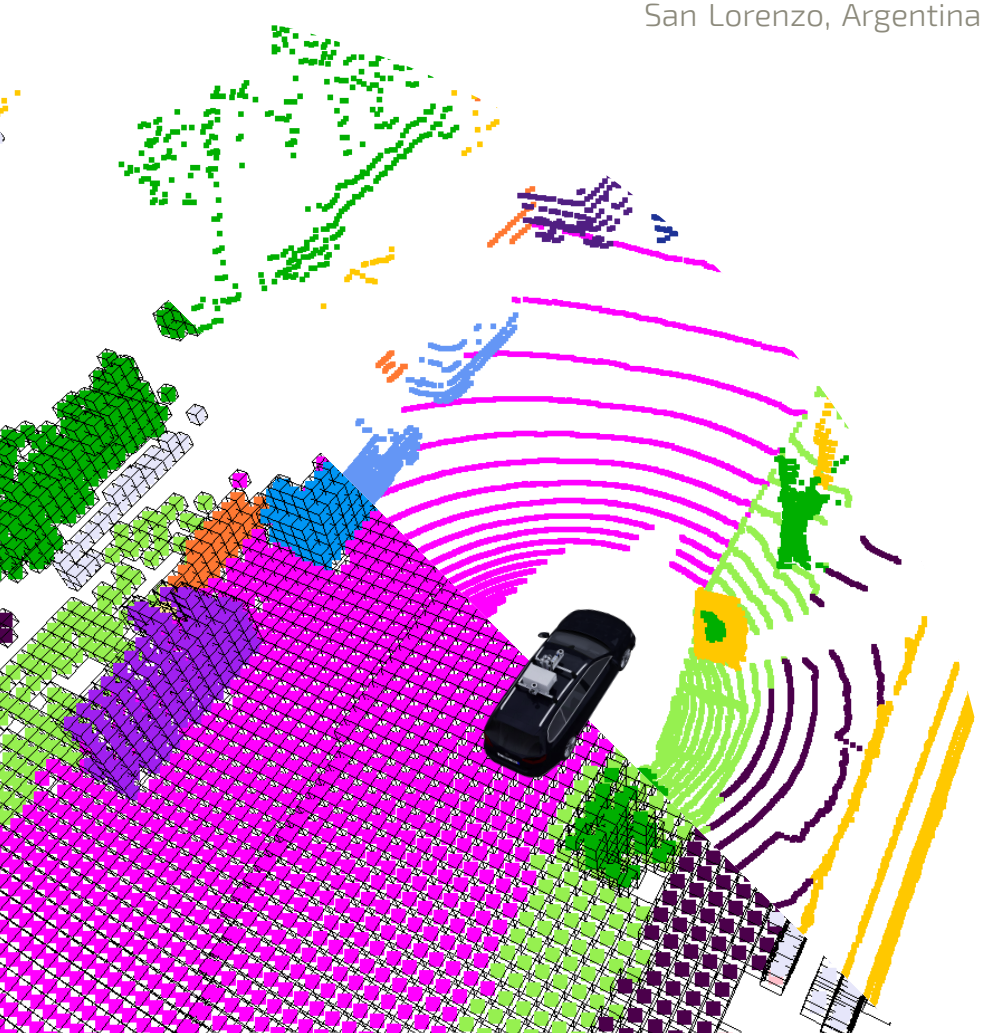
Spatio-Temporal Scene Understanding for Vehicles in Urban Environments

von

Rodrigo Nicolas Marcuzzi

aus

San Lorenzo, Argentina



Bonn 2026

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

Korreferent:
Prof. Dr. Bastian Leibe, RWTH Aachen University, Germany

Tag der mündlichen Prüfung: 16.01.2026

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

Abstract

Mobile robots are already present in our lives to support people in tasks they are not able or do not want to do. Driving machinery across large fields to help in food production, replacing human workers in repetitive tasks in production lines, and helping us to keep our homes clean are some of the applications nowadays. Robots do not get tired, do not get distracted, and can perform tedious tasks without complaining. One of the long-dreamed tasks that we have been wanting to automate is driving.

For autonomous driving, vehicles must perceive their environment accurately to make decisions and plan ahead in a safe way. This includes spatial awareness of the 3D surroundings and, at the same time, understanding their surroundings by knowing the semantic meaning of each element of the scene. For example, identifying the road and sidewalk to know where to drive and where not to. At the same time, it is important to recognize the surrounding buildings and traffic signs and identify other traffic participants such as cars and pedestrians. However, that is not enough to make decisions in dynamic environments. Such autonomous systems must also be able to track cars over time, estimate their velocity, and predict their future actions to make their own decisions, navigate, and avoid accidents. This means that the robot must have a 3D spatial representation of the scene, which also includes semantic meaning to recognize the surroundings and identify and track other traffic participants.

To understand their surroundings, robots are usually equipped with sensors such as cameras and LiDAR scanners. Each sensor has its own strengths and weaknesses, and they may provide complementary information. RGB cameras provide images with texture and color, and make it easier to understand details in the scene, but do not work in low-light conditions and lack 3D information. LiDAR sensors provide 3D information about the geometry of the scene, but with a lower resolution and lack real color information, which makes interpreting the surroundings more challenging.

The main contribution of this thesis is methods for spatio-temporal and semantic understanding of the surrounding scene. We propose methods for the individual sensor modalities, e.g., LiDAR and RGB cameras, to investigate the capabilities and challenges of each modality independently. In Part I, we work

with data from a rotating LiDAR sensor. Given that they already provide geometric information of the 3D surroundings, we focus on estimating the semantic meaning of each part of this 3D scene. Furthermore, we investigate how to enable these approaches to be scalable and learn all aspects of the task directly from data instead of relying on hand-picking parameters.

In Part II, we use images from conventional RGB cameras, which contain rich texture and color information. To achieve 3D semantic scene understanding using this data modality, we can leverage existing image segmentation models and therefore focus on estimating the geometry of the surrounding 3D scene combined with this semantic knowledge. We leverage all the available image information and use offline computer vision methods to generate labels. These labels can be used to train neural networks such that they can predict the geometry of the scene in an online fashion without relying on LiDAR data or human annotations.

We provide methods to tackle spatio-temporal semantic scene understanding tailored for individual data modalities by addressing the specific challenges and exploiting the strengths of the used sensor. Our proposed approaches have been evaluated on publicly available datasets and published in peer-reviewed journals and conferences. Furthermore, the implementations of our methods are open-source, as well as the generated data, to enable future research using them as a starting point.

Zusammenfassung

Mobile Roboter sind bereits in unserem Alltag präsent und unterstützen Menschen bei der Ausführung von Aufgaben, die sie entweder nicht selbst erledigen können oder nicht ausführen möchten. Beispiele hierfür sind das autonome Führen von Maschinen zur landwirtschaftlichen Nutzung, der Ersatz menschlicher Arbeitskraft bei repetitiven Tätigkeiten in Produktionslinien sowie die Unterstützung bei der Reinigung privater Haushalte. Roboter ermüden nicht, lassen sich nicht ablenken und führen monotone Arbeiten zuverlässig aus. Eine der seit Langem angestrebten und visionären Anwendungen der Robotik ist das autonome Fahren.

Um autonom agieren zu können, müssen Fahrzeuge ihre Umgebung präzise erfassen, fundierte Entscheidungen treffen und ihre Fahrweise vorausschauend sowie sicher planen. Dies setzt ein räumliches Verständnis der dreidimensionalen Umgebung sowie ein semantisches Verständnis der Szene voraus – das heißt, die Fähigkeit, einzelnen Bestandteilen der Umgebung eine Bedeutung zuzuweisen. Beispielsweise ist es erforderlich, Straßen und Gehwege zu erkennen, um zwischen befahrbaren und nicht befahrbaren Bereichen zu unterscheiden. Ebenso ist das Erkennen von Gebäuden, Verkehrsschildern sowie anderen Verkehrsteilnehmern wie Fahrzeugen und Fußgängern essenziell.

Für die Entscheidungsfindung in dynamischen Umgebungen genügt jedoch eine bloße Momentaufnahme nicht. Autonome Systeme müssen in der Lage sein, andere Verkehrsteilnehmer über die Zeit hinweg zu verfolgen, deren Geschwindigkeit abzuschätzen und deren zukünftige Bewegungen vorherzusagen. Nur so können sie eigene Entscheidungen treffen, sicher navigieren und potenzielle Kollisionen vermeiden. Daraus ergibt sich die Notwendigkeit einer semantisch angereicherten, dreidimensionalen Repräsentation der Szene, die sowohl die Umgebung als auch deren dynamische Elemente umfassend beschreibt. Zur Wahrnehmung der Umgebung kommen typischerweise unterschiedliche Sensortechnologien zum Einsatz, insbesondere Kameras und LiDAR-Scanner. Beide Sensormodalitäten besitzen spezifische Vor- und Nachteile und liefern komplementäre Informationen. Während RGB-Kameras Bilder mit hoher Farb- und Texturauflösung liefern,

jedoch keine Tiefeninformationen und nur eingeschränkt bei schlechten Lichtverhältnissen einsetzbar sind, bieten LiDAR-Sensoren präzise geometrische 3D-Informationen, allerdings mit geringerer Auflösung und ohne Farbinformationen, was die semantische Interpretation erschwert.

Der zentrale Beitrag dieser Dissertation liegt in der Entwicklung von Verfahren zum raum-zeitlichen und semantischen Verständnis der Umgebung. Es werden Methoden für einzelne Sensormodalitäten – insbesondere LiDAR und RGB-Kameras – vorgestellt, um deren jeweilige Potenziale und Herausforderungen isoliert zu untersuchen.

Teil I befasst sich mit der Verarbeitung von Daten rotierender LiDAR-Sensoren. Da diese bereits geometrische Informationen der 3D-Umgebung bereitstellen, liegt der Fokus auf der Schätzung semantischer Bedeutungen innerhalb dieser Repräsentationen. Zudem wird untersucht, wie diese Ansätze skalierbar gestaltet werden können, indem sie sämtliche Aufgabenbestandteile direkt aus den Daten lernen, ohne auf manuell festgelegte Parameter angewiesen zu sein.

Teil II widmet sich der Nutzung von RGB-Kamerabildern, die reichhaltige Textur- und Farbinformationen enthalten. Um ein semantisches Verständnis der 3D-Szene auf Basis dieser Modalität zu erreichen, werden bestehende Verfahren zur Bildsegmentierung herangezogen und mit Methoden zur Tiefenschätzung kombiniert. Dabei kommen computer vision-basierte Verfahren zur automatischen Labelgenerierung zum Einsatz, wodurch das Training von Neuronalen Netzen für die geometrische Prädiktion ganz ohne Einsatz von LiDAR oder manuell annotatierten Daten ermöglicht wird.

Die entwickelten Methoden zur raum-zeitlichen semantischen Szenenanalyse wurden gezielt an die jeweilige Sensormodalität angepasst und berücksichtigen deren spezifische Herausforderungen. Die vorgeschlagenen Verfahren wurden auf öffentlich zugänglichen Datensätzen evaluiert und im Rahmen von Fachzeitschriften und Konferenzen mit Peer-Review veröffentlicht. Darüber hinaus stehen sowohl die Implementierungen der entwickelten Methoden als auch die erzeugten Datensätze als frei zugänglich zur Verfügung, um zukünftige Forschung zu unterstützen und weiterführende Entwicklungen zu ermöglichen.

Acknowledgements

When I first thought about pursuing a Ph.D. I was not sure about what to expect. I knew I would have to "do research" but I had no idea what I would do. Nevertheless, full of fears and uncertainties, I moved to another continent where I could not even communicate in the local language to follow my dream. Now, almost five years after that decision, I can say that I am happy I had the courage. I indeed spent the last years researching, which meant reading, coding, teaching, discussing, exploring, and more. I had an amazing time, which was only possible thanks to a lot of people along the way, and I would like to thank each one of them.

First and foremost, I want to express my deepest gratitude to my supervisor, Cyrill Stachniss. For believing in me and giving me the possibility to join his team of awesome people, and guiding me throughout this path. His support and help since day one were invaluable and allowed me to make mistakes and grow enormously. Furthermore, he shaped a work environment in which it was enjoyable to work and collaborate with colleagues.

Next, I would like to thank Jens Behley, who was always available to discuss problems and new ideas and to help me give direction to my research. He "provided knowledge and support" consistently throughout the years and supported me in the many different stages to make an idea come to reality, and always challenged me by being the very much needed reviewer 2.

I would like to thank Mario Munich, who welcomed me in a new country and trusted me, giving me the possibility to start my robotics journey.

During my Ph.D. I had the pleasure of sitting facing my dear colleague and friend, Lucas, my second opinion in big and small decisions, and support in all our projects. I was also lucky enough to work alongside my friend Louis, with whom I had many fruitful discussions and who helped me crack the latest details in my endeavours with his technical clarity and humor. Along with them, I worked with Matteo, Elias, Federico, and Gianmarco, who, apart from being patient with me while learning Italian, contributed to creating an amazing working environment to contribute to and create a beautiful friendship. We shared great times in the lab and outside of it, including games, food, and mainly good laughs. Special thanks to the big cooks for sharing their delicious meals, time, and laughs with

me, even though I was never cooking and always stealing food.

Besides them, I thank my colleagues for the shared moments, discussions, and for making working in the lab a great experience. Thanks to Linn, Stary, Luca, Meher, Yue, Rhiney, Haofei, Benedikt, Juluis, Niklas, Jan, Liren, Gupta, and Thomas.

Arriving in a new country and a new culture was not easy, and I would like to thank Claus and Eve for welcoming me with open arms and helping me in my new world. A special thanks goes to Claus for the infinite whiteboard talks, advice, adventures, and slackline sessions. It was great to already have a friend in Bonn and to share this path with him.

I would also like to thank Nacho, who supported me since my decision to apply to the Ph.D. position and was always available to discuss the best practices to be a scientist, and who taught me a lot about work and life. Apart from moving to a new country, arriving in Germany during a pandemic made it especially tricky, but it was made much easier and cosier thanks to Peter, Kay, and Hannes, who received me like family and taught me about the German ways and the uses of Sriracha.

In Bonn, I had the pleasure of meeting my friend and teacher, Lucho, who showed how me to appreciate different kinds of knowledge and abilities, and with whom I spent countless hours talking and enjoying nature most of the time close to a slackline. I am also very happy to have met Mari, who I would like to thank for the beautiful friendship we developed, the talks, the events, and support during this period of changes and uncertainties. In the past years, I also had the pleasure to share lots of talks and adventures with my friend Lucas Casuccio, to whom I am very grateful for his support in the PhD process and for the amazing relationship we formed, which I cherish.

During these years, I was also lucky and privileged to meet Jami, my partner, my team player, my love. Someone who supported me with every small thing and who was always there to hear me out and give me energy. I am very grateful for having found you, for helping me grow, and for sharing this path with you.

I also want to thank my family for supporting me in my life decisions and always being there to share any news or visit. Thanks, Gaston, Pablo, Neri, and San. My biggest and heartfelt thank you goes to my parents, Marcos and Patricia, without whom none of this would have been possible, who supported me and taught me since day one, and were always by my side to encourage me. Gracias de corazón.

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.

Contents

Acronyms

BEV bird's eye view

CNN convolutional neural network

GPU graphics processing unit

LiDAR light detection and ranging

MLP multi layer perceptron

ReLU rectified linear unit

SFM structure from motion

SLAM simultaneous localization and mapping

Chapter 1

Introduction

Autonomous robots are becoming increasingly present in our daily lives. These systems have the potential to support or even replace humans in many tasks that are too complex, repetitive, boring, or dangerous. They are already deployed in various industries, offering automation of repetitive tasks without the drawbacks of human fatigue, distraction, or inconsistency. In agriculture, robots can drive across large fields to plant, weed, and harvest crops. In warehouses, they assist with transporting goods and performing inventory monitoring. In the manufacturing industry, robots are used on production lines and provide a way of automating processes with high precision and efficiency without breaks. At home, autonomous vacuum cleaners support us in maintaining our home clean while autonomous lawnmowers handle yard maintenance at low cost. One of the long-standing dreams is to use autonomous cars to replace humans in driving.

Autonomous vehicles bring with them many promises and offer a view of how a future without human drivers might look. Such systems have the potential to make roads safer by preventing traffic accidents, most of which are caused by human error, distractions, poor decision-making, or fatigue. Unlike humans, autonomous cars do not lose focus, do not tire, and can process information and react more quickly. Autonomous trucks could improve efficiency in logistics by operating continuously without the need to rest, even when driving for long distances on quiet routes. Moreover, the overall traffic flow could benefit from more intelligent, coordinated driving decisions, with a reduction in traffic jams. Scenarios like these, as also illustrated in Figure 1.1, could potentially be improved by autonomous driving cars.

To safely navigate the environment, autonomous robots need to gather 3D information to perceive and understand their surroundings. This includes identifying road elements and other road users, and interpreting and anticipating their intentions to make informed decisions accordingly. Scene understanding

(a) Road safety

(b) Logistics

(c) Traffic flow

Figure 1.1: Examples of how autonomous vehicles can (a) make roads safer by preventing traffic accidents, (b) improve the efficiency in logistics by operating without breaks, even in long and calm routes, and (c) prevent traffic jams and improve traffic flow by making more intelligent driving decisions. Images from Wikimedia Commons.

involves detecting, among other things, drivable areas, non-driveable areas such as sidewalks, parking spaces, and overall different parts of the scene, such as vegetation and the surrounding buildings. Importantly, an autonomous vehicle needs to identify other traffic participants, such as cars, pedestrians, and cyclists, at an instance level and track them over time to estimate their trajectories and predict their future actions. Perception is a critical component of autonomy, as it serves as the foundation for planning and control. The safety and effectiveness of decision-making directly depend on the quality of the perception system to understand the scene. These systems must function in a wide variety of dynamic scenarios, like rural areas where the roads are irregular, cities with dense traffic and pedestrians, and high-speed highways. Furthermore, the operation of autonomous vehicles must be guaranteed across diverse weather and lighting conditions.

To achieve such an understanding of the surrounding scene, autonomous vehi-

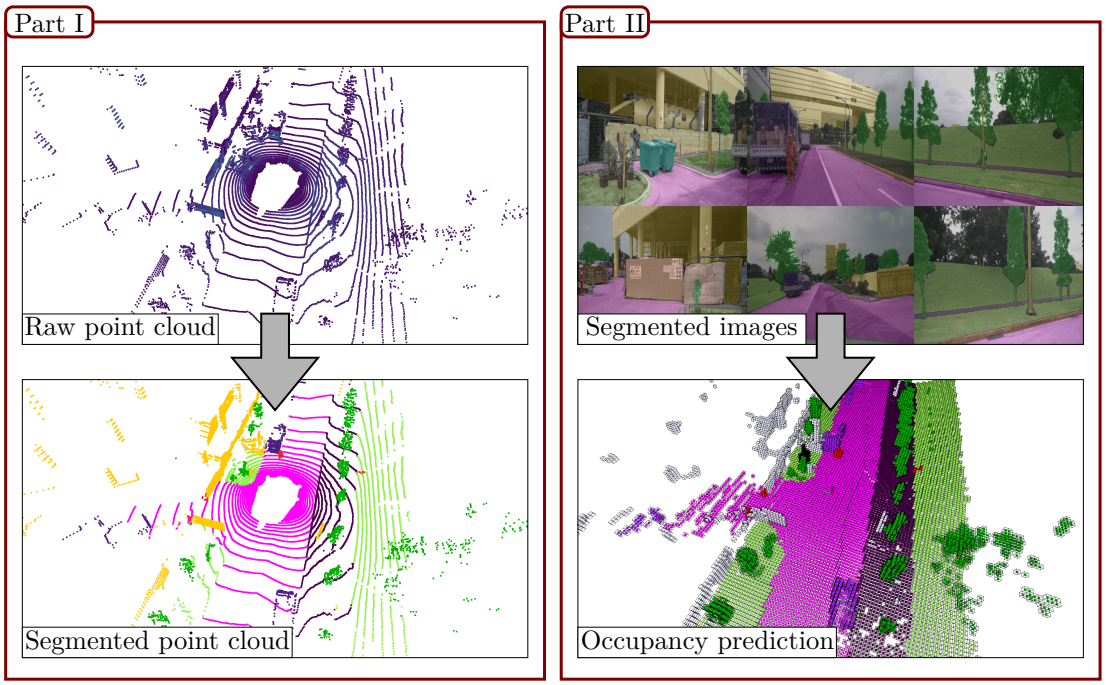


Figure 1.2: LiDAR point clouds provide a 3D representation of the scene and therefore in Part I we focus on semantically segmenting the scene. Segmented RGB images provide a semantic 2D representation of the scene, and consequently, in Part II we place our focus on the spatial representation of the scene by reconstructing the 3D surroundings via occupancy prediction.

cles are equipped with multiple complementary sensors such as RGB cameras and light detection and ranging (LiDAR) sensors. Sensor redundancy enhances safety by ensuring continuous operation if one sensor fails, while the complementary nature of the obtained information improves scene understanding. For instance, RGB cameras are cheap and offer high-resolution images with rich texture and color. This detailed information makes it easier to semantically segment the scene. However, camera data is inherently 2D and lacks direct depth information to perceive the 3D surroundings, and struggle in low-light conditions. LiDAR sensors, on the other hand, provide accurate 3D geometric information as distance to surfaces but typically have a smaller spatial resolution and do not provide real color, making object recognition more challenging. Thus, each sensor brings unique strengths and limitations. Studying these sensors individually allows for a deeper understanding of how each contributes to semantic scene understanding. This approach enables us to investigate the capabilities and limitations of each modality for scene understanding without relying on the availability of a full sensor suite and the added complexity of sensor fusion. Moreover, some autonomous systems might rely on a reduced set of sensors and be limited to a single data modality, such as RGB cameras or LiDAR, due to cost, hardware constraints, or specific use cases. Understanding what can be achieved with only

LiDAR or only RGB cameras is therefore crucial. In this thesis, we investigate spatio-temporal and semantic scene understanding using LiDAR and camera data separately. This understanding of the environment involves not only capturing the 3D structure of the surroundings but also interpreting their semantic content and temporal evolution, which enables the autonomous system to comprehend its context and supports successful planning and navigation. Semantic segmentation contributes to this interpretation by assigning semantic meaning to the elements of the scene. This allows the system to recognize, for example, drivable and non-driveable areas, buildings, vegetation, vehicles, pedestrians, and more. However, it is also crucial for the vehicle to identify individual traffic participants such as cars, buses, pedestrians, and cyclists, which is key to making informed decisions about future actions. By combining semantic and instance-level information, panoptic segmentation offers a more unified understanding of the 3D scene, enabling the robot to interpret the surroundings and differentiate between individual instances. Furthermore, due to the dynamic nature of urban environments, an autonomous vehicle must also be capable of tracking each agent over time. The inclusion of this temporal knowledge enables the estimation of their trajectories and the prediction of future behaviour. In this context, 4D panoptic segmentation integrates all this information, enabling a comprehensive perception of the environment and supporting autonomous decision-making.

To achieve this level of scene understanding, it is essential to leverage the strengths of different sensing modalities. By studying each modality in isolation, we develop methods tailored to the specific strengths and challenges of each sensor. This separation serves as a foundation for future multi-sensor fusion research.

In Part I, we improve LiDAR semantic scene understanding through our methods. Since this data modality intrinsically provides a 3D representation of the scene, we focus on semantically segmenting the scene as shown in Figure 1.2 (left) and on temporally tracking individual traffic participants.

In Part II, we tackle semantic scene understanding using only RGB cameras as sensors. Given that this data modality provides rich texture and color information which facilitates semantic segmentation, we use an existing segmentation model and shift our focus to the 3D spatial representation of the scene. Specifically, we reconstruct the surroundings of the autonomous system by dividing the 3D space into voxels and predicting their occupancy and semantic class, as shown in Figure 1.2 (right).

1.1 Main Contributions

In this thesis, we investigate how to achieve spatio-temporal and semantic scene understanding, which is a cornerstone in enabling autonomous vehicles to safely

navigate complex environments. To analyze the challenges of individual data modalities, we independently study this task relying only on LiDAR data in Part I and RGB cameras in Part II. This section summarizes the main contributions of the individual parts of this thesis.

The first contribution presented in Chapter 4, is a method to extend existing 3D panoptic segmentation models to perform 4D panoptic segmentation and obtain time-consistent instance IDs via the tracking-by-detection paradigm. We use a pretrained panoptic model to individually segment each scan and associate instances over time using appearance and motion information. To obtain an appearance model of each instance, we propose a contrastive aggregation network to compute instance-wise feature vectors that are similar over time and perform the association between previous and current instances via feature similarity. We train this network using contrastive learning, where the positive samples are the same instance in LiDAR scans from consecutive timesteps, and the negatives are all the other instances in a batch consisting of multiple LiDAR scans from different timesteps. The novelty of this method is our contrastive aggregation network to obtain temporally consistent appearance information for each instance. This is achieved by using contrastive learning, commonly used for representation learning, between the computed instance feature and the many instance features that we provide as positive and negative samples. The outcome is a model that segments the surrounding LiDAR scan and is able to identify and track individual traffic participants over time.

Since most methods for 4D panoptic segmentation rely on post-processing steps such as clustering and instance associations, and rely on hand-tuning parameters, it is not possible to train the model end-to-end and learn segmentation and association from data. As a first step in that direction, we present our second contribution in Chapter 5, which is a method to perform 3D panoptic segmentation in an end-to-end manner without the need for post-processing steps like clustering and without the need for hand-tuning parameters. We reformulate the panoptic segmentation as the prediction of a set of non-overlapping binary masks along with their semantic classes. We rely on a sparse convolutional backbone as a feature extractor, a transformer decoder, and a set of learnable queries that we employ as mask proposals, from which we obtain the binary masks and the semantic classes. These mask proposals are refined via consecutive decoder layers, which consist of cross-attention to allow the queries to interact with the point-wise features, and self-attention to allow the queries to attend to each other. The end-to-end training process allows the model to learn how to segment instances of different shapes and sizes without the need to hand-tune any parameters. This makes the model more scalable since it can potentially learn from more available data. This leads to a more robust segmentation by including data from different

environments in different locations across the world. Furthermore, the model could potentially segment any type of LiDAR scan if multiple LiDAR sensors are used in the collection of the training data.

The third contribution presented in Chapter 6 is a method to perform 4D panoptic segmentation in an end-to-end manner without the need for any post-processing steps like clustering or associations and without the hand-tuning of parameters usually needed to fuse appearance and motion information. We extend the method presented in Chapter 5 by dividing the learnable queries into two sets: detection and tracking queries. Our detection queries segment newly appearing instances and stuff classes, while the tracking queries segment the same instance over time to implicitly perform tracking. We continuously update the tracking queries using the last detection of the instance to adapt to appearance changes and successfully track the instance over time. Since the LiDAR scans are recorded sequentially, after we detect an instance, we can leverage information about its position to guide its segmentation in the next timestep. To include this spatial prior information about the tracked instances, we further propose position-aware mask attention, a modified version of the cross-attention mechanism. We keep track of the instance's position and size and use it to compute a Gaussian-like kernel to modify the attention weights. This modification helps in the segmentation by making the query focus on the area around the position of the tracked instance. These modifications allow our model to perform 4D panoptic segmentation without the need for post-processing steps and optimize jointly for segmentation and association, learning how to combine different cues directly from the data, and making the model more scalable. As a result, our approach is more scalable and can potentially learn from data from different LiDAR sensors in different locations around the world.

In Part II, we tackle 3D semantic scene understanding of the surrounding scene using only RGB cameras as sensors. While the estimation of semantics can be handled by standard deep learning methods, the major problem lies in the fact that RGB cameras lack depth information. Therefore, we focus on estimating the structure of the surrounding 3D scene using images from a single timestep in combination with a segmentation method. The fourth contribution presented in Chapter 7 is a method to generate sparse occupancy pseudo labels relying solely on RGB images. We can use these labels to supervise networks for 3D semantic occupancy prediction without the need for a LiDAR sensor or manual labeling. We propose to leverage all the available images from the training set and use bundle adjustment to align the images of each scene and obtain camera poses. We use this information to generate a point cloud and project it into each camera to obtain sparse scale-aware depth images to replace the usual depth supervision that assumes the availability of a LiDAR sensor. To supervise our method directly

with occupancy values, we voxelize the environment and perform ray casting using the filtered depth images to set each voxel as occupied or free, and obtain sparse occupancy pseudo labels. To predict not only the occupancy but also the semantics of each voxel in the 3D space, we leverage a visual foundation model to semantically segment the RGB images and combine them with the depth images to generate sparse semantic occupancy pseudo labels as explicit supervision. This allows us to train our model without relying on LiDAR scans or occupancy ground truth labels.

The fifth contribution presented in Chapter 8 is a method to generate pseudo labels relying on RGB images only, which we can use to supervise networks for 3D panoptic occupancy prediction. We build on top of the work presented in Chapter 7 and use only images and bundle adjustment to obtain scale-aware depth images. Additionally, we use a visual foundation model to obtain semantic and instance segmentation of the RGB images. We perform ray casting and generate pseudo labels containing not only the semantics but also the instances of the scene. Given the static scene assumption made during bundle adjustment, the pseudo labels do not include dynamic objects. We leverage a 3D foundation model to obtain dense depth predictions and use them to detect dynamic objects and add them to the pseudo labels. This allows us to train a model using only RGB images and predict the geometry, semantics, and instances of the scene, including the moving objects.

Overall, this thesis presents five contributions to spatio-temporal and semantic scene understanding for autonomous vehicles. In Part I, we use only LiDAR data and focus on the segmentation of the surrounding scene and the instance tracking over time. Our goal is to remove post-processing steps like clustering and association and train our model end-to-end to learn segmentation and association directly from the data, making it more scalable. In Part II, we use only RGB images and concentrate on the reconstruction of the surrounding scene in the form of occupancy prediction. We propose to only use RGB images during training and leverage a visual foundation model and structure-from-motion to generate occupancy pseudo labels to predict the geometry of the scene alongside the semantics and instances. Moreover, we use a 3D foundation model to obtain dense depth predictions and account for dynamic objects in the pseudo labels, which allows us to predict the occupancy of these moving objects.

We published the implementation of the presented methods and the generated pseudo labels to facilitate future research.

1.2 Publications

Most parts of this thesis have been published in the following peer-reviewed conference and journal articles:

- R. Marcuzzi, L. Nunes, L. Wiesmann, I. Vizzo, J. Behley, and C. Stachniss. Contrastive Instance Association for 4D Panoptic Segmentation using Sequences of 3D LiDAR Scans. In *IEEE Robotics and Automation Letters (RA-L)*, 7(2):1550–1557, 2022. DOI: 10.1109/LRA.2022.3140439

- R. Marcuzzi, L. Nunes, L. Wiesmann, J. Behley, and C. Stachniss. Mask-Based Panoptic LiDAR Segmentation for Autonomous Driving. In *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1141–-1148, 2023. DOI: 10.1109/LRA.2023.3236568

- R. Marcuzzi, L. Nunes, L. Wiesmann, E. Marks, J. Behley, and C. Stachniss. Mask4D: End-to-End Mask-Based 4D Panoptic Segmentation for LiDAR Sequences. In *IEEE Robotics and Automation Letters (RA-L)*, 8,(11):7487–7494, 2023. DOI: 10.1109/LRA.2023.3320020

- R. Marcuzzi, L. Nunes, E. Marks, L. Wiesmann, T. Läbe, J. Behley, and C. Stachniss. SfmOcc: Vision-Based 3D Semantic Occupancy Prediction in Urban Environments. In *IEEE Robotics and Automation Letters (RA-L)*, 10(5):5074–5081, 2025. DOI: 10.1109/LRA.2025.3557227

- R. Marcuzzi, L. Nunes, E. Marks, X. Zhong, J. Behley, and C. Stachniss. SfmPanOcc: Vision-Based Panoptic Occupancy Prediction in Urban Environments. (under review)

1.3 Collaborations

In addition to the aforementioned articles, I was also involved as a co-author in the following peer-reviewed conference and journal publications which are not part of the thesis:

- 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

- L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination. In *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2116–2123, 2022. DOI: 10.1109/LRA.2022.3142440

- 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. In *IEEE Robotics and Automation Letters (RA-L)*, 7(3):6107–6114, 2022. DOI: 10.1109/LRA.2022.3166544

- L. Wiesmann, R. Marcuzzi, C. Stachniss, and J. Behley. Retriever: Point Cloud Retrieval in Compressed 3D Maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022. DOI: 10.1109/ICRA46639.2022.9811785

- L. Nunes, X. Chen, R. Marcuzzi, A. Osep, L. Leal-Taixé, C. Stachniss, and J. Behley. Unsupervised Class-Agnostic Instance Segmentation of 3D LiDAR Data for Autonomous Vehicles. In *IEEE Robotics and Automation Letters (RA-L)*, 7(4):8713–8720, 2022. DOI: 10.1109/LRA.2022.3187872

- L. Nunes, L. Wiesmann, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. Temporal Consistent 3D LiDAR Representation Learning for Semantic Perception in Autonomous Driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. DOI: 10.1109/CVPR52729.2023.00505

- E. Marks, M. Sodano, F. Magistri, L. Wiesmann, D. Desai, R. Marcuzzi, J. Behley, and C. Stachniss. High Precision Leaf Instance Segmentation in Point Clouds Obtained Under Real Field Conditions. In *IEEE Robotics and Automation Letters (RA-L)*, 8,(8):4791–4798, 2023. DOI: 10.1109/LRA.2023.3288383

- F. Magistri, R. Marcuzzi, E. Marks, M. Sodano, J. Behley, and C. Stachniss. Efficient and Accurate Transformer-Based 3D Shape Completion and

Reconstruction of Fruits for Agricultural Robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024. DOI: 10.1109/ICRA57147.2024.10611717

- 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

- L. Nunes, R. Marcuzzi, J. Behley, and C. Stachniss. Towards Generating Realistic 3D Semantic Training Data for Autonomous Driving. arXiv Preprint, vol. arXiv:2503.21449, 2025. DOI: 10.48550/arXiv.2503.21449

- M. Sodano, F. Magistri, E. Marks, F. Hosn, A. Zurbayev, R. Marcuzzi, M. V. R. Malladi, J. Behley, and C. Stachniss. 3D Hierarchical Panoptic Segmentation in Real Orchard Environments Across Different Sensors. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2025. DOI:10.1109/IROS60139.2025.11246899

- E. Marks, L. Nunes, F. Magistri, M. Sodano, R. Marcuzzi, L. Zimmermann, J. Behley, and C. Stachniss. Tree Skeletonization from 3D Point Clouds by Denoising Diffusion. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2025.

1.4 Open Source Contributions

During my doctorate, I published all proposed approaches as open-source implementations to support the reproducibility of results.

- **CA-Net**: Contrastive association for 4D panoptic segmentation `https://github.com/PRBonn/contrastive_association` (Chapter 4)

- **MaskPLS**: Mask-based 3D panoptic lidar segmentation `https://github.com/PRBonn/MaskPLS` (Chapter 5)

- **Mask4D**: Mask-based 4D panoptic lidar segmentation `https://github.com/PRBonn/Mask4D/` (Chapter 6)

- **SfmOcc**: Vision-based 3D semantic occupancy prediction `https://github.com/PRBonn/SfmOcc/` (Chapter 7)

- **SfmPanOcc**: Vision-based 3D panoptic occupancy prediction `https://github.com/PRBonn/SfmPanOcc/` (Chapter 8)

Chapter 2

Basic Techniques: Neural Networks for Point Clouds

To achieve scene understanding based on 3D point clouds, the first step is usually to extract a feature vector for each point, which is generally done using neural networks. In this section, we focus on the most common neural network architectures to extract point features and process light detection and ranging (LiDAR) data.

2.1 3D Convolutional Neural Networks

The most common architecture to extract features from input data are convolutional neural network (CNN)s. Different from images, which consist of a regular dense grid of pixels, a 3D point cloud $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N\}$ is an irregular and unorganized set of 3D points $\boldsymbol{p}_i \in \mathbb{R}^3$ that do not follow any grid-like structure. To be able to apply convolutional networks to such a type of data, one option is to treat the environment as a 3D dense voxel grid, which enables the definition of neighborhoods and convolutions. Point coordinates are then discretized using a given spatial voxel size $\Delta s \in \mathbb{R}$. For each point, we obtain a discrete coordinate $\boldsymbol{u} \in \mathbb{Z}^3$ as

$$\boldsymbol{u} = \left[\left\lfloor \frac{x}{\Delta s} \right\rfloor, \left\lfloor \frac{y}{\Delta s} \right\rfloor, \left\lfloor \frac{z}{\Delta s} \right\rfloor \right]^\top, \tag{2.1}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation.

Based on the 3D voxel grid, we can define a 3D convolution at a discrete coordinate $\boldsymbol{u} \in \mathbb{Z}^3$. This means applying a convolution kernel $\mathsf{W} \in \mathbb{R}^{K^3 \times N_{\text{out}} \times N_{\text{in}}}$, where K is the kernel size, on the input features with size N_{in} in the local neighborhood defined by the kernel. We can represent the local neighborhood of each

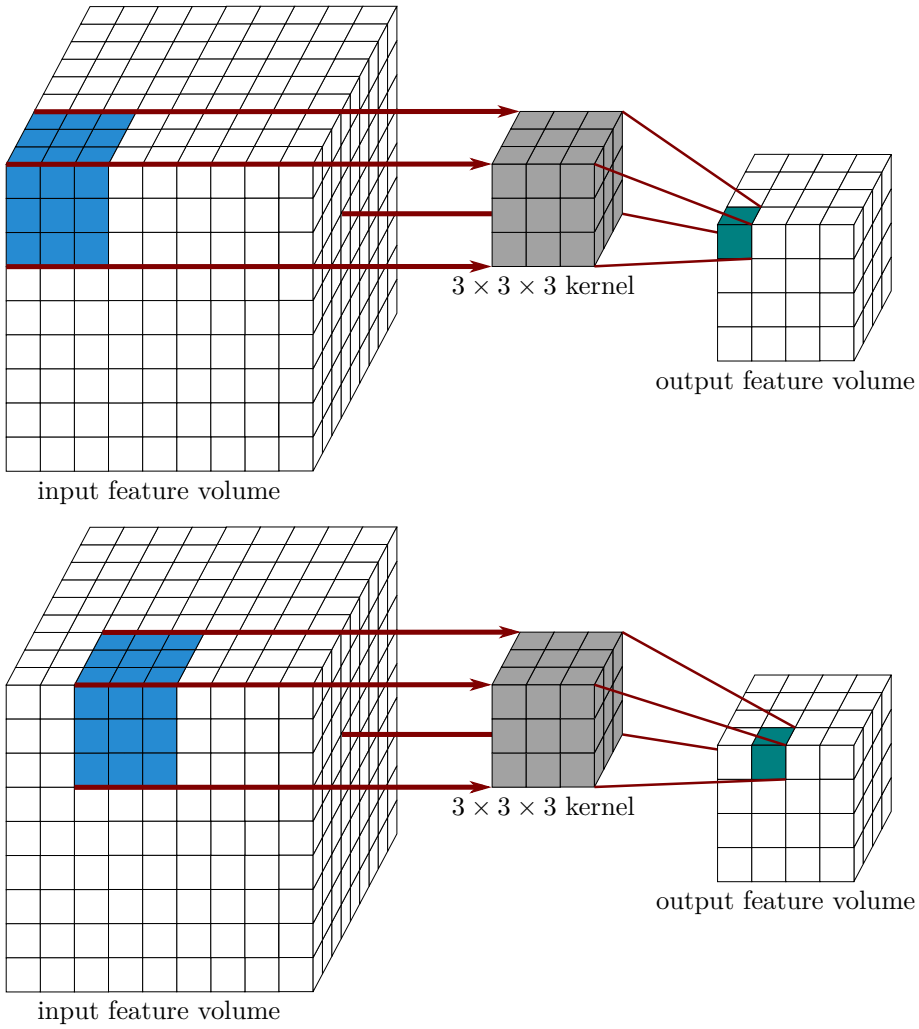


Figure 2.1: Example of the application of a 3D convolutional kernel (gray) at different locations (blue) of an input feature volume to obtain output features at different locations (green).

location $\boldsymbol{u}$ as a set of offsets $\mathcal{V}^3(K)$ and split the kernel into K^3 matrices, obtaining a matrix $\boldsymbol{W}_{\boldsymbol{i}} \in \mathbb{R}^{N_{\text{out}} \times N_{\text{in}}}$ for each offset $\boldsymbol{i} \in \mathbb{Z}^3$. This way, we express the convolution as the sum over the products of each kernel matrix $\boldsymbol{W}_{\boldsymbol{i}}$ with input feature $\boldsymbol{f}^{\text{in}}_{\boldsymbol{u}+\boldsymbol{i}} \in \mathbb{R}^{N_{\text{in}}}$ at location $\boldsymbol{u} + \boldsymbol{i}$. The 3-dimensional convolution at location $\boldsymbol{u} \in \mathbb{Z}^3$ is defined as

$$\boldsymbol{f}^{\text{out}}_{\boldsymbol{u}} = \sum_{\boldsymbol{i} \in \mathcal{V}^3(K)} \boldsymbol{W}_{\boldsymbol{i}} \boldsymbol{f}^{\text{in}}_{\boldsymbol{u}+\boldsymbol{i}}, \tag{2.2}$$

where $\mathcal{V}^3(K)$ is the set of offsets the kernel is operating on with $|\mathcal{V}^3(K)| = K^3$. In Figure 2.1, we illustrate an example of a 3D convolution. The $3 \times 3 \times 3$ kernel (gray) is applied at different locations (blue) of the input feature volume to obtain the output features at each location (green).

2.2 Sparse Convolutions

When representing the environment using a 3D voxel grid, the number of voxels grows cubically with the spatial dimension, and therefore, the 3D convolutions do not scale well for large environments. The point clouds obtained from LiDAR sensors are spatially sparse, resulting in a majority of empty voxels where the result of the convolutions is zero. To speed up inference and minimize memory footprint, Liu et al. [87] propose to compress a neural network by pruning redundant parameters, but the convolution still operates on dense grids. Different formulations for 3D sparse convolutions have been investigated by Graham et al. [41] and Choy et al. [25]. In this thesis, we use the definition and implementation provided by Choy et al. [25] in the MinkowskiEngine library. They represent the input data as sparse tensors that contain only non-empty input voxels and apply the convolutional kernels only to the non-empty input voxels.

A sparse tensor is an extension of a sparse matrix where non-empty elements of the voxel grid are represented as a set of voxel coordinates and their corresponding features. A 3D sparse tensor with N coordinates $\boldsymbol{C}$ and features $\boldsymbol{F}$ is defined as:

$$\boldsymbol{C} = \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \boldsymbol{f}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{f}_N^\mathsf{T} \end{bmatrix}, \tag{2.3}$$

where x_i, y_i, z_i are the discretized coordinates using Equation (2.1). In a sparse voxel grid, not all neighbors are present, and therefore, different from dense convolutions in Equation (2.2), the sparse convolutions are defined by the kernel and the input and output coordinates. Choy et al. [25] propose to use the generalized convolution on a sparse tensor to account for this, which is defined as

$$\boldsymbol{f}_{\boldsymbol{u}}^{\text{out}} = \sum_{\boldsymbol{i} \in \boldsymbol{u} + \mathcal{V}^D(\boldsymbol{u}) \cap \boldsymbol{C}^{\text{in}}} \boldsymbol{W}_{\boldsymbol{i}} \boldsymbol{f}_{\boldsymbol{u}+\boldsymbol{i}}^{\text{in}} \quad \text{for } \boldsymbol{u} \in \boldsymbol{C}^{\text{out}}, \tag{2.4}$$

where $\mathcal{V}^D$ is the set of offsets that defines the local neighborhood given by the kernel, $\boldsymbol{C}^{\text{in}}$ is the set of non-empty input coordinates and $\boldsymbol{C}^{\text{out}}$ is the set of non-empty output coordinates. To only consider non-empty voxels for the feature computation, the set of offsets applied to location $\boldsymbol{u}$ is intersected with the set of non-empty coordinates $\boldsymbol{C}^{\text{in}}$, and furthermore, we only compute output features for non-empty voxels with coordinates in $\boldsymbol{C}^{\text{out}}$. We show an example of a 2D sparse convolution in Figure 2.2. The input 2D image presents non-zero elements (blue) and therefore the output feature map also contains only non-zero elements in those spatial locations (green). When applying the kernel (gray) in the convolution, we only consider the non-zero input values (blue).

In this thesis, we use the MinkowskiEngine library for sparse convolutions to process LiDAR data and extract features in Chapter 4 and Chapter 5.

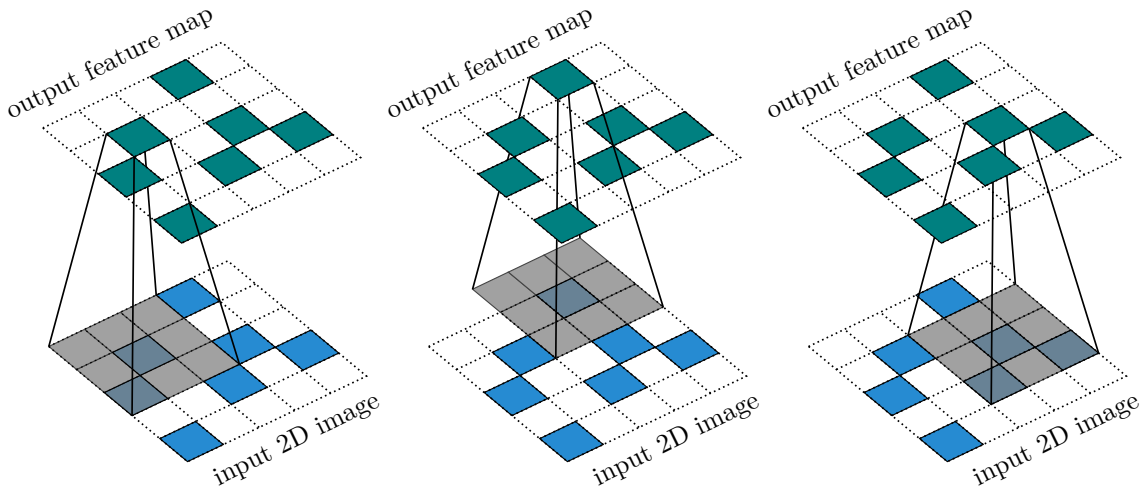


Figure 2.2: Example of a sparse convolution. Instead of computing output features for all input positions, we find non-zero elements in the input (blue) and only compute output features (green) for these input positions, using only non-zero inputs when applying the kernel (gray).

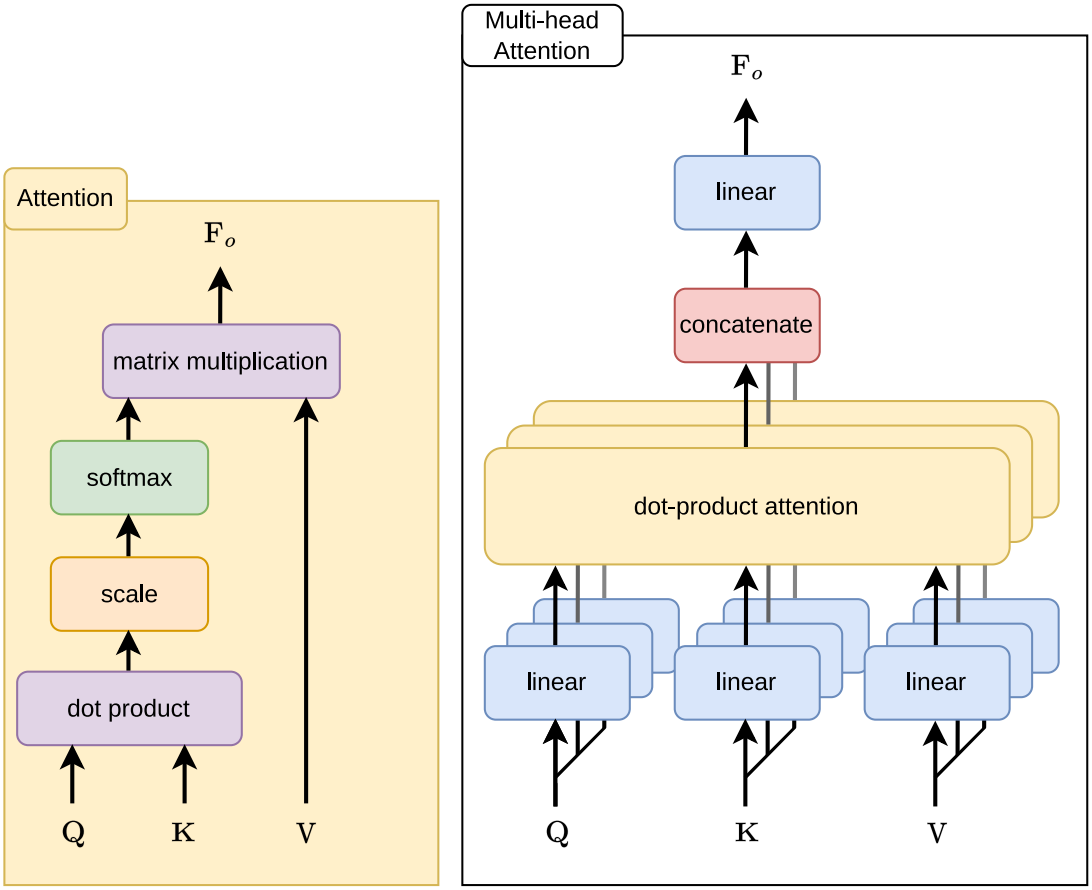
2.3 Transformer Architecture

Convolutional neural networks extract features by capturing local spatial patterns in the data. Transformers, on the other hand, are able to capture global dependencies and context, focusing on the relationship between features. They were originally proposed for the natural language processing domain, but in this section, we discuss Transformers for point cloud processing. At the core of the transformer is the attention mechanism [142] that weighs the importance of the input to extract features that capture dependencies in the data.

Given the source features $F_s \in \mathbb{R}^{N_s \times D}$, the objective is to obtain output features $F_o \in \mathbb{R}^{N_t \times D}$ for some target features $F_t \in \mathbb{R}^{N_t \times D}$. The first step is to encode these features using linear projections to obtain three matrices: queries $Q = W_v F_t$, keys $K = W_k F_s$, and values $V = W_v F_s$. The naming is an analogy to the retrieval process in a database, where queries represent what information we are looking for, keys are the "address" of each piece of information, and values are the actual information to be retrieved. The attention mechanism boils down to a linear combination of the value vectors $V \in \mathbb{R}^{N_s \times D}$ based on the similarity between the queries $Q \in \mathbb{R}^{N_t \times D}$ and the keys $K \in \mathbb{R}^{N_s \times D}$ to obtain new features $F_o \in \mathbb{R}^{N_t \times D}$

$$F_o = AV = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right) V. \tag{2.5}$$

The weighting $A \in \mathbb{R}^{N_t \times N_s}$ for the linear combination is called attention weight and is computed through the dot product between each query and each key, and encodes the similarity between all pairs. This means that the higher the similarity



(a) Attention Mechanism (b) Multi-Head Attention

Figure 2.3: The attention mechanism shown in (a) computes the dot product between Q and K, which follows a scaling and a softmax operation. Given these computed weights, the output features F_o are a linear combination of V. The multi-head attention depicted in (b) computes attention independently for different parts of the features and combines them together via concatenation and a final linear layer.

between a query-key pair, the higher the contribution of the associated value to the output feature. To ensure that attention weights are always positive and add up to one, each row of A is scaled by $1/\sqrt{D}$ followed by a softmax activation. We show a diagram of the attention mechanism in Figure 2.3(a).

To allow the model to capture different types of relationships and patterns in the input data, the features are split based on their dimension, and multiple attention "heads" are independently applied to each part. Instead of relying on a single attention mechanism, the so-called multi-head attention showed in Figure 2.3(b) processes the information in parallel, each head focusing on different aspects of the feature. Briefly, this means splitting the features into a fixed number of groups and applying attention to each one of them to focus on the similarities of each particular part. The output features are obtained by concatenating the output of each attention head and using a linear projection.

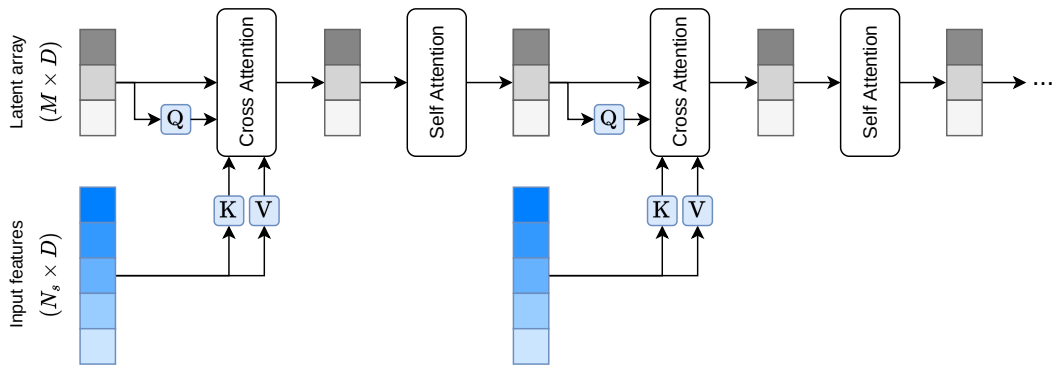


Figure 2.4: Diagram of the Perceiver. This model uses asymmetric cross-attention between the latent array L and the input features F_s followed by self-attention applied to the latent array. Since $M \ll N_s$, the complexity of the operations is drastically reduced.

The attention weights A encode feature similarities but do not account for spatial relationships, which is crucial to understand the local context. Using point cloud data, the 3D position of each point and the relation to other nearby points provide valuable information that should be included in the feature extraction. To add this spatial information, a positional encoding is typically added to the features. The standard encoding [142] consists on evaluating trigonometric functions with different magnitudes and frequencies using the Cartesian coordinates of each point. The obtained positional encoding is then added to the features to provide the spatial information.

Depending on the source and the target features, attention can be divided into self-attention, where source and target are the same $F_s = F_t$, and cross-attention, where source and target features are different. Normally, the attention operations are stacked together to form a transformer "layer" which consists of a self-attention operation followed by cross-attention and finally a linear layer called feedforward network.

2.4 Asymmetric Attention

In self-attention, the sequence lengths are equal $N_s = N_t$ and the attention weights A grow quadratically with the sequence length. We are interested in processing LiDAR point clouds, which contain around 100,000 points, and therefore using the raw point cloud as input to the self-attention leads to unacceptable computational cost and memory consumption. To handle very large inputs, Jaegle et al. [64] propose the Perceiver architecture to leverage an asymmetric

attention mechanism to iteratively distill inputs into a tight latent bottleneck, as shown in Figure 2.4. The authors propose to use a fixed small learnable latent array $\mathsf{L} \in \mathbb{R}^{M \times D}$, where $M \ll N_s$ is a hyperparameter according to the task, and use cross-attention to project the high-dimensional input features $\mathsf{F}_s \in \mathbb{R}^{N_s \times D}$ (such as point-wise features) to this fixed-dimensional latent bottleneck before processing it using multiple Transformer layers in the latent space. The model iteratively attends to the input features by alternating cross-attention and self-attention blocks and eliminates the quadratic scaling problem of all-to-all attention. By leveraging the asymmetric cross-attention, the Perceiver is performing end-to-end clustering of the inputs with query positions as cluster centers. Since the queries are a projection of the learned latent array with size $M \times D$, the self-attention requires a matrix multiplication with matrices of small dimensions and in this case the self-attention, with complexity $O(M^2)$, where $M \ll N_s$ is not a costly operation. The cross-attention, on the other hand, with complexity $O(M \times N_s)$, presents a large reduction in computational cost and memory consumption, making it possible to apply this architecture to large input sequences. In the image domain, several works adopt this configuration to perform object detecion [18], semantic segmentation [23], and panoptic segmentation [22].

In this thesis, we use the learnable latent array and asymmetric attention in Chapter 5 and Chapter 6 to perform 3D and 4D panoptic segmentation.

2.5 Mask Attention

The global context of the Transformer architecture leads to good performance but suffers from slow convergence because it takes many training epochs for cross-attention to learn to attend to specialized regions. To improve performance and achieve faster convergence, Cheng et al. [22] propose mask attention, a variant of cross-attention that, instead of attending to all locations in the input features, restricts the attention to within the foreground region of a predicted mask for each query. The mask attention modifies the attention weights computation via

$$\mathsf{F}_o = \mathsf{A}\mathsf{V} = \text{softmax}\left(\mathcal{M} + \frac{\mathsf{Q}\mathsf{K}^\top}{\sqrt{D}}\right)\mathsf{V}. \tag{2.6}$$

Where the attention mask $\mathcal{M}$ at feature location (x) is

$$\mathcal{M}(x) = \begin{cases} 0 & \text{if } \mathsf{M}(x) = 1 \\ -\infty & \text{otherwise} \end{cases}.$$

Here, $\mathsf{M} \in \{0, 1\}^{N_s}$ is the predicted binary mask. In their work, Cheng et al. [22] stack multiple decoder layers and in each layer predict a mask for each query to use in the next mask attention operation.

In this thesis, make use of the mask attention in Chapter 5 to perform 3D panoptic segmentation and modify it in Chapter 6 to add spatial prior information and perform 4D panoptic segmentation

2.6 Use in this Thesis

In this section, we provide a short description of where we use the different neural network architectures for point cloud processing. As convolutional neural networks are able to generate descriptive local features, we use them to extract point-wise features throughout the thesis. In Chapter 4, we use sparse convolutions to extract per-instance features and in Chapter 5 and Chapter 6 we use them to build U-shaped architectures to extract point-wise features from the input point cloud. Transformers, in contrast, are good at capturing long-range dependencies, and using an asymmetric cross-attention together with a learnable latent array allows for processing of larger inputs. In Chapter 5 and Chapter 6, we use a modified version of the Perceiver [64] together with mask attention [22] to perform 3D and 4D panoptic segmentation. The learnable latent array represents mask proposals, which iteratively attend to the point-wise features via cross-attention and others via self-attention.

Chapter 3

Related Work

To achieve autonomous driving without the need for human intervention, self-driving vehicles need to fully understand their surroundings geometrically and semantically, using the information coming from a suite of sensors that complement each other. While passive sensors, such as RGB cameras, provide rich, high-resolution image information, they lack depth information, which can be added by incorporating a LiDAR sensor into the hardware stack.

Using RGB images for semantic scene understanding was drastically boosted by the recent advances in neural networks, particularly large-labeled datasets [45, 72, 86] and pretrained models [72, 91, 127]. However, RGB images do not provide a 3D geometric structure of the scene, and the reconstruction of an accurate and dense 3D model using only camera data in near real-time as input poses a significant challenge. In contrast, LiDAR scans directly provide a 3D representation of the surrounding scene as a point cloud, and the main challenge lies in the segmentation of such point clouds and the semantic part of the scene understanding.

We divide this chapter into two sections, corresponding to Part I and Part II of this thesis. In Section 3.1, we introduce approaches that tackle different tasks related to semantic scene understanding using LiDAR data. In Section 3.2, we focus on the methods and techniques involved in performing occupancy prediction using images as input.

3.1 LiDAR Semantic Scene Understanding

LiDAR sensors are active sensors that provide a geometric representation of the scene as the distance to each surface around the vehicle. This depth information is beneficial for spatial understanding and can provide complementary information to achieve a more complete scene understanding. Therefore, there has been a lot of interest in advancing point cloud processing [44]. In this section, we provide an overview of the different tasks that tackle different aspects of scene understanding

for autonomous vehicles using only LiDAR information. We describe the aim of the individual tasks, such as semantic segmentation, instance segmentation, and multi-object tracking, and provide an overview of the extensive related work and how the individual tasks have been tackled. Given the unordered nature of the LiDAR data, we describe methods that use different representations for point cloud processing, such as using the raw unstructured point cloud, voxel grids, or range images. Some tasks are a combination of individual tasks and try to provide a more holistic scene understanding. In this context, we describe and provide works that tackle panoptic segmentation and 4D panoptic segmentation. We also describe different paradigms to tackle panoptic segmentation, such as top-down and bottom-up approaches, the formulation of mask segmentation, and the usage of contrastive learning in a supervised setup.

3.1.1 Semantic Segmentation.

This task aims to provide a semantic label to each point in the point cloud, and it can be tackled using different data representations.

In the seminal work PointNet [121], Qi et al. propose to directly process the unordered point cloud using multi layer perceptron (MLP)s along with pooling operations to extract a single feature vector for the whole unordered set of points. To extend it to a hierarchical approach, PointNet++ [123] applies PointNet repeatedly to overlapping partitions of the input set at different scales. Each layer learns local features and downsamples its input point cloud using further points to evenly cover the whole set. These approaches process the point cloud directly, which is usually time-consuming and computationally expensive for large point clouds in outdoor scenes. To alleviate the usage of expensive sampling operations, RandLA-Net [55] relies on random point sampling to infer per-point semantics for a large-scale point cloud. However, since random sampling can discard key information, the authors propose a local feature aggregation module to capture complex local structures by progressively increasing the receptive field of each 3D point and preserving the geometric details. Still processing the raw, unstructured point cloud, other works build point kernels [138] and define continuous convolutions for an unordered set of points. The convolutional kernel consists of a set of kernel points with associated positions. During training, the kernel learns a weight for each position and the continuous weight space is obtained through interpolation of these discrete positions. Other approaches [27, 82, 101, 171, 173] project the LiDAR point cloud into a 2D image and segment it using 2D convolutional networks, leveraging well-known architectures [48, 125] for image processing. Focusing on achieving fast LiDAR-only semantic segmentation, Milioto et al. [101] propose to exploit range images as an intermediate representation and use a 2D CNN to perform image segmentation. The proposed method, RangeNet++,

operates on a spherical projection of the input point cloud, similar to a range image, to exploit the way the points are detected by a rotating LiDAR sensor. This yields an efficient approach but leads to issues caused by discretization or blurry CNN outputs. To resolve these issues, the authors transfer the semantics from 2D to 3D to recover all points from the original point cloud, regardless of the discretization during the projection to the range image. To further enhance the performance, the authors propose a post-processing step to clean the effects of the discretization and inference artifacts using a graphics processing unit (GPU)-based kNN-search algorithm that operates on all points. Following this line of work, Cortinhal et al. [27] also project the point cloud into a 2D image and propose several architectural modifications to enhance performance. The authors introduce a context module before the encoder to capture the global context information in the full 360° LiDAR scan, which consists of a stack of residual dilated convolutions, fusing receptive fields at various scales. The focus of their work is to perform uncertainty-aware semantic segmentation, computing the epistemic and aleatoric uncertainties for each point in the cloud, which can be propagated to the subsequent modules, such as decision-making to achieve safe maneuver planning or emergency braking. To achieve even faster inference and keep the network lightweight for mobile robotics applications, Li et al. [82] propose a method that uses multiple feature paths to different scales independently and balances the computational resources between scales. The top path extracts low-level features using shallow layers operating on high-resolution feature maps, and the bottom path uses more complex operations and operates on low-resolution feature maps to extract high-level semantic information. To avoid redundant computations across the paths and make the network more efficient, the authors propose an additional dense top-to-bottom interaction strategy between the scales, where feature maps from top paths are passed to all lower paths. Instead of relying on spherical projections, Zhang et al. [171] project the LiDAR point cloud into a bird's eye view (BEV) representation using polar instead of cartesian coordinates. The authors argue that not only the size but also the shape of the perception field matters in the feature extraction process, and therefore use polar coordinates to account for the imbalanced point distribution caused by the distance-dependent sparsity of LiDAR point clouds. To this end, the authors propose to let the CNN perception fields track the special ring structure by partitioning a LiDAR scan with a polar grid. They connect the leftmost and rightmost columns of feature vectors to be connected and modify the CNN to be capable of convolving continuously on the polar grid.

State-of-the-art approaches strike a balance between memory usage and performance by partitioning the space in 3D voxels [25, 178] to keep topological relations and use sparse 3D convolutions [41], which operate only in the occupied

space, which represents a low percentage of the whole scene. In this context, Zu et al. [178] divide the space into cylindrical voxel partitions to meet the key properties of the outdoor LiDAR point clouds, namely sparsity and varying density with distance to the sensor. This way, a more balanced representation is achieved by covering further away regions with larger cells due to the increasing grid size while maintaining the 3D topology and geometric point relations. Similarly, Lai et al. [75] propose to use radial windows to partition the space into narrow and long voxels to overcome the disconnection and limited receptive fields for the sparse distant points. They directly aggregate information from dense close points to the sparse distant ones and boost the performance mainly for the distant points. There are also works [1, 137, 159, 160] that combine the different representations, such as voxel, points, and range images, to extract features and merge them to get a better performance.

3.1.2 Instance Segmentation

While semantic segmentation assigns a class to each 3D point, it does not differentiate between instances belonging to the same class. To this end, instance segmentation seeks to identify the individual objects in the scene by assigning the same ID to points belonging to the same instance.

Lang et al. [76] use PointNets to learn a representation of point clouds organized in vertical columns or pillars. The extracted features for each pillar are stored in a bird's eye view (BEV) 2D image to preserve object scales and the local range information. They use 2D convolutional networks to process these 2D images and avoid the use of expensive 3D convolutions.

Several works [66, 145] use bottom-up approaches and two-branch networks to extract point features and predict semantic labels and offsets to the instance center. They use these offsets to shift the points and cluster points into instances to obtain the final predictions. Jiang et al. [65] utilize the original and the offset-shifted point coordinate sets to take advantage of their complementary strength and also predict a score for each cluster for better filtering. Vu et al. [145] performs soft grouping followed by top-down refinement. The method allows points to be associated with multiple classes to mitigate problems from semantic prediction errors and learns to categorize false positive instances as background. Following these methods, Hu et al. [54] tackle class-agnostic instance segmentation by removing the background points and using hierarchical clustering. They search over a large space of candidate segmentations and select one where individual instances score well according to a data-driven point-based model of "objectness." Zhang et al. [169] propose a dense feature encoding technique to allow localization and segmentation of far-away small objects and a strategy to handle severe class imbalance. Moving away from the bounding box representation of objects,

Yin et al. [163] propose to represent 3D objects as points. They first detect centers of objects using a keypoint detector [175] over the 2D projected point cloud and regress other attributes like 3D size, orientation, and velocity from the center's point feature. In a second stage, their proposed CenterPoint uses additional point features to recover the local geometric information to perform detection and tracking. Based on transformers, Carion et al. [18] first performs object detection in images by predicting bounding boxes with learnable queries and solving an end-to-end set prediction problem. They add a mask head like Mask R-CNN [47] to decode masks and obtain instance segmentation. Misra et al. [102] extends this work to 3D point clouds by randomly sampling a set of points and using them as queries in the transformer decoder. This allows them to vary the number of predictions used at inference to trade performance for running time. Schult et al. [130] builds on recent transformer-based methods for object detection and instance segmentation, and represent each 3D object with an instance query and use the transformer decoder to iteratively attend to point cloud features at multiple scales. As a result, they can directly decode all instance masks in parallel.

3.1.3 Panoptic Segmentation

Panoptic Segmentation [69] refers to the joint task of tackling semantic and instance segmentation using 3D point clouds. The semantic classes are divided into uncountable "stuff" classes and countable "thing" classes. The task requires providing for each point in the point cloud a semantic class, and for the points belonging to "thing" classes, an additional instance ID. Panoptic segmentation methods can be divided depending on the data representation used to process the point clouds. To deal with 3D data while keeping the runtime low, some approaches project it into a 2D representation like range images [61, 79, 100, 132] or bird's-eye-view [176] and use 2D convolutional networks. They leverage 2D vision architectures but lose 3D geometric information, harming the final performance. Another alternative is to divide the space into 3D partitions to maintain the geometric relations between the data points and apply 3D convolutional networks. Gasperini et al. [37] use point convolutions [138] to improve the results but require a higher computational cost and apply hard downsampling. Hong et al. [51] use sparse convolutions [41] and improve results by voxelizing the space using cylindrical coordinates to match the distance-dependent density of the LiDAR point clouds. Regarding the fusion of semantic and instance segmentation, panoptic segmentation methods can also be divided into bottom-up and top-down approaches. Top-down methods [61, 132] often follow image-based works and add a semantic head to Mask R-CNN [47] and perform panoptic segmentation on range images. Sirohi et al. [132] propose to project the point cloud into a range image,

but also to use 3D information from the point cloud. The method consists of a shared backbone that encodes geometric information and aggregates semantically rich range-aware multi-scale features. The authors propose to use scale-invariant semantic and instance segmentation heads and introduce a panoptic fusion module, which they supervise by a panoptic periphery loss function. Bottom-up methods [51, 79, 80, 100, 124] are predominant in the LiDAR domain. They use the semantic segmentation predictions to filter the "stuff" points and apply a clustering method as post-processing to get the final instance segmentation, which does not allow training these methods end-to-end.

Milioto et al. [100] propose a single-stage and real-time capable panoptic segmentation approach based on a shared encoder and two decoders. The authors use an instance decoder to predict, for each point, an offset that points toward the object center, combined with a loss function [12, 106] to pull spatial embeddings of points belonging to the same instance together.

This leads to the segmentation of individual instances. The semantic encoder provides semantic labels, and the extracted embeddings are used to aid the clustering of object centers. Key to their method is a distance-aware tri-linear upsampling that enables the usage of larger output strides compared to transpose convolutions, leading to substantial savings in computation time. Hong et al. [51] use a strong backbone with cylinder convolutions and use the extracted features as input to both the semantic and the instance branch. The authors argue that commonly used clustering algorithms like BFS or DBSCAN are incapable of handling complex autonomous driving scenes with non-uniform point cloud distributions and varying instance sizes, and therefore propose a dynamic shifting module, a clustering method that adapts kernel functions on the fly for different instances. Following a proposal-free approach, Razani et al. [124] use an instance-level network that processes the semantic results by constructing a graph convolutional network to identify foreground objects and fuse them with the background classes. The network first over-segments the point cloud and subsequently processes them using 3D sparse convolutions to embed each cluster, which is treated as a node in the graph, and its embedding as a node feature. To generate the final predictions, the graph CNN predicts edges between cluster pairs, obtaining instance segmentation. Li et al. [80] use a backbone that fuses fine-grained voxel features and 2D bird's eye view features with different receptive fields to capture both detailed and global information while considering accuracy and inference speed. To model the interaction between "thing" points and enhance the offset regression, the authors propose a knn-transformer module, which focuses on the relationships of spatial distance among local points. To group points into instances, the authors introduce a clustering pseudo heatmap paradigm. They first shift the points, and then project them onto a BEV im-

age using the number of projected points per pixel as a score. In this pseudo heatmap, the instance centers are yielded by a window-based max-pooling and then are used to cluster all things points, removing this way the separate heatmap learning branch. Gasperini et al. [37] propose a learning-based clustering method to train the model end-to-end by optimizing two complementary loss functions based on a confusion matrix between ground-truth objects and a fixed number of predictable instances. Still, they require several post-processing steps to reduce under- and over-segmentation effects and fix the clustering errors to achieve competitive performance. In contrast, Li et al. [78] propose a cluster-free instance segmentation head by pillarizing foreground point embeddings and finding the connected pillars with pairwise embedding comparison.

3.1.4 Mask Segmentation

In their pioneering work for object detection in images R-CNN [39], Girshick et al. proposed to apply convolutional neural networks to bottom-up region proposals to localize and segment objects. This approach is based on a region proposal algorithm and uses a CNN to extract a single feature vector for each region, which is later classified by category-specific linear SVMs. The authors emphasize that, because R-CNN operates on regions, it can be easily extended to the task of semantic and instance segmentation. He et al. [47] extend this work by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. This approach tackles the instance segmentation tasks by predicting a set of binary masks and corresponding semantic classes. Instead of using region proposals, Carion et al. [18] propose to see object detection as a direct set prediction problem, removing the need for hand-designed components like non-maximum suppression. The approach uses an encoder-decoder architecture based on transformers [142], where the self-attention mechanisms explicitly model all pairwise interactions between elements in a sequence, making this architecture suitable for set prediction by removing duplicate predictions. To compute the set loss, they use bi-partite matching and pair all ground-truth bounding boxes with the predictions made using a fixed set of learnable object queries. To generalize the model to produce panoptic segmentation in a unified manner, the authors add a mask head on top of the decoder outputs. Cheng et al. [23] propose to remove the bounding boxes to perform semantic and panoptic segmentation by using the set of learnable queries to directly output binary masks along with their semantic classes. They use a fully convolutional network to extract image features and a transformer decoder to refine the queries, which act as mask proposals. The binary masks are obtained via the dot product between the per-pixel embeddings from the CNN and the mask proposals. The model tackles both semantic and instance segmentation tasks in a unified manner without any

change in the model and losses. Extending this work, Cheng et al. [22] apply this concept to achieve semantic, instance, and panoptic segmentation in a top-down way without manually-designed post-processing steps like non-maximum suppression to remove duplicated predictions. Similar to previous works, this approach [23] relies on a few key components, namely a backbone feature extractor, a pixel decoder, a Transformer decoder, and a set of learnable queries as mask proposals. Here, Cheng et al. introduce masked attention to replace the cross attention in the Transformer decoder. This way, they restrict the attention to localized features within a predicted mask instead of attending to all locations in the image in the standard cross-attention, which leads to faster convergence and improved performance.

These approaches have been validated in the 2D image domain, but have not been applied to outdoor 3D LiDAR point clouds, where the different data modalities might pose different challenges.

3.1.5 Multi-object Tracking

Multi-object tracking in the vision domain is usually solved using the tracking-by-detection paradigm [126, 156] formulated as a data association task consisting of two steps: obtaining object detections in the current frame and associating them with previous detections across time. This paradigm usually relies on the Hungarian Algorithm to perform a bi-partite matching between current and past detections, relying on a matrix that provides a cost to each temporal assignment of instances. To construct such a cost matrix, Weng et al. [156] rely only on a constant velocity motion model, providing a strong tracking baseline using 3D object detectors and representing objects as bounding boxes. Instead of representing objects as bounding boxes, Yin et al. [163] represent, detect, and track objects as points. They first detect centers of objects using a keypoint detector and regress them to other attributes such as 3D size, orientations, and velocity. In a second stage, they refine the estimates using more point features on the object and perform tracking by greedy closest-point matching.

Other approaches [28, 135] rely on an appearance model. They use metric learning to generate distinct instance embeddings, and they use feature similarity to build the cost matrix for the data association. To obtain such embeddings, they provide, during training, positive and negative samples to enforce that positive pairs of detections have more similar features than negative pairs. The samples are detections of the same and other objects over time. Among these works, Dong et al. [28] use a triplet loss with one positive and one negative sample. Hermans et al. [49] use hard batch triplet loss to mine a hard negative sample, which is more beneficial for learning. Son et al. [135] add an extra positive sample to increase temporal consistency. The performance of the tracking

relies on the quality of the appearance and motion models [7]. When using deep learning to tackle this task, the focus of the research is on learning these models and their effective combination. In the image domain, recent state-of-the-art object detectors build on the ideas of DETR [18] and propose end-to-end tracking methods [98, 166] reusing the output queries for tracking. They save the output queries of the previous steps and use them with the current camera images to decode consistent instances over time. These works usually focus on a single class or a small subset of classes and only track bounding boxes without segmentation.

3.1.6 Contrastive Learning

Contrastive learning for self-supervised representation-learning [19, 20, 141] became popular due to the performance improvement on image classification tasks. These methods tackle the lack of annotated data with a pre-training step that requires no labels. They rely on data augmentations to generate different views of one sample and train the network with a pretext task that aims to learn an embedding space in which the generated augmented views have similar representations and are different from other samples. In this setup, a data point is used as an anchor, positive samples are augmented views of it, and the negative samples are the augmented views of other data points in the batch. The loss seeks to compare the generated encodings for each sample in an unsupervised way without labels. The goal is to learn a generic feature representation in which embeddings from positive samples lie close together and far from embeddings of negative samples. To include more negative samples, He et al. [46] use two networks to obtain features. One is updated normally via back propagation, and the second one is a momentum-moving average of the first one. The output of the second network is accumulated over time to obtain a large memory bank of negative samples. The momentum update makes the second network update more smoothly, which leads to much more consistent representations in the memory bank.

The losses used in these methods [19, 20, 141] are inspired by noise contrastive estimation [104] or N-pair losses [134], and they are typically applied at the last layer of a neural network. At inference time, the last layer is dropped, and the embeddings from a previous layer are used to transfer to a downstream task via fine-tuning or direct retrieval tasks.

Related to contrastive learning, the losses based on triplets [129, 155] are used to learn representations in a supervised setting where the choice of positive and negative pairs is guided by the labels. The main distinction with the triplet losses is the number of positive and negative pairs per data point. The triplet losses use one positive and one negative pair per anchor. The positive is chosen from the same class and the negative from other classes, usually requiring hard-negative mining to achieve good performance [129]. The main difference is that in

contrastive losses, many negative pairs are used for each anchor. These samples are normally randomly chosen, assuming a very low probability of false negatives.

When labels are available, Khosla et al. [67] propose the supervised contrastive loss, which aims to use several positive along with many negative samples. With this approach, clusters of points of the same class are pulled together in embedding space, while at the same time pushing apart clusters of different classes. The main differences with self-supervised contrastive learning are the inclusion of many positives per anchor in addition to many negatives, and the fact that the positives are samples of the same class as the anchor, rather than being data-augmented versions of the anchor. This loss [67] can be seen as a generalization of the triplet [155] and N-pair losses [134]. Using many positives and many negatives for each anchor yields good performance without the need for hard negative mining, which is usually difficult to tune. In the context of multi-object tracking, contrastive learning could be used at an instance level to obtain descriptive instance embeddings acting as an appearance model, distinctive between different instances.

3.1.7 4D Panoptic Segmentation

There have been a few works to unify tracking and segmentation tasks. In the image domain, Kim et al. [68] propose video panoptic segmentation to extend panoptic segmentation [69] from 2D images to the video domain, focusing on short and sparsely labeled video snippets in an offline setting. This new task aims to simultaneously predict object classes, bounding boxes, masks, instance ID associations, and semantic segmentation, while assigning unique answers to each pixel in a video. Video panoptic segmentation allows for evaluating applications that require a holistic video segmentation, and the temporally dense panoptic segmentation of a video can be used as an intermediate representation for other tasks, such as temporal reasoning to anticipate the behaviors of the participants. While most of the approaches use synthetic data or only sparse annotations, Weber et al. [152] tackle this task in real-world scenarios by requiring dense interpretation in both spatial and temporal domains. In contrast, Voigtlaender et al. [144] extend the multi-object tracking task with segmentation to tackle the task of instance segmentation and tracking. They propose to jointly address detection, tracking, and segmentation with a single convolutional network.

Such tasks aim to segment the 2D images captured from the scene and identify objects over time, but in the image domain. However, autonomous vehicles need to understand the 3D surroundings and localize traffic participants in a 4D continuum. Hurtado et al. [61] propose the task of multi-object panoptic tracking (MOPT) to unify semantic segmentation, instance segmentation, and multi-object tracking both in the image and in the LiDAR domain. Instead of

combining the predictions of the three heads in a post-processing step, the authors propose an architecture consisting of a shared backbone with the 2-way Feature Pyramid Network, and task-specific instance, semantic, and tracking heads together with a fusion module to adaptively yield the multi-object panoptic tracking output.

4D Panoptic Segmentation [2] is the extension of 3D panoptic segmentation of LiDAR scans to the temporal domain by additionally requiring temporally consistent instance IDs. The task boils down to predicting, for each point, a semantic class and a unique instance ID for the whole sequence. To tackle such a task, a line of work is to use off-the-shelf 3D panoptic segmentation networks [101, 138] to first segment the scene and obtain the instances in each scan and then associate the segmented instances over time to obtain consistent instance IDs. These methods are not end-to-end trainable because they rely on an off-the-shelf network. Aygun et al. [2] address the task with their proposed method 4D-PLS. They use as input 4D volumes by aggregating a few consecutive LiDAR scans and perform panoptic segmentation in a bottom-up fashion. Because they use a 4D point cloud, they cluster instances across time and space, simultaneously achieving instance segmentation and tracking. Given that they use short sequences as input, in a post-processing step, they associate the short sequence predictions to get sequence-level predictions. In 4D-PLS, the object instances are modeled as Gaussian probability distributions over the foreground points, and the network predicts the Gaussian parameters, together with object centers. They generate tracklets by clustering points over space and time. Given an object center, they assign points to this instance by evaluating each point under the Gaussian probability function-based on the feature of each point.

Following that line of work, Hong et al. [52] also align and overlap consecutive LiDAR point clouds and use the resulting 4D point cloud as input. They further use sparse convolutions over cylindrical partitions [178] to extract point-wise features, a semantic head to predict the semantic logits for each point, and an instance branch to predict offset vectors to regress the instance center for each point. To obtain tracklets, the authors regress to the instance center of the overlapped point clouds. As a clustering method, they propose a dynamic shifting (DS) module to iteratively apply a kernel function as in Mean Shift. However, instead of using kernels with fixed bandwidth, which do not adapt to instances with varying sizes, the DS module automatically adapts the kernel function for each LiDAR point. This is done by weighting across several bandwidth candidates to dynamically adapt to the optimal one. With the DS module, the regressed centers can be dynamically shifted to the correct cluster centers.

Kreuzberg et al. [73] design an architecture called 4D-StOP and follow 4D-PLS [2] in that they aggregate consecutive point clouds as input, but propose to

generate spatio-temporal object proposals using voting-based center predictions. In this context, each point in the input 4D point cloud votes for a corresponding center. To aggregate these individual tracklets proposals, the authors use geometric features to effectively generate a video-level 4D scene representation.

Previous methods [2, 52] use clustering and generate only one cluster center per instance, which means generating only one proposal for each object, and this limits the quality of their representation. In contrast, 4D-StOP [73] represents tracklets as spatio-temporal object proposals, usually used for object detection and instance segmentation, and aggregates these proposals into the final tracklets. To obtain the tracklets, the authors use Hough voting [122] to make each feature point vote for the closest center point to generate object proposals, which are aggregated using high-level learned geometric features.

More recently, Zhu et al. [177] propose rotation-equivariant modules that they apply to existing feature extractors of previous methods [2, 73] to enhance the feature extraction and improve performance. In their work, the authors build on top of previous methods [2, 73]. They aggregate consecutive input point clouds to create an input 4D volume and perform clustering over space and time to obtain the instances. As a difference, however, Zhu et al. [177] review the center and offset prediction in the instance clustering step and predict invariant scalar fields and equivariant vector fields instead.

3.2 Occupancy Estimation

RGB cameras are passive sensors that transform the light reflected from the scene into images, capturing visual information in a 2D plane. They provide rich texture and color information with high resolution and can be found at the core of the sensor suites of many autonomous systems due to their low cost. The recent advancements in computer vision in the image domain, with the large amounts of data [45, 72, 86] and pretrained models [72, 91, 127] contribute enormously to the task of semantically understanding each individual image. However, the information provided by RGB cameras lies in the 2D image plane and does not include 3D spatial information, key to the spatio-temporal understanding of the surrounding scene. There are multiple ways of obtaining 3D information using only RGB camera information, such as 3D object detection, depth estimation, and semantic occupancy prediction.

Semantic occupancy estimation aims to use only RGB images from the current timestep to reconstruct the surrounding scene and provide semantic knowledge. It specifically divides the scene into a voxel grid and seeks to predict for each voxel whether it is occupied or not, and its corresponding semantic class. Compared to 3D object detection, occupancy estimation provides a more fine-grained

representation, and in contrast to monocular depth estimation, it seeks to also reconstruct the environment in occluded areas. This vision-centric task aims to tackle 3D scene understanding using only RGB cameras to provide crucial information needed for navigation and decision-making in autonomous vehicles.

In this section, we examine the main works belonging to the task of 3D occupancy prediction of the 3D surroundings given only camera information. We highlight recent research directions and common approaches, together with the current challenges.

3.2.1 Depth Estimation

Depth estimation is a key task in predicting 3D geometry given images as input. Early works on depth estimation [31, 88] use ground truth depth maps as supervision signal and see monocular depth estimation as a regressive problem. These methods usually rely on neural networks to predict depth maps for single frames and use the difference between predicted and ground truth depth as loss function. Eigen et al. [31] use a convolutional neural network composed of a global coarse-scale network and a local fine-scale network to predict depth images end-to-end. Liu et al. [88] propose to refine the estimated depth by considering depth information of neighboring pixels. They first regress the depth map with a CNN using multi-level image patches that they create via super-pixels. Finally, they refine the predicted depth from super-pixel to pixel level via a hierarchical conditional random field. To avoid relying on ground truth depth, more recent self-supervised methods [89, 116, 146, 174] use as supervision signal the photometric constraints [40] between frames in a monocular video. These methods follow the work proposed by Zhou et al. [174] and usually use a depth network to predict the depth maps and a second pose network to regress the transformation between consecutive frames. This way, the depth is optimized using multi-view consistency between consecutive frames. To reduce the effect of dynamic objects and occlusion, Wang et al. [146] design geometry-based masks to consider the blank regions on the reconstructed images due to the view changes and the occlusions generated during reprojection. Towards a universal and flexible depth estimation model, Piccinelli et al. [118] propose a model capable of reconstructing metric 3D scenes from single images across domains.

To adapt to multi-camera setups normally used for autonomous vehicles, some approaches [43, 148, 153] tackle depth estimation using the multi-view surrounding information and are able to predict real-world scale given the transformations between cameras. In particular, Wei et al. [153] use a single network to process all surrounding views simultaneously. They employ a transformer architecture with cross-view self-attention to fuse information from the multiple cameras. Furthermore, they adopt a two-frame structure from motion (SFM) in the overlap

between cameras to obtain sparse scale-aware pseudo depths to pre-train the model and increase performance. Instead of separately predicting the ego-motion for each camera, they predict a single ego-motion for the whole rig of cameras and transfer it to each view for better ego-motion consistency. This highlights how even sparse depth supervision obtained from structure-from-motion can help in obtaining real-world depth predictions. As another pretraining strategy relying only on posed images, Xu et al. [158] use Gaussian splatting to leverage large-scale multi-view posed datasets and pretrain depth estimation models, showing the synergy between Gaussian splatting and depth estimation. Guizilini et al. [43] extends monocular to full surrounding depth estimation by adding pose consistency constraints and leveraging the spatio-temporal nature of the captured data. They use spatio-temporal photometric constraints between different cameras to increase the amount of overlap, thanks to the ego-motion.

Recently, 3D foundation models such as DUst3R [149] and MASt3R [77] have demonstrated impressive performance across a wide range of 3D vision tasks. Given two input images, they can produce dense, per-pixel 3D point clouds along with confidence maps in a single feed-forward pass, and further estimate depth images, camera intrinsics and camera poses based on the inferred geometry. Leveraging Vision Transformer-based architectures [29] and a large amount of training data, these models exhibit remarkable generalization capabilities and can be extended to multi-view settings for large-scale dense 3D reconstruction [30]. Despite the impressive performance in static scene reconstruction, their ability to recover the geometry of dynamic objects remains limited due to the lack of corresponding training data. To address this issue, Zhang et al. [170] fine-tunes DUst3R using multiple RGB-D datasets captured in dynamic environments, realizing robust reconstruction for not only static background but also dynamic objects.

3.2.2 3D Occupancy Prediction

Using a grid map to represent the environment was first introduced by Moravec and Elfes [105] for indoor mapping using mobile robots. They propose to discretize the environment into equally sized cells, which can be either free or occupied. Each cell stores a probability of being occupied and is updated with consecutive sensor observations obtained by the robot. This representation was adopted by the robotics community for perception and navigation due to its effectiveness and simplicity. Stachniss et al. [136] extend this representation to handle non-static environments by modeling and filtering dynamic objects, and Grisetti et al. [42] improve the efficiency and accuracy of grid mapping in SLAM systems. To represent the space more efficiently, Hornung et al. [53] use octree structures to enable multi-resolution 3D mapping in their OctoMap framework. With the advent of deep learning, more recent methods [128] use neural networks

to infer occupancy from sparse and noisy sensor data for autonomous driving, expanding the applicability of occupancy grids for outdoor driving scenarios, where understanding the free space and obstacles is critical for safe navigation. In the context of perception for autonomous vehicles, the 3D occupancy prediction task aims to provide geometric and semantic information about the scene by dividing it into a predefined voxel grid and predicting the state of each voxel. A similar task named 3D semantic scene completion was first introduced as a benchmark in the SemanticKITTI [4] dataset based on a LiDAR sensor and a stereo camera setting. While some works focus on reconstructing the surroundings using LiDAR data [24, 81, 162, 180] others use monocular or stereo cameras [16, 60] or fuse multiple data modalities [17]. Based on the nuScenes dataset [14], Occ3D-nuScenes [139] provides 3D semantic occupancy ground-truth by aggregating LiDAR scans, where the input used for the task is the six surrounding views captured by the system.

Various methods [35, 56, 60, 139, 150] build on top of 3D object detection architectures [57, 84] to extract 2D features and lift them into a common 3D space. These architectures consist of an image feature extractor, a lifting operation to project 2D features to 3D space, and a final neural network to obtain the final occupancy prediction. While the image feature extraction is generally done with a CNN like ResNet [48], the lifting into 3D space follows one of two paradigms. One of them is the one proposed by Philion et al. [117] and adopted by following works [57, 83, 85]. It involves implicitly projecting image features to the 3D space by estimating a 3D depth distribution and lifting the 2D image features into a BEV grid. The first step (lift) involves predicting a categorical distribution over depth and a context vector for each pixel. This way, they can obtain a feature at each point along the ray as the outer product between them. The second operation (splat) involves using "pillars" (like done in PointPillars [76]) to convert the large point cloud from the previous step into a BEV image, which can be processed by a CNN. The other paradigm is proposed in the work by Li et al. [84] and adopted by other works [164]. At its core is a set of BEV queries, which are grid-shaped learnable parameters designed to query features in BEV space from multi-camera views via deformable attention [179]. After extracting 2D image features, they lift each query to a pillar and sample reference points, which are projected into each view given the intrinsic and extrinsic parameters. These points are then used to sample features in each 2D view around their locations and later aggregate the features using the proposed spatial cross-attention. The obtained 2D BEV feature grid can be processed by a 2D CNN to tackle the downstream task. Following works [139, 154, 172] use 3D volume queries instead of a BEV grid to preserve spatial information. They follow the same procedure and sample points, which they project into each view to sample features and

aggregate them. In this case, the volume features are processed by a 3D CNN. To reduce memory consumption but increase performance, TPVFormer [60] proposes to use a tri-perspective view (TPV) representation instead of relying on a BEV grid or a volume representation. This representation adds two perpendicular planes to the BEV and models each point in the 3D space by aggregating its projected features on the three planes. On the same line, Lu et al. [93] propose to use an ocree [97] for occupancy prediction to adapt to objects and semantic regions with different shapes and sizes.

Due to the challenging nature of the 3D occupancy task, most state-of-the-art methods rely on costly 3D ground truth and laborious annotations. Instead of using 3D voxel labels, some approaches [9, 114, 115] propose to use segmented LiDAR scans for supervision instead. They project the segmented point clouds into each camera to obtain sparse depth and semantic maps. This kind of supervision relies on the availability of a 3D LiDAR sensor and the laborious manual labeling of each point cloud. To relax the requirements for both supervision and inference, some approaches [34, 59, 168, 172] use only unlabeled camera images. They learn the geometry of the scene implicitly by supervising with a few temporally adjacent frames and optimizing for a proxy task like volume rendering [99]. To perform 3D semantic occupancy prediction, these methods [34, 59, 168] use pre-trained open-vocabulary segmentation models [72, 91, 127] to obtain semantic labels for each image during training and use this as semantic supervision.

Recently, a few works [91, 150, 165] aim to provide not only geometric and semantic information but also to identify individual instances. Wang et al. [150] propose to jointly perform occupancy prediction, semantic segmentation, and object detection using RGB images as input. They learn occupancy in a coarse-to-fine manner and use voxel queries to perform object detection using DETR [18]. Liu et al. [91] propose a sparse architecture that iteratively prunes empty voxels to predict occupancy in a sparse way. In a following step, they perform semantic and instance segmentation of the occupied voxels using sparse queries and a transformer decoder. Yu et al. [165] first predict occupancy leveraging FlashOcc [164] for its efficiency, and predict instance offsets and heatmaps, and use clustering to group voxels into instances in a bottom-up fashion. To perform 3D panoptic occupancy prediction, these methods rely on voxel-level annotations ground truth as supervision.

Part I

LiDAR 4D Panoptic Segmentation

Chapter 4

Constrastive Instance Association for 4D Panoptic Segmentation

In the context of mobile robotics and self-driving cars, spatio-temporal semantic scene understanding is critical for autonomous navigation in dynamic environments. These systems need spatial awareness of the surroundings to recognise the different parts of the scene, like where to and where not to drive, and identify the other agents in the scene and follow their behaviour over time. This way, an autonomous vehicle can plan and make decisions accordingly for safe navigation. 3D LiDAR sensors provide accurate geometric 3D information of the surroundings, but in the form of a sparse representation that lacks color and texture, which makes the segmentation of the scene a challenging task. In this first part of the thesis, we focus on the segmentation of the 3D scene relying only on 3D LiDAR data.

Different tasks related to dynamic scene understanding, i.e., object detection [122, 131, 161], semantic segmentation [55, 82, 101, 178], instance segmentation [66, 76, 145], and multi-object tracking [28, 135, 156], are often tackled separately. An initial step into providing a more holistic description of the scene is panoptic segmentation [69], which aims to fuse semantic and instance segmentation. The idea is to assign a semantic class to every 3D point in the LiDAR scan and differentiate between instances with a unique ID. However, understanding the dynamics requires knowing the previous trajectories of the traffic participants and identifying them in the current scene. In this chapter, we extend the panoptic segmentation task and include this temporal information, addressing the problem of 4D panoptic segmentation using LiDAR scans recorded by an outdoor platform, as proposed by Aygun et al. [2]. This requires assigning to each 3D point in a temporal sequence of scans a semantic class and a temporally consistent instance ID, as illustrated in Figure 4.1.

In 3D panoptic segmentation for LiDAR scans, state-of-the-art methods [51,

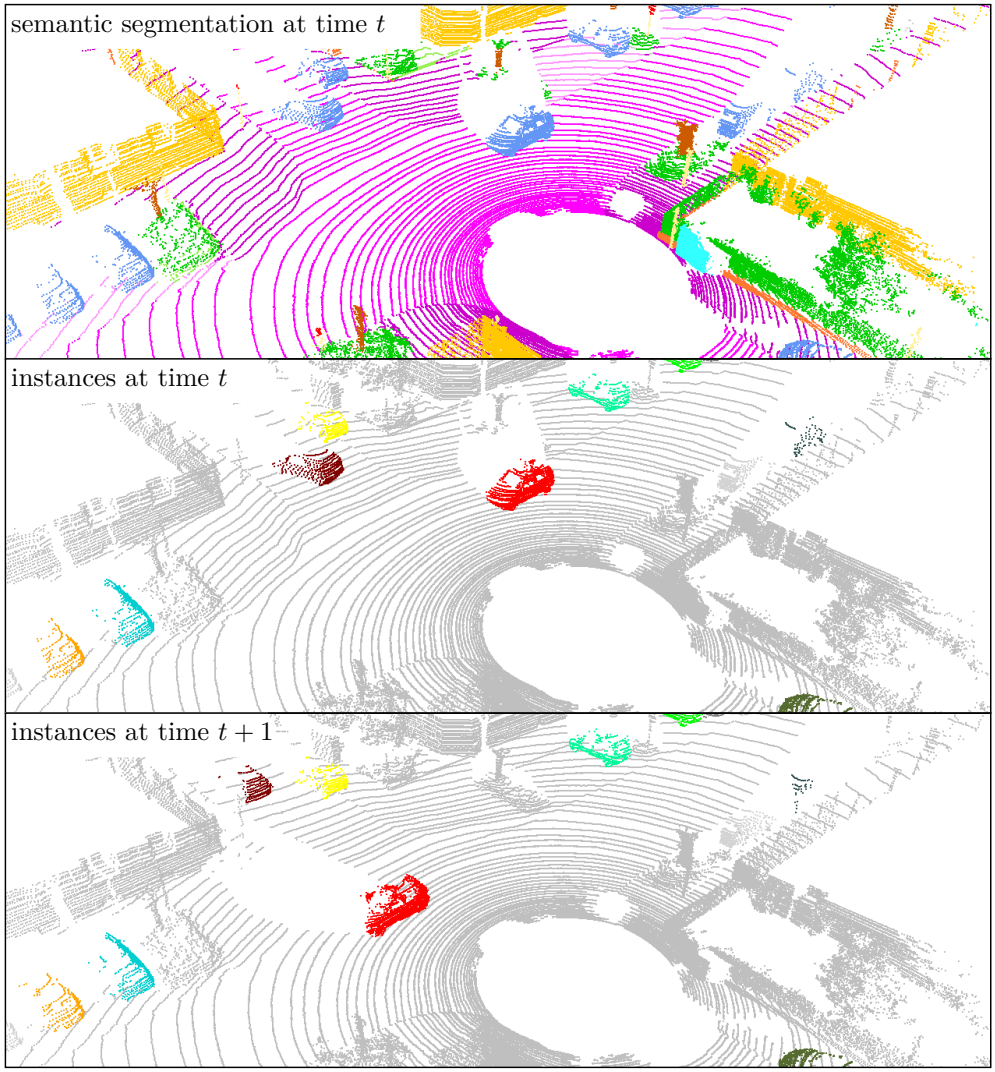


Figure 4.1: Our method allows for performing semantic segmentation (top) and temporally consistent instance segmentation (bottom) of 3D LiDAR scans. The different instance colors depict different IDs.

37, 100] rely on their strong 3D feature extractors. We argue that these features can also be leveraged to generate temporally consistent instance-wise embeddings to extend the segmentation to the temporal domain by means of the tracking-by-detection paradigm: identifying objects in individual 3D scans and then performing temporal instance associations via feature similarity.

The main contribution of this chapter is a novel method based on a sparse convolutional neural network, which takes a 3D point cloud representation of LiDAR scans as input and outputs, for each point, a semantic label and a temporally consistent instance ID over the whole sequence. To perform associations of detections over time, previous methods for multi-object tracking utilize feature similarity. They usually learn these features by providing a single positive and negative sample [28] or adding an extra positive [135] to increase temporal

consistency. We propose to leverage instance labels to learn descriptive features by providing several positive and negative samples to obtain temporally consistent instance features and perform the associations using similarity between these learnt features. To learn such instance-wise features, we use contrastive learning. However, different from contrastive learning for self-supervised representation-learning [19, 20, 141], where they use a data point as an anchor, positive samples are augmented views of it, and the negative samples are other data points in the batch, we leverage the available instance labels. We use the Supervised Contrastive Loss [67] and provide many negative and positive samples. The negatives are other instances in the batch, and the positives are the same anchor instance in other timesteps, which is different from self-supervised contrastive loss, where positives are generated through augmentations of the anchor. We build on top of a single-scan panoptic segmentation network to obtain the semantic and instance predictions as well as point-wise features. Our proposed contrastive aggregation network takes the point-wise features from the panoptic backbone as input to generate instance-wise embeddings that are similar over time. Finally, our association module combines appearance and motion cues to perform the instance associations and generate consistent instance IDs over time.

There are existing methods that perform segmentation and tracking simultaneously, like the architecture proposed by Hurtado et al. [62], which includes a tracking head to perform the instance associations across time. Similarly, we add instance associations to a panoptic segmentation network using the extracted features from the encoder-decoder, but we do not retrain it. This allows our method to leverage features from arbitrary panoptic backbones since it is not targeted at a particular architecture, and we furthermore leverage several positive and negative samples instead of just using a triplet loss [28].

In sum, we propose an approach that is able to perform 4D panoptic segmentation of LiDAR scans using a frozen panoptic segmentation backbone and associating instances across time via feature similarity. We achieve state-of-the-art performance in 4D panoptic segmentation even without sensor pose estimates used in previous approaches. The implementation of our approach is publicly available at `https://github.com/PRBonn/contrastive_association`.

4.1 Our Approach

The goal of the approach proposed in this chapter is to achieve 4D panoptic segmentation of LiDAR scans, which means tackling simultaneously semantic and instance segmentation in the *3D spatial* and *1D temporal* domain. We achieve this by predicting, for each 3D point in the LiDAR scan, a semantic label, and for each object, a temporally consistent instance ID. To address this task, we follow

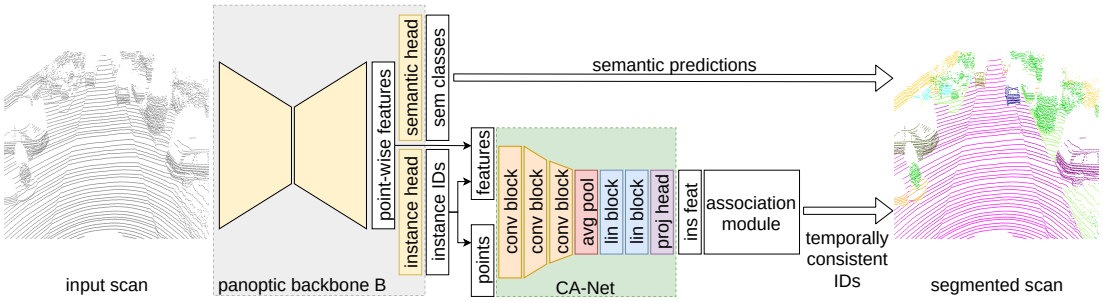


Figure 4.2: Overview of our method. Given the current 3D LiDAR scan, we use a panoptic segmentation backbone B to obtain semantic classes, instance IDs, and features for each point. We select the points and features for each instance using the IDs and input them into the CA-Net to obtain instance-wise features. These features are used to perform instance associations using our association module to assign temporally consistent instance IDs. Combining this with the semantic predictions, we obtain 4D panoptic segmentation.

prior work [62, 156] and use the tracking-by-detection paradigm [74] to identify objects in single 3D scans and then perform temporal associations via feature similarity.

Figure 4.2 gives an overview of our approach. We adopt the point cloud representation of LiDAR scans, which represents each point by its 3D coordinates. We use a frozen panoptic segmentation network (backbone), to obtain semantic predictions, instance predictions, and features for every point in the scan. Using the instance predictions, we select the points and features belonging to each instance and apply our contrastive aggregation network (CA-Net) to produce instance-wise features. Finally, our association module assigns instance IDs, which are consistent throughout the whole temporal sequence, by combining appearance information from the instance-wise features and motion information using a constant velocity motion model. Combining this with the semantic predictions, we obtain 4D panoptic segmentation.

To be able to learn descriptive features, the CA-Net is trained with the supervised contrastive loss [67]. Using the instance labels, we identify the same object over time and use the same instance in different scans as positive samples. The negative samples are all other instances in the same scan and other scans in the training batch.

4.1.1 Panoptic Segmentation Backbone

The first element of our approach is a frozen panoptic segmentation network to obtain semantic predictions, instance predictions, and point-wise features. To achieve panoptic segmentation, bottom-up methods [37, 51, 100, 176] use a shared

feature extractor to obtain point-wise features based on convolutions and apply task-specific heads to obtain semantic and instance features. The instance features are clustered in a post-processing step to produce instances.

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

In this chapter, we use DS-Net [51] as the backbone B. Since we are not modifying nor retraining the backbone and only use it to get predictions and features, our approach can potentially be applied to other panoptic segmentation networks.

4.1.2 Contrastive Aggregation Network

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

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

Due to the inherent sparsity of the data, we use sparse convolutions [25] to process the voxelized scan as it leads to better performance and reduces computational cost [41].

4.1.3 Association Module

After computing instance-wise embeddings $\mathcal{I}^F$ using our CA-Net, we maintain consistent instance-wise IDs by associating the instances $\{I_1 \dots I_M\}$ in the current scan with the corresponding instances $\{I_1 \dots I_K\}$ in the previous ones. To achieve this, we follow a similar procedure as previous works [7, 156]. We compute a cost matrix $\mathsf{C} \in \mathbb{R}^{M \times K}$ between all M instances in the current scan and all K instances identified in previous scans, by means of the similarity between their features. Then, the problem can be formulated as a bipartite graph matching problem that can be solved using the Hungarian method [74] to determine the pairs of associations between current and previous instances. The Hungarian method was designed for $M = K$, this means that it requires square cost matrices. However, we might have disappearing or newly appearing instances, which leads to a non-square matrix. To adapt this method, we add dummy rows or columns with zero cost to ensure that the matrix is square. After computing the assignments, we simply ignore the ones that include a dummy row or column in the pair because they do not belong to the original data.

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

To add information about the movement of the targets, we apply a constant velocity assumption for all objects by adding a constant velocity motion model independent of the sensor ego-motion. This means that the model does not estimate the ego-motion but the motion of all the objects in the scene. We fuse the appearance and motion information in the cost matrix as a linear combination of the feature cost and the center distance between current instances and the predicted positions of the previous instances. Each entry of the cost matrix C is an association cost between the new instance m and the previous instance k:

$$\mathsf{C}_{m,k} = \alpha_f \cdot (1 - sim(\boldsymbol{f}_m, \boldsymbol{f}_k)) + \alpha_d \cdot \|\boldsymbol{c}_m - \boldsymbol{c}_k\|, \tag{4.1}$$

where $\boldsymbol{f}_m$ is the feature and $\boldsymbol{c}_m \in \mathbb{R}^3$ the center coordinates of instance m, $sim(\cdot)$ is the cosine similarity function $sim(\boldsymbol{f}_m, \boldsymbol{f}_k) = \boldsymbol{f}_m^\top \boldsymbol{f}_k / (\|\boldsymbol{f}_m\| \|\boldsymbol{f}_k\|)$, and α_f, $\alpha_d \in \mathbb{R}$ are importance weights for the individual feature and distance costs. For re-identification, the motion model is applied continuously also to the deactivated

targets (not associated with any other instance) to estimate their position in case of occlusions or missing detections.

4.1.4 Input Features

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

Inspired by the recent findings in natural language processing [142], we seek to enhance the point-wise features with spatial knowledge using positional encodings. We compute for each instance point $\boldsymbol{p}_l \in \mathcal{I}_j^P$ a fixed positional encoding $\boldsymbol{e}_l \in \mathbb{R}^{D_B}$. We generate for each point a Fourier feature position encoding [64] with the same dimension as the point-wise features. We select a maximum frequency $\max_{\text{freq}}$ and sample, for each coordinate, from a series of sine and cosine functions with frequencies evenly spaced in $[0, \max_{\text{freq}}]$ on a logarithmic scale. Then, we use a padding of zeros to reach the same dimension as the point feature. To get the final input to the network, the feature and the positional encoding are combined by performing the addition of both. Due to their similarity, using the point-wise features as the only input to our CA-Net does not provide enough information to learn distinct instance-wise features to make the associations. By adding the positional encodings as extra information, we exploit the spatial relations between the instances and create more distinct features, and help the network learn better instance-wise features for this task.

4.1.5 Instance Point Extraction

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

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

The problem of split instances is due to the imperfect clustering applied in the backbone to obtain the instance predictions. It can wrongly cluster the points into instances, dividing one instance into two smaller ones, as shown in Figure 4.3. To mimic this, we create an augmentation, which we call *split*, to separate the points on each side of an imaginary plane to split the instance. We sample a random unit normal vector $\boldsymbol{n} \in \mathbb{R}^3, \|\mathbf{n}\| = 1$ and a random instance point $\boldsymbol{p}_l \in \mathcal{I}_j^P$ to generate a randomly oriented plane with Hessian normal form $\boldsymbol{n}^\top \boldsymbol{p}_l = d$, where d is the distance from the plane to the origin. The remaining points after the augmentation are the query points $\boldsymbol{q}_l$, which lie in the half-space of the normal:

$$\mathcal{I}_{aug} = \left\{ \boldsymbol{q}_l \in \mathcal{I}_j^P \mid \boldsymbol{n}^\top \boldsymbol{q}_l^C - d > 0 \right\}, \tag{4.2}$$

which leads to split instances.

The incomplete instances are caused by the wrong semantic class predicted for some instance points, usually at the borders of the instances. Their semantic class assigns them to the background, and they are not considered as part of the instance, as can be seen in Figure 4.3. To address it, we create an augmentation, which we call *contour*, to discard points in the contour of the instances. We first normalize the point coordinates to the range $[-1, 1]$, generate a random maximum coordinate value $\gamma \in \mathbb{R}$ and save the indices of the points with smaller absolute coordinates that γ. We use these indices to select the instance points (with unnormalized coordinates) to keep after the augmentation, i.e.,

$$\mathcal{I}_{aug} = \left\{ \boldsymbol{p}_l \in \mathcal{I}_j^P \mid |\tilde{x}_l| < \gamma \wedge |\tilde{y}_l| < \gamma \wedge |\tilde{z}_l| < \gamma \right\}, \tag{4.3}$$

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

We illustrate our augmentations applied to instances in Figure 4.3. It is important to notice that the augmentations do not change the position of any point but rather drop some of them. As an extra augmentation, we randomly drop cuboids of points belonging to the instance [167]. We discuss the influence of these augmentations in Section 4.3.3.

4.1.6 Pose Information

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

By adding the sensor pose estimates using a simultaneous localization and mapping (SLAM) approach [6], we can improve the instance associations by leveraging this extra information. We use the pose estimates to transform the

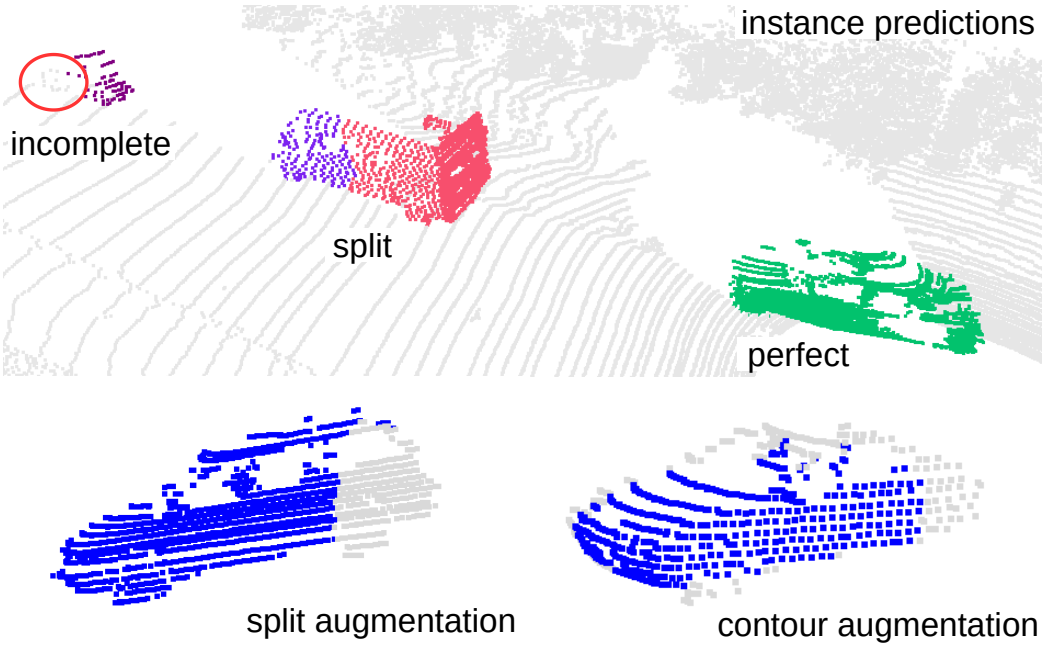


Figure 4.3: Problems in the instance predictions using the backbone. Incomplete, split, and perfect instance (top). Proposed *split* augmentation (left) and *contour* augmentation (right). The gray points are discarded.

positions of previously observed objects into the current local coordinate frame. This way, objects have consistent positions over time, and the constant velocity motion model only predicts the motion of the other objects with respect to the ego-motion. We recompute the features of the previously observed instances by updating their positional encoding using the consistent positions in the current coordinate frame. We obtain more similar features for the same instance at different points in time, which improves the instance associations as illustrated in Section 4.2.3

4.1.7 Contrastive Training

For the multi-object tracking problem, we use the instance head to get the predicted instances $\{\mathcal{I}_1, \dots, \mathcal{I}_M\}$ in the current point cloud $\mathcal{P}$ and associate them with previous instances across time. Several works [28, 49, 62, 135] use metric learning to learn instance representations by comparing the embedding of one anchor object with one or a few positive and negative samples. Contrastive learning is used in self-supervised representation learning [20, 141] to train a network, which is later fine-tuned on a downstream task. The main idea of those approaches is to use data augmentation to generate two versions of one anchor sample and train the network to learn to pull the embeddings of these samples (positives) together and push them away from all other samples (negatives).

Given a batch of samples $\mathcal{B}$, the anchor) and the negatives (augmented versions of different anchors). In the self-supervised contrastive loss InfoNCE [141], an encoder is used to obtain the feature vectors $\boldsymbol{z}_j$ for each augmented sample j. Let $r(j)$ be the index of the other augmented sample from the same anchor j and $\mathcal{A}(j)$ the set of all indices in the batch except j. Then, the loss function takes the form:

$$\mathcal{L}_{self} = -\sum_{j \in \mathcal{B}} \log \frac{\exp\left(\boldsymbol{z}_j^\top \boldsymbol{z}_{r(j)}/\tau\right)}{\sum\limits_{a \in \mathcal{A}(j)} \exp\left(\boldsymbol{z}_j^\top \boldsymbol{z}_a/\tau\right)}, \tag{4.4}$$

where τ is a temperature parameter.

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

The appearance can, however, change significantly in scans temporally far from each other. We define a temporal window of scans $\Delta \in \mathbb{N}$, in which we consider instances similar enough to perform the associations and from which the positive samples are drawn. As Δ is the number of scans from which we extract instances, it is the maximum number of positive samples to consider in the loss function. Using the labels, we select a set of positive samples $\mathcal{M}(j)$ considering the instances with the same ID in consecutive scans in the temporal window. As negative samples, we use all other instances in the batch, as depicted in Figure 4.4. To learn from many positive in addition to the many negative samples, we use the supervised contrastive loss [67]:

$$\mathcal{L}_{sup} = -\sum_{j \in \mathcal{B}} \frac{1}{|\mathcal{M}(j)|} \sum_{p \in \mathcal{M}(j)} \log \frac{\exp\left(\boldsymbol{f}_j^\top \boldsymbol{f}_p/\tau\right)}{\sum\limits_{a \in \mathcal{A}(j)} \exp\left(\boldsymbol{f}_j^\top \boldsymbol{f}_a/\tau\right)}, \tag{4.5}$$

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

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

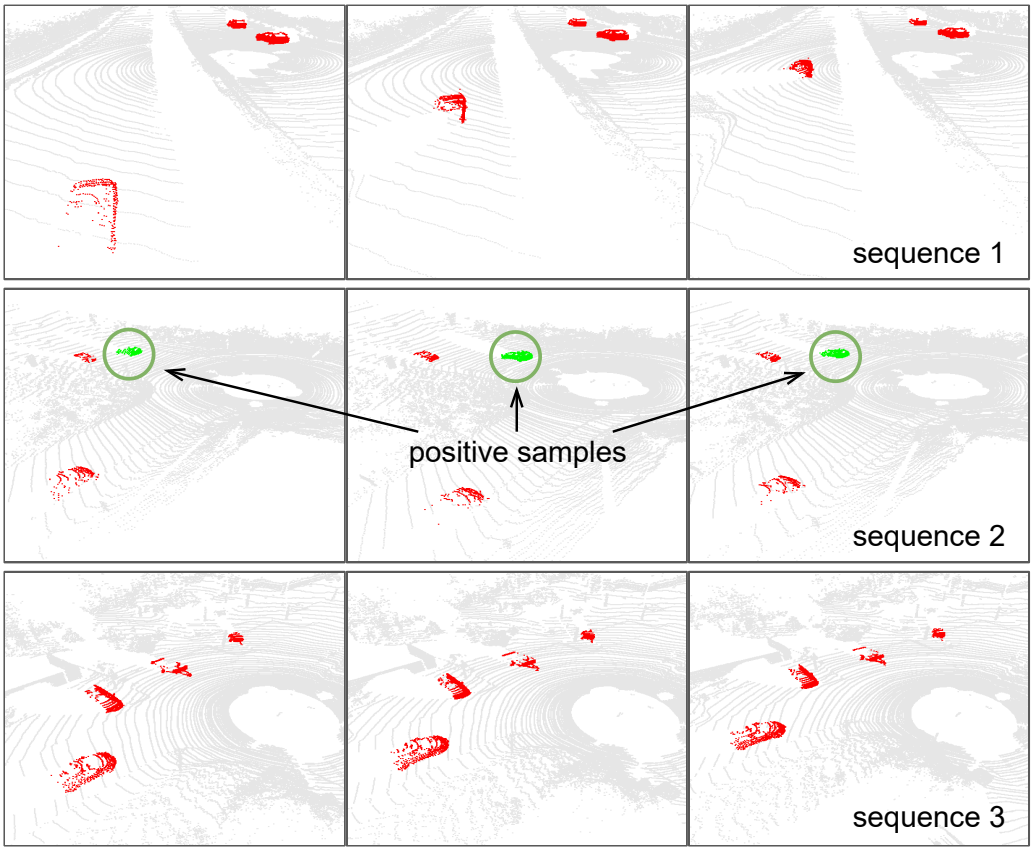


Figure 4.4: In our contrastive setup, for each instance, positive samples are the same instance in different scans (green) in a sequence, and negative samples are all the other instances (red) in all scans and all other sequences in the batch.

4.2 Experimental Evaluation

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

4.2.1 Experimental Setup

We evaluate our method on SemanticKITTI [3, 4], which consists of 22 sequences from the KITTI odometry dataset [38]. Sequences 00 to 10 are used for training, leaving sequence 08 as the validation set, and the test set consists of sequences 11 to 21. The provided annotations are point-wise semantic and instance la-

Method	LSTQ	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}
RangeNet++[101] + PP + MOT	35.5	24.1	52.4	64.5	35.8
KPConv [138] + PP + MOT	38.0	25.9	55.9	**66.9**	47.7
RangeNet++[101] + PP + SFP	34.9	23.3	52.4	64.5	35.8
KPConv [138] + PP + SFP	38.5	26.6	55.9	**66.9**	47.7
4DPLS[2]	56.9	56.4	57.4	66.9	51.6
CA-Net (ours) (without pose estimates)	60.0	59.5	**60.6**	66.9	**52.0**
CA-Net (ours) (with pose estimates [6])	**63.1**	**65.7**	**60.6**	66.9	**52.0**

Table 4.1: 4D panoptic segmentation on SemanticKITTI test set. All metrics reported in percent (%). MOT [156]; SFP - scene flow based propagation [103]; PP - PointPillars [76]. Note that adding pose information to our single-scan panoptic backbone does not influence the segmentation performance.

bels [5]. To evaluate our performance, we use the LiDAR Segmentation and Tracking Quality (LTSQ) metric $LSTQ = \sqrt{S_{\text{cls}} \cdot S_{\text{assoc}}}$, see [2] for details. This metric consists of two terms, the classification score S_{cls}, which measures how well each LiDAR point is labeled semantically, and the association score S_{assoc}, which evaluates how accurately points are assigned to the correct instance identity over time without semantic influence. Since our panoptic segmentation backbone is frozen and we use its single-scan predictions, our results do not change the segmentation results represented by the S_{cls} term. Thus, we focus instead on the S_{assoc} term to evaluate the quality of our associations.

4.2.2 Implementation Details

We build on top of DS-Net [51] as a panoptic segmentation backbone. The feature extractor provides point-wise feature vectors $\boldsymbol{p}_i^F \in \mathbb{R}^{D_B}, D_B = 128$. For the instance clustering, we use mean shift [26].

For our CA-Net, the convolutional blocks are 3D sparse convolutions [25], followed by a batch normalization layer [63] and leaky rectified linear unit (ReLU) [94] as activation layer. The sparse linear blocks consist of a linear layer followed by a batch normalization layer and a leaky ReLU as activation layer. The final projection head is a linear layer that projects the intermediate instance-wise embeddings into a feature space of dimension $D_A = 1024$ in which we use feature similarities to associate instances across time.

We keep deactivated targets for $n_{old} = 8$ frames and use different thresholds to handle new targets and re-identification for the appearance and the motion model. For the feature similarity, we use $T_{new_f} = T_{old_f} = 0.7$ and for the center distance, we use $T_{new_d} = T_{old_d} = 2$. The weights for the feature cost and the center

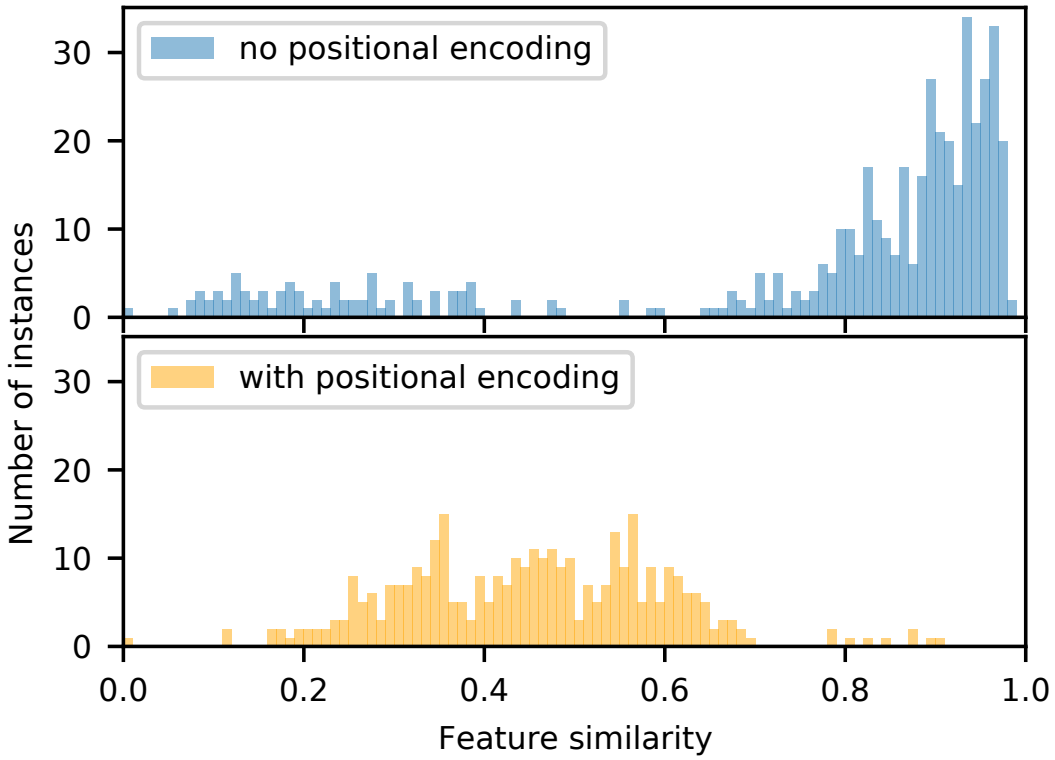


Figure 4.5: Histograms of similarities between instances' features in a sequence of scans. Without positional encoding (top), the histogram shows a high density at high similarities. Hence, there is a large number of instances with similar features, which makes it difficult to identify the same instance in different scans, as different instances in the sequence have similar features. With the positional encoding (bottom), most of the feature similarities are in the middle range, and only a few features have high similarity. Therefore, features from different instances are more distinct, which makes them better suited for associating instances.

distance in Equation (4.1) are $\alpha_f = 0.4$ and $\alpha_d = 0.7$. In the loss function, see Equation (4.5) we use $\tau = 0.1$. At each step, the positive and negative samples are selected from a sequence of scans of random length $\Delta \in [2, 5]$. This way, at each step, the loss considers samples from sequences of variable length, which provides a different number of instances.

4.2.3 4D Panoptic Segmentation Results

The first experiment evaluates the performance of our approach on the Semantic-KITTI test set and supports the claim that we achieve state-of-the-art results on the 4D panoptic segmentation task. In this experiment, we compare our approach with the recent and high-performing method by Aygun et al. [2] and baselines using semantic segmentation networks [101, 138] and combining the predictions from the PointPillars (PP) object detector [76] with a constant velocity motion

model [156] or using scene flow propagation (SFP) [103]. Note that the approach by Aygun et al. [2] and the baselines rely on the sensor poses to either aggregate scans or transform them into a global coordinate frame where objects have consistent positions and are easier to associate. For these approaches, this knowledge is a prerequisite.

We outperform all previous methods *without* relying on the pose estimations from a SLAM approach. See Table 4.1. Both experiments using our method show the same segmentation performance S_{cls}, IoU^{St}, and IoU^{Th} since we use the semantic predictions from the same single-scan backbone and the pose information does not modify the semantic segmentation results of *a single scan*. More detailed results are shown in Table 4.1.

4.3 Ablation Studies

4.3.1 Positional Encoding

First, we analyze the influence of adding the positional encoding to the point-wise features. We show the similarity between point-wise features from the backbone. To this end, we generate instance-wise features $\boldsymbol{f}_j$ by averaging the point-wise features for each instance:

$$\boldsymbol{f}_j = \frac{1}{|\mathcal{I}_j^F|} \sum_{\boldsymbol{p}_l \in I_j^P} \boldsymbol{p}_l^F, \tag{4.6}$$

and compute their similarities.

In a sequence of scans, there is a large number of highly similar instance features, which makes it hard to distinguish them, as shown Figure 4.5 (top).

To illustrate how the spatial encoding helps, we add the positional encoding generated from the point coordinates as explained in Section 4.1.4, to each point-wise feature and then average them to get an instance feature:

$$\boldsymbol{f}_j = \frac{1}{|\mathcal{I}_j^F|} \sum_{\boldsymbol{p}_l \in \mathcal{I}_j^P} \boldsymbol{p}_l^F + \boldsymbol{e}_l, \tag{4.7}$$

where $\boldsymbol{p}_l^F \in \mathbb{R}^{D_B}$ is the point-wise feature from the backbone and $\boldsymbol{e}_l \in \mathbb{R}^{D_B}$ is the positional encoding for the l^{th} instance point. As can be seen in Figure 4.5 (bottom), when adding the positional encoding, the amount of similar instance-wise features is visibly reduced. Thus, instances of different objects have a large distance in the feature space.

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

Table 4.2: Association performance S_{assoc} [%] using different instance-wise features for association in SemanticKITTI validation set.

4.3.2 Feature Design

In the next experiment, we have a closer look at our feature design decisions. We generate instance-wise feature vectors and associate the instances using only their feature similarity, without relying on any other information, to better show the performance. Table 4.2 shows the association performance of the different feature options, namely pooling the features and positional encodings and using them as input to the CA-Net to obtain the instance-wise features.

As reflected in the experiments, the information about the position of the instances plays a crucial role in the association process. However, the features from the backbone allow us to exploit some extra information, but, as discussed in Section 4.3.2, they are similar for objects of the same semantic class, and adding a positional encoding improves the performance. Lastly, adding the positional encoding to each point $\boldsymbol{p}_l + \boldsymbol{e}_l$, we obtain even better results. This shows that both parts are important and that leveraging the point-wise features from the backbone improves the instance association performance.

4.3.3 Instance Augmentations

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

As shown in Table 4.3, predictions only helps to increase S_{assoc} by 0.3 percent points. Each individual augmentation outperforms training with the labels or the predictions. Particularly, the *split* augmentation has the strongest influence. Combining all three augmentations results in a 13.7 percent points improvement compared to the training using only the labels.

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

Table 4.3: Influence of the selection of input points and the different augmentations in the association quality S_{assoc}.

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

Table 4.4: Influence of the different components of the approach in S_{assoc} on SemanticKITTI validation set.

4.3.4 Method Components

Table 4.4 shows the influence of different components on on the performance in terms of S_{assoc} on the validation set and label each combination as (A) to (F).

In (A), we only use the point-wise features from the backbone as input to our CA-Net and perform the associations via instance feature similarity. Since no spatial information is included, instances at different positions in the scan can be wrongly associated. In (B), we add the positional encoding to the input features, see Section 4.1.4, which adds spatial information, visibly improving the results. In (C), we add the SLAM poses [6] to update the previous instance positional encoding, see Section 4.1.6. This way, objects have consistent positions and are easier to associate. In (D), we use only the constant velocity motion model to perform the associations. This result is better than (A) and shows that motion and appearance models are necessary to achieve good association results. In

(E), we add the motion model to (B). We combine cues from the appearance and motion models, improving the results obtained when the models are used separately. In (F), we add the sensor ego-motion poses estimates and obtain the best performance.

4.3.5 Length of the Temporal Window

By defining the temporal window Δ, we change the number of scans in the batch and with it, the number of instances to consider. This way, we chose the number of positive samples considered in the loss function and the number of negatives in the batch. In Figure 4.4, for example, the window length is 3, i.e., scans recorded typically within 30 ms.

The influence of the length of the sequence in the performance is shown in Table 4.5, where the average number of instances per batch is also listed. To better analyse the influence of the temporal window, in this experiment, we remove the motion model and only use the feature similarity to associate the instances.

Increasing the length of the window increases the total number of instances and, with it, the number of negative and positive samples. As discussed by Chen et al. [20], contrastive learning benefits from more negative samples, facilitating convergence, and, according to Khosla et al. [67], the number of positive samples also helps to learn a better feature representation. This behaviour is observed when the length of the window increases from 2 to 5 scans.

Wang et al. [151] study the effects of the number of samples in contrastive learning. They find that providing only a few samples leads to under-clustering. This means that the model cannot efficiently learn to discover the dissimilarity between samples, and the generated embedding clusters are insufficient for the model to separate objects from different classes. On the contrary, an excessive number of samples leads to over-clustering. In this case, the model cannot efficiently learn the feature representation and separates objects of the same class into different clusters.

The increase of the window length further than 5 scans does not lead to much improvement, and the performance even drops with a window of 9 scans. We hypothesize that this is due to the over-clustering effect. To overcome this, we use a random window length between $\Delta \in [2, 5]$. Then, in the same batch, some sequences are long and provide a large number of instances, and some others provide a small number of positive and negative samples. This way, we compensate and avoid both undesirable behaviours, reaching a more balanced distribution of samples and improving the performance even more than when using a larger window length.

window lenght	number of instances	S_{assoc} [%]
2	75	66
3	115	68.4
5	190	69.6
7	270	69.9
9	340	69.5
random	95	**71.2**

Table 4.5: Influence of the lenght of temporal window Δ in the S_{assoc} in the validation set. Random means a random value between 2 and 5.

4.4 Conclusion

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

However, the method presented in this chapter requires hand-tuning some hyperparameters for the clustering algorithm to obtain instance segmentation, and furthermore, the fusion between appearance and motion information is sensitive to the importance weights α_d and α_f to build the cost matrix C to associate new and previous instances. In the current setup, our approach is not able to learn the clustering and association end-to-end from the data because those are non-differentiable operations.

Instead of relying on hand-tuning hyperparameters for the instance segmentation and association, we would like to train our model end-to-end and learn from the data how to segment instances and how to fuse appearance and motion information to associate them over time. This would allow our model to potentially scale better to new data since it would be able to adapt to it without hand-tuning

parameters. The first step to achieve this is removing the clustering algorithm and obtaining end-to-end 3D panoptic segmentation, which we investigate in the next chapter.

Chapter 5

Mask-Based 3D Panoptic Segmentation

As we argued before, semantic scene understanding is a key requirement for autonomous vehicles navigating in dynamic environments, since they need to understand their surroundings geometrically and semantically to plan and act appropriately. Panoptic segmentation [69] provides a description of the surroundings by jointly tackling semantic segmentation to semantically interpret the surroundings and instance segmentation to identify object instances. On LiDAR point clouds, panoptic segmentation is usually solved in a bottom-up manner, consisting of multiple steps: predicting the semantic class for each 3D point, using this information to filter out "stuff" points, and clustering "thing" points to obtain instance segmentation. Finally, the semantic and instance predictions are merged together using majority voting to solve conflicts between predictions. These methods address panoptic segmentation in multiple separate steps and do not optimize for the task to tackle, but rather for separate tasks. Usually, one branch is optimized for semantic segmentation, and instance segmentation is done by optimizing for a proxy task like predicting offset vectors [51], embedding vectors [100], and/or object centers [2]- and later gsrouping the points into instances using an off-the-shelf clustering algorithm [15, 26].

Furthermore, this clustering is a post-processing step with associated hyperparameters, which are usually hand-tuned and do not adapt to instances of different sizes, like cars and pedestrians or different datasets. As mentioned in Section 4.4, we aim to use neural networks to learn the segmentation and association of LiDAR point clouds, leveraging the available data instead of relying on heuristics or the hand-tuning of hyperparameters. In this context, the clustering algorithm usually adopted for instance segmentation is not differentiable and does not allow for end-to-end training. When analysing recent works on image

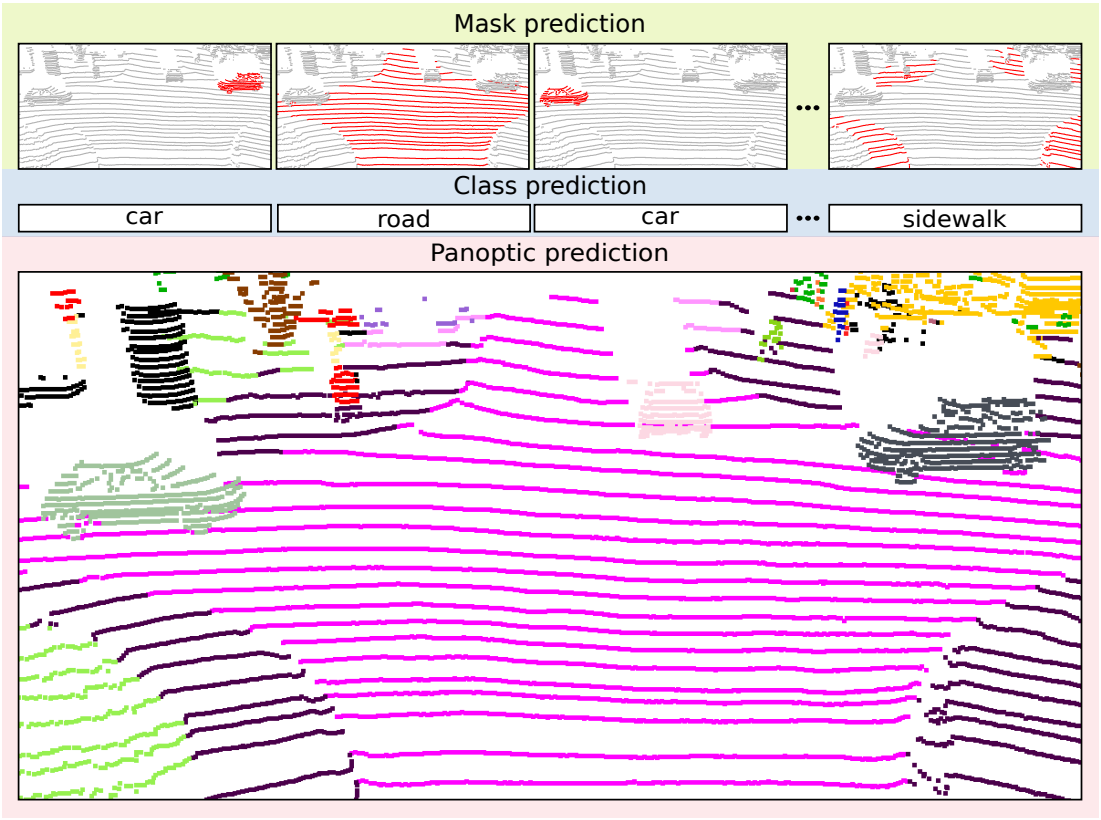


Figure 5.1: Panoptic segmentation by predicting binary masks and their corresponding semantic class. Each mask represents an individual instance of a "thing" class, i.e., car, or a complete "stuff" class, i.e., road, sidewalk. Top: predicted binary masks. Middle: predicted mask class. Bottom: panoptic prediction by merging the masks.

segmentation [22, 23, 147], there is a trend to tackle this problem by directly predicting a set of binary masks and semantic classes without relying on post-processing steps. However, this paradigm has not yet been applied to outdoor 3D LiDAR point clouds, where the distance-dependent sparsity and large number of points might pose a challenge for the aforementioned approaches.

In this chapter, we propose to remove the clustering step and perform panoptic segmentation on 3D LiDAR point clouds by directly predicting a set of binary masks and their corresponding semantic classes. The method is end-to-end trainable and does not require hand-tuning hyperparameters to work on data from different sensors. It is able to learn from the data how to segment instances of different shapes and sizes. Directly optimizing for panoptic segmentation in an end-to-end manner allows our method to be potentially more scalable since the addition of more data or a modification in the training dataset does not require manually adjusting parameters. The method would learn from this modified version of the dataset and leverage the existing data to learn and adapt to the different instances provided during training.

In contrast to our proposed approach, recent methods for panoptic segmentation [51, 79, 80, 100, 124] are bottom-up approaches and rely on clustering algorithms to group points into instances. Seeking to automatically adjust the clustering step to instances of different sizes, Hong et al. [51] propose a dynamic shifting module that adapts kernel functions on the fly. Li et al. [80] introduce a clustering pseudo heatmap paradigm where they shift the points with predicted offsets and project them to a BEV image to create a heatmap where they perform the clustering. Still, these methods require clustering as a post-processing step, which makes them not end-to-end trainable. To make the approach end-to-end trainable, Gasperini et al. [37] propose a learning-based clustering method and optimize two complementary losses based on a confusion matrix but require several extra post-processing steps to reduce under- and over-segmentation caused by the clustering and achieve competitive performance. Li et al. [78] propose a cluster-free instance segmentation head by pillarizing foreground point embeddings and finding connected components via pairwise embedding comparison. They manage to remove the clustering algorithm but achieve a lower performance compared to state-of-the-art methods.

Inspired by recent works operating on images [18, 23], we adopt a backbone network for feature extraction, a set of learnable feature vectors as mask proposals, and a transformer decoder to predict non-overlapping binary masks and their semantic classes, like depicted in Figure 5.1. Each mask represents an individual instance of a "thing" class or a complete "stuff" class. We base our approach on the work by Cheng et al. [22], which deals with image data only. Compared to images, we have to deal with the sparsity of the LiDAR data while avoiding the high memory requirements of 3D convolutions. Therefore, we use a backbone based on 3D sparse convolutions to extract multi-scale features. A modified transformer decoder combines learnable queries and features from different resolutions through masked attention [22]. We generate the final 3D masks by combining the output queries of the decoder and the full resolution point features. To deal with the voxelization effects, we compute point-wise features through interpolation instead of directly feeding voxel features to the decoder. To improve the quality of 3D masks, we merge semantic and spatial information by adding fixed positional encodings to full-resolution point features.

The main contribution of this chapter is a method to perform panoptic segmentation on 3D LiDAR point clouds by directly predicting a set of binary masks and their corresponding semantic classes that can be trained end-to-end without any heuristic post-processing steps, like clustering. To the best of our knowledge, we present the first approach for mask-based panoptic segmentation in 3D point clouds. Our experiments suggest that our approach (i) performs better than common approaches relying on clustering algorithms that might need hyperparameter

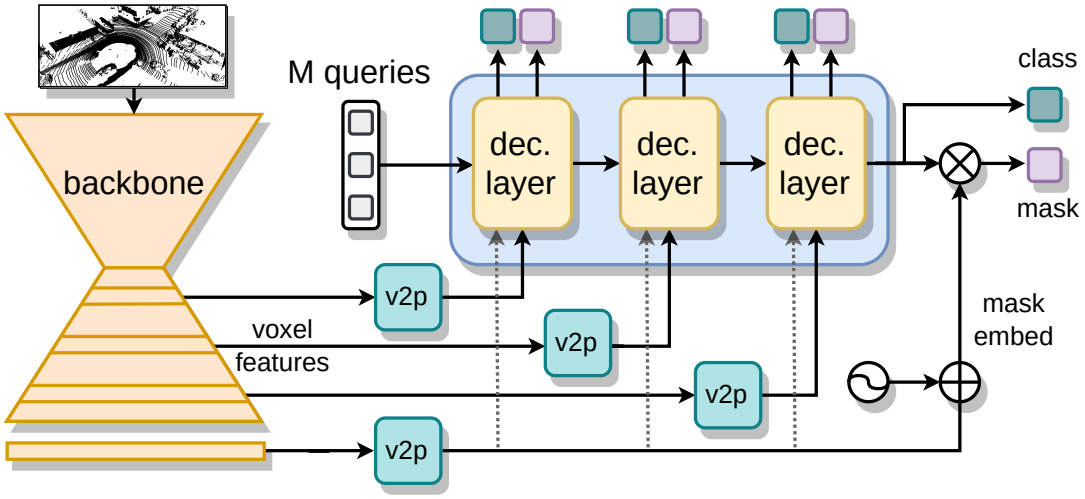


Figure 5.2: Overview of our method. The 3D feature extractor processes the input point cloud to obtain multi-resolution features. The voxel-to-point operation (v2p) transforms voxel features into point features. The learnable queries and point features are inputs to the decoder layers. The predicted masks are obtained via the dot product between mask embeddings and output queries. The class prediction is decoded from the output query embeddings.

tuning for different datasets; and (ii) achieves competitive performance even with a simple feature extractor. The implementation of our approach is available at `https://github.com/PRBonn/MaskPLS`.

5.1 Mask-based Panoptic LiDAR Segmentation

5.1.1 Task Definition

Given a point cloud $\mathcal{P} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_N\}$ consistint of N points with coordinates $\boldsymbol{p}_i \in \mathbb{R}^3$, we represent it as a matrix

$$P = \begin{bmatrix} \boldsymbol{p}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{p}_N^\mathsf{T} \end{bmatrix}, \tag{5.1}$$

where each row i represents a point $\boldsymbol{p}_i \in \mathbb{R}^3$. Our objective is to segment P into a set $\mathcal{Y} = \{(\boldsymbol{m}_i, c_i)\}_{i=1}^K$ of K non-overlapping binary masks $\boldsymbol{m}_i \in \{0,1\}^N$ and their semantic classes $c_i \in \{1, \dots, C\}$ for C classes. Each mask represents either a single instance of a "thing" class or a complete "stuff" class.

Our model predicts a set $\mathcal{Z} = \{(\hat{\boldsymbol{m}}_i, \hat{p}_i)\}_{i=1}^M$ of M binary masks predictions $\hat{\boldsymbol{m}}_i \in [0,1]^N$ with values between 0 and 1; and probability distributions $\hat{\boldsymbol{p}}_i$ over the classes $\{1, \dots, C\} \cup \{\varnothing\}$, where $\varnothing$ is an additional "no object" class. Note that

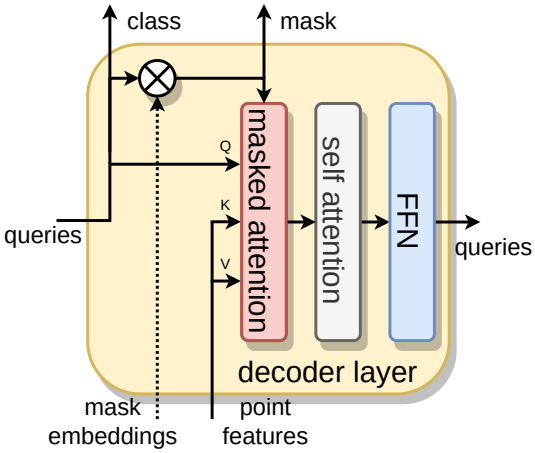


Figure 5.3: Each decoder layer consists of masked attention between queries and mask embeddings, followed by self-attention and a feedforward network (FFN). Apart from refining the queries, each layer outputs an intermediate binary mask to use in the following layer. The intermediate predictions are only used for training and are supervised with the same loss function.

we assume $M > K$ and include a "no object" class that is assigned to the extra $M - K$ masks.

We follow prior works [18, 22, 23] and use an architecture consisting of a feature extractor to obtain multi-scale features and a transformer decoder combining learnable queries and backbone features to obtain the segmented output. Figure 5.2 shows an overview of our method. We voxelize the input point cloud and extract voxel features at different resolutions via the backbone. We convert voxel features into point features and feed these to the transformer decoder along with M learnable queries (learnable feature vectors). The final M mask predictions are obtained by combining the mask embeddings and the output query embeddings while the mask semantic classes are decoded from the output queries.

5.1.2 Feature Extractor

We use a feature extractor to obtain multi-scale features as input to the decoder. Given the sparse nature of the LiDAR data, we use a MinkowskiNet [25] backbone composed of 3D sparse convolutions to preserve spatial knowledge and obtain meaningful features with a low memory footprint. To show that our approach can be applied to other backbones, we also evaluate it with CylinderNet [178] using cylindrical voxels, and SPVCNN [137] using sparse point-voxel convolutions.

The backbone extracts voxel features $\boldsymbol{F}_v \in \mathbb{R}^{C_v \times N_v}$ at different resolutions, where C_v is the feature dimension and N_v is the number of voxels, which depends on feature stride, and full-resolution point features $\boldsymbol{F}_p \in \mathbb{R}^{C_p \times N}$, where C_p

is the feature dimension. We adopt multi-scale features to help segment small instances. To use these features as decoder input, we convert the voxel features into point features with the voxel-to-point function (v2p) $f : \mathbb{R}^{C_v \times N_v} \rightarrow \mathbb{R}^{C_v \times N}$ that we discuss in Section 5.1.6

5.1.3 Transformer Decoder

The decoder takes queries as mask proposals and multi-scale features and outputs the final queries used for the mask predictions. Following the work by Cheng et al. [22], we use a modified transformer decoder layer to leverage optimization and training efficiency improvements by using masked attention. Compared to the standard cross-attention, where the queries attend to all the keys, masked attention restricts the attention to a subset of keys localized inside a mask. The decoder layer is depicted in Figure 5.3 and consists of masked attention between queries and backbone features, followed by self-attention and a feedforward network (FFN). Each masked attention combines the M learnable queries with features from different resolutions to enhance the training and help the model segment small objects. We group three decoder layers into a transformer decoder block that we repeat L times. After the decoder blocks, we obtain the output query embeddings $\mathsf{Q} \in \mathbb{R}^{M \times C_P}$. We keep spatial reasoning during the decoding step, adding fixed point-wise positional encodings [142], expressed by $\mathsf{P}_e \in \mathbb{R}^{N \times C_p}$ to the point features input to each cross-attention module and to the full resolution features to obtain mask embeddings $\mathsf{E} = \mathsf{F}_p + \mathsf{P}_e$.

5.1.4 Mask Prediction

The final step is to compute the mask predictions using output queries and point features. We obtain the probability distributions $\hat{\boldsymbol{p}}_i$ over the semantic classes by applying a linear layer over the output queries. For the masks predictions, we want to obtain the mask scores $\mathsf{M} \in \mathbb{R}^{N \times M}$, i.e., the score of each point for each of the M predicted masks $\hat{\boldsymbol{m}}_i \in [0, 1]^N$, where $\hat{\boldsymbol{m}}_i$ is represented by the i^{th} column of M. We compute the mask scores via the dot product between mask embeddings and output queries, and apply a sigmoid activation function $\mathsf{M} = \text{sigmoid}(\mathsf{E}\mathsf{Q}^\top)$. We can obtain the final prediction by performing argmax twice: over the semantic class for each mask and over the mask scores for each point. In practice, we filter out masks with a large part of their binary mask occluded by other predictions to reduce false positives as done in previous works [18, 23].

Apart from the final predictions, we generate intermediate predictions to enhance the training by supervising them with the same loss function as for the final predictions. Before each decoder layer, we use the mask embeddings E and the queries of the previous step to get the intermediate predictions as shown in

Figure 5.3. The masked attention function requires a binary mask to allow the queries to attend only to the point features inside it. We obtain them by applying a threshold $\tau = 0.5$ over the intermediate mask predictions. However, if all the points are masked out, we do not apply any mask and attend to all the points.

5.1.5 Loss Function

To train our approach, we follow MaskFormer [23] and perform a bipartite matching between the M predictions $\mathcal{Z}$ and the K ground-truth masks $\mathcal{Y}$ to supervise the training. As mentioned before, the number of predictions M is larger than the number of ground-truth masks K. We complete with "no object" class to obtain a one-to-one matching. We define an assignment cost matrix C for all pairs of prediction-ground-truth, taking into account the mask and the semantic class. Each element of the matrix takes the form:

$$C_{i,j} = -\hat{p}_j(c_i) + \mathcal{L}_{mask}, \tag{5.2}$$

where $\hat{p}_j(c_i)$ is the probability of class c_i and $\mathcal{L}_{mask}$ is the combination of binary cross entropy and dice loss:

$$\mathcal{L}_{mask} = \lambda_{\text{dice}}\text{Dice}(m_i, \hat{m}_j) + \lambda_{\text{ce}}\text{BCE}(m_i, \hat{m}_j). \tag{5.3}$$

We obtain the optimal matching given the fixed cost function using the Hungarian method [74], similar to the approach presented in the previous chapter. After the matching between ground-truth masks and predictions, we compute the loss function $\mathcal{L}_m$ over the K matched pairs. It consists of the mask loss $\mathcal{L}_{mask}$ for mask segmentation, and cross-entropy for the semantic class, as follows:

$$\mathcal{L}_m = \sum_{j=1}^{K} -\lambda_{\text{cls}} \log \hat{p}_j(c_i) + \mathcal{L}_{mask}. \tag{5.4}$$

For the $M - K$ unmatched masks, we want the model to predict the "no object" class. We only use the cross entropy loss that we down-weight by a factor $\alpha < 1$ to compensate for class imbalance since there is usually a majority of non-matched masks. The loss takes the form:

$$\mathcal{L}_n = \sum_{j=K+1}^{M} -\alpha \log \hat{p}_j(\emptyset), \tag{5.5}$$

and the final loss is the sum of both: $\mathcal{L} = \mathcal{L}_m + \mathcal{L}_n$.

Inspired by previous works [22, 70], we randomly sample a fixed number of S points to calculate the mask loss instead of using all the N points in the LiDAR scan to reduce computation. To improve the backbone features, we add an auxiliary loss to directly supervise them. We use a linear layer to convert from full-resolution point features to class probabilities for each point. We supervise the logits with cross-entropy at the point level.

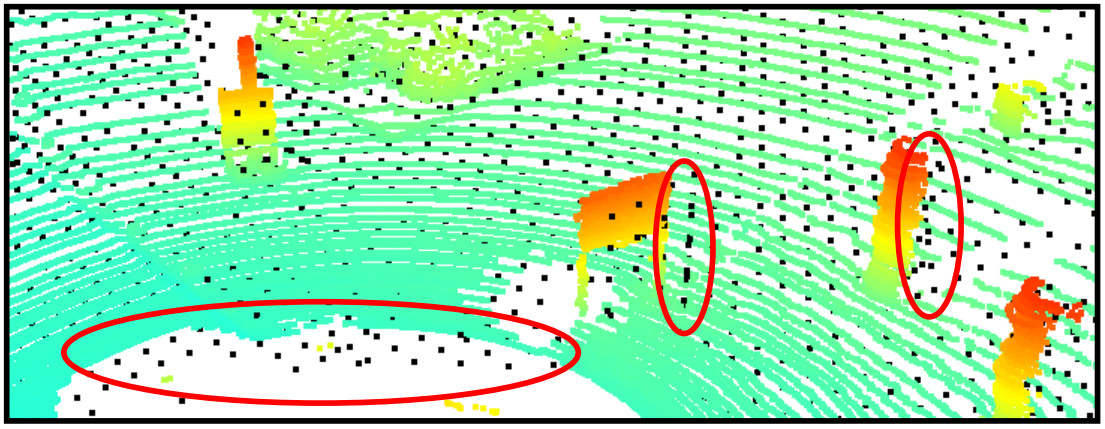


Figure 5.4: Downsamplig and voxelization effects. The original point cloud is colored by height. Black points show the result of voxelization and downsampling. We show artificial points of the downsampled version in regions with no points from the original point cloud.

5.1.6 Point-Level Multi-Scale Features

While using downsampled feature maps as input to transformer blocks works in the image domain, the voxelization of the real-world point clouds, and the downsampling operations can lead to an undesired behavior. The voxels in lower scales cover a bigger volume, but are represented just as a point located in the voxel center. Its coordinates represent artificial points that do not exist in the original point cloud. This can be seen in the Figure 5.4, where we show the original full point cloud colored with the point height and in black, the same point cloud but voxelized and downsampled by a factor of 4. The signaled points are located in regions in which there are no points in the original point cloud. Using the voxel features F_v in the masked attention harms the performance since we are using non-existing points. To account for this, we project the voxel features to the full point cloud with the voxel-to-point operation (v2p) and feed the resulting point features to the transformer decoder. The v2p function could consist of copying the same feature to all points inside the same voxel. To avoid having repeated features for all points inside each voxel, we perform a k-nearest-neighbors interpolation between voxel centers and the points of the full-resolution scan. We discuss the performances of these two v2p functions in Section 5.2.7.1.

5.2 Experimental Evaluation

The main focus of the work presented in this chapter is to perform panoptic segmentation in an end-to-end manner by removing the classic clustering step and directly predicting binary masks with their corresponding semantic classes. We

present our experiments to show the capabilities of our method and support our key claims, which are: (i) our approach performs better than common approaches relying on clustering algorithms that might need hyperparameter tuning for different datasets, and (ii) it achieves competitive performance even if using a simple feature extractor without any modification.

5.2.1 Experimental Setup

To evaluate our method, similar to Chapter 4, we use SemanticKITTI [3, 4], which provides point-wise semantic and instance annotations [5] for 22 sequences of the KITTI odometry dataset [38]. Sequences 00 to 10 are used for training, except for sequence 08, which serves as the validation set and sequences 11 to 21 for the test set. We run further experiments on nuScenes [14], which consists of 1000 sequences with 20 seconds of duration recorded with a 32-beam LiDAR sensor. The annotations are created every 0.5 s and include 16 semantic classes, including 10 "thing" classes.

We use panoptic quality (PQ), segmentation quality (SQ), and recognition quality (RQ) as metrics to evaluate panoptic segmentation following the task definition by Kirillov et al. [69]. This process requires first matching predicted and ground truth segments based on their intersection over union (IoU) to later compute the SQ and RQ terms. The segmentation quality assesses how well the predicted segments overlap with the ground truth spatially, while the recognition quality measures how well the model detects and classifies the correct number of instances. Panoptic quality is the multiplication of both SQ and RQ to measure how good the model is at both recognizing and precisely segmenting all "things" and "stuff" in the scene. Given that the semantic segmentation performance plays an important role in panoptic segmentation and in the panoptic quality, we also report the mean intersection over union (mIoU) across all semantic classes.

5.2.2 Implementation Details

We use $L = 3$ decoder blocks (9 decoder layers). In the loss function, we adopt $\lambda_{\text{dice}} = \lambda_{\text{ce}} = 5, \lambda_{\text{cls}} = 2$, $\alpha = 0.1$ and the number of randomly sampled points $S = 50000$. We set the number of learnable queries $M = 100$. We train the models for 100 epochs using AdamW [92] optimizer and step learning rate schedule. The initial learning rate is 0.0001, and we decay the learning rate by a factor of 10 at epoch 80. As augmentations, we apply random rotations, flips along the X or Y axis, and random scaling.

5.2.3 Comparison with Clustering as Post-Processing

This experiment supports claim (i) made in this chapter, and shows that our proposed method performs better than approaches relying on clustering algorithms. For that, we compare the performance of three backbones with different post-processing algorithms with our proposed end-to-end model. We build our models on top of three widely used 3D semantic segmentation networks as feature extractors without making any change, namely MinkNet [25], CylinderNet [178], and SPVCNN [137]. We see that our method works with potentially any given backbone, and using a better semantic segmentation network would help to achieve better panoptic segmentation results. We name our models MaskPLS-M (MinkNet), MaskPLS-C (CylinderNet), and MaskPLS-S (SPVCNN).

For the clustering approaches, we follow common practices [51] and add an instance head composed of three convolutions and two linear layers to predict, for each point of a "thing" class, an offset vector to the instance center. During training, we filter out the "stuff" points using the labels and semantic predictions during inference. As post-processing, we shift the points using the predicted offsets and use a clustering algorithm to group points into instances. We use HDBSCAN [15], mean shift [26], and the dynamic shifting module [51]. The results on the validation set of SemanticKITTI are presented in Table 5.1 and show that our method consistently improves RQ and PQ for all the different backbones and outperforms all clustering methods in terms of PQ while being end-to-end trainable. MaskPLS-M achieves a better PQ but a lower mIoU with respect to the baselines. The cause might be that, for the baselines, the backbone is trained only for semantic segmentation, while MaskPLS is trained for a different objective, which comprises both semantic and instance segmentation. Therefore, the balance between both tasks might harm the performance of a single one. In our experiments, we saw that even though the mIoU decreases, the SQ increases for most of the "thing" classes, making it easier to separate points into instances.

5.2.4 Panoptic Segmentation Performance

The next experiment evaluates the output predictions, and its results show that our approach achieves competitive performance by just using a simple feature extractor without any modification and thus supports claim (ii) made in this chapter. We compare the performance of our approach MaskPLS-M and previous works on the SemanticKITTI test set and show the results in Table 5.2. We achieve competitive performance using a simple ResNet-like model without any modification as feature extractor. Furthermore, we surpass previous end-to-end approaches like Panoster [37] and CPSeg [78]. We rank first on the leaderboard among open-source projects. As mentioned before, the semantic segmentation

Method	PQ	SQ	RQ	mIoU
MinkNet + HDBSCAN	57.9	79.8	67.6	63.4
MinkNet + MeanShift	58.3	79.8	68.0	63.4
MinkNet + DS	58.2	**80.4**	67.9	**63.5**
MaskPLS-M (ours)	**59.8**	76.3	**69.0**	61.9
CylinderNet + HDBSCAN	55.3	77.0	65.5	63.2
CylinderNet + MeanShift	56.4	76.5	67.1	**63.5**
CylinderNet + DS	57.7	**77.6**	68.0	**63.5**
MaskPLS-C (ours)	**58.9**	75.7	**68.4**	63.4
SPVCNN + HDBSCAN	52.5	**77.5**	62.9	60.0
SPVCNN + MeanShift	53.0	**77.5**	65.0	60.0
SPVCNN + DS	52.8	77.3	63.3	59.9
MaskPLS-S (ours)	**55.5**	75.0	**65.1**	**60.6**

Table 5.1: Panoptic segmentation results on SemanticKITTI validation set. All metrics are reported in percent (%).

performance plays an important role in panoptic segmentation and in the panoptic quality. The state-of-the-art approaches achieve better panoptic quality while relying on a backbone with a much better semantic segmentation performance. Our approach achieves a comparably good SQ, particularly for "things". This shows that it can better differentiate the detected segments, and this results in more accurate instance predictions. In our approach, the semantic segmentation performance heavily relies on the backbone performance. Potentially, a better backbone would improve our panoptic segmentation results.

5.2.5 Operation on a Different Dataset

With this experiment, we aim to show that we do not need to tune any hyperparameters to use our model in a different dataset and provide support for claim (i). We conduct a similar experiment to the one in Section 5.2.3 but on nuScenes. Table 5.3 shows a comparison between the performances of our approach and different clustering methods as post-processing step with MinkNet and CylinderNet as backbones on the nuScenes validation set. For a fair comparison, we retrain DS-Net [51] with the annotations provided by Fong et al. [32]. Our approach outperforms the baselines in terms of PQ, also in this dataset, recorded with a different LiDAR sensor. Furthermore, our approach adapts to this different dataset without tuning any hyperparameter, while in the case of the clustering algorithms, it is usually necessary to specify at least a bandwidth or the minimum cluster size. The performance of our approach does not match state-of-the-art

Method	PQ	PQ+	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}	mIoU
Panoster [37]	52.7	59.9	64.1	80.7	49.4	58.5	83.3	55.1	68.2	78.8	59.9
CPSeg [78]	57.0	63.5	68.8	82.2	55.1	64.1	86.1	58.4	72.3	79.3	62.7
EfficientLPS [132]	57.4	63.2	68.7	83.0	53.1	60.5	87.8	60.5	**74.6**	79.5	61.4
PVCL [90]	59.1	65.7	69.6	84.0	59.8	66.7	89.2	58.6	71.6	80.3	64.0
GP-S3Net [124]	60.0	**69.0**	**72.1**	82.0	**65.0**	**74.5**	86.6	56.4	70.4	78.7	**70.8**
SCAN [160]	**61.5**	67.5	**72.1**	84.5	61.4	69.3	88.1	**61.5**	74.1	**81.8**	67.7
Panoptic-PHNet [80]	**61.5**	67.9	**72.1**	**84.8**	63.8	70.4	**90.7**	59.9	73.3	80.5	66.0
MaskPLS-M (ours)	58.2	63.3	68.6	83.9	55.7	61.7	89.2	60.0	73.7	80.0	62.5

Table 5.2: Panoptic segmentation results on SemanticKITTI test set. All metrics are reported in percent (%).

Method	PQ	PQ+	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}	mIoU
EfficientLPS [132]	62.0	65.6	73.9	83.4	56.8	68.0	83.2	70.6	83.6	83.8	65.6
Panoptic-PolarNet [176]	63.4	67.2	75.3	83.9	59.2	70.3	84.1	70.4	83.5	83.6	66.9
Panoptic-PHNet [80]	**74.7**	**77.7**	**84.2**	**88.2**	**74.0**	**82.5**	**89.0**	**75.9**	**86.9**	**86.8**	**79.7**
MinkNet + HDBSCAN	52.6	57.4	63.2	69.0	59.1	71.2	81.2	47.9	57.4	60.2	61.6
MinkNet + MeanShift	54.4	58.9	64.8	69.5	62.6	74.0	82.6	48.3	58.1	60.0	61.6
MaskPLS-M (ours)	57.7	60.2	66.0	71.8	64.4	73.3	84.8	52.2	60.7	62.4	62.5
CylinderNet + DS	51.4	56.1	62.4	68.3	58.8	70.9	81.0	46.0	56.2	59.0	56.7
MaskPLS-C (ours)	57.4	59.8	66.2	71.8	63.6	72.7	84.6	52.9	61.4	62.5	60.2

Table 5.3: Panoptic segmentation results on nuScenes validation set. All metrics are reported in percent (%).

methods because we do not change any hyperparameters to re-train it on this different dataset. To improve the performance, some parameters should be changed to account for, for example, for the different LiDAR density in this new dataset.

In Figure 5.5, we depict examples of the instance segmentation performances with different clustering radii applied to the semantic predictions obtained by MaskPLS. We show that the clustering algorithms usually need a different radius for different classes and different datasets. We use MinkNet and mean shift, and compare it with our approach. First, we select a radius of 1.2 m for SemanticKITTI, and the instances are correctly segmented. However, on nuScenes, several pedestrians are clustered together. Second, we select a radius of 0.5 m for nuScenes, where the pedestrians are separated, but on SemanticKITTI, a vehicle is split into two instances. Our method, in contrast, correctly segments instances of different classes in both datasets despite the different sensors and without hyperparameter tuning.

5.2.6 Effect of Mask Embeddings

We generate the mask predictions via the dot product between point features and query embeddings. Thus, the quality of the masks relies on the quality of

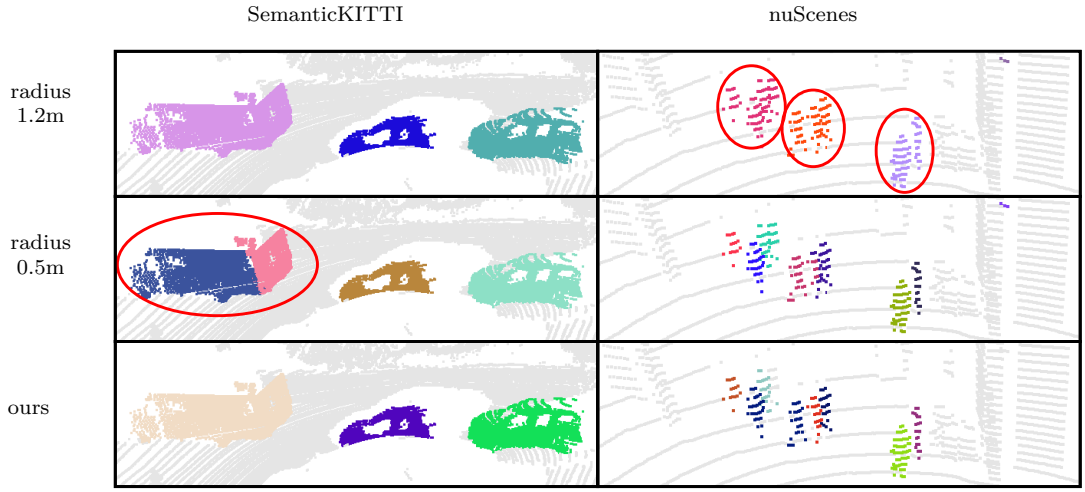


Figure 5.5: Comparison of instance segmentation using mean shift with different clustering radius and our approach. In the top row, we select a radius for SemanticKITTI and, when evaluating on nuScenes, several pedestrians are grouped as a single instance. In the middle row, we select a radius for nuScenes. The pedestrians are correctly separated, but a vehicle is split into two instances. In the bottom row, our approach correctly segments instances in both datasets without changing any hyperparameter.

the point features, which should contain both semantic and spatial information. This can be seen in Figure 5.6.a, where we show the cosine similarity between the mean feature of a car that we want to segment and the rest of the points. All the car points have a higher similarity than the background points, showing the semantic information in the point-wise features. Furthermore, the closer the points are to the desired car, the higher the similarity, depicting the spatial information. However, the obtained masks are not always accurate enough to segment a single instance. In our example in Figure 5.6.b, the predicted mask also includes a part of a second car.

We argue that the problem is caused by using the backbone features F_p to generate the masks, i.e., as mask embeddings E. In this configuration, the point features have to account for spatial and semantic information at the same time. We propose to divide the mask embeddings into spatial and semantic embeddings to relax the dependency of the point-wise features with respect to the spatial information. We use point-wise fixed positional encodings P_e [142] to capture the spatial information. For the semantic information, we use the output features from the backbone F_p. This way, the backbone features do not have to account for spatial information and only focus on accurate semantic information to better describe the scene. The final mask embeddings are the sum of both: $E = F_p + P_e$. In Figure 5.6.c, we show the feature similarity between the mean feature of the desired car and all the other points after this change. The similarity is now higher for most car points, showing that the features are mainly focusing on semantic

#	sem loss	point feat.	interp.	pos enc	PQ [%]	runtime [ms]
A	-	-	-	-	51.9	390
B	✓	-	-	-	52.3	390
C	✓	✓	-	-	52.7	**283**
D	✓	✓	✓	-	53.4	313
E	✓	✓	✓	✓	**54.9**	315

Table 5.4: Influence of the design choices in panoptic quality and inference time in the validation set of SemanticKITTI.

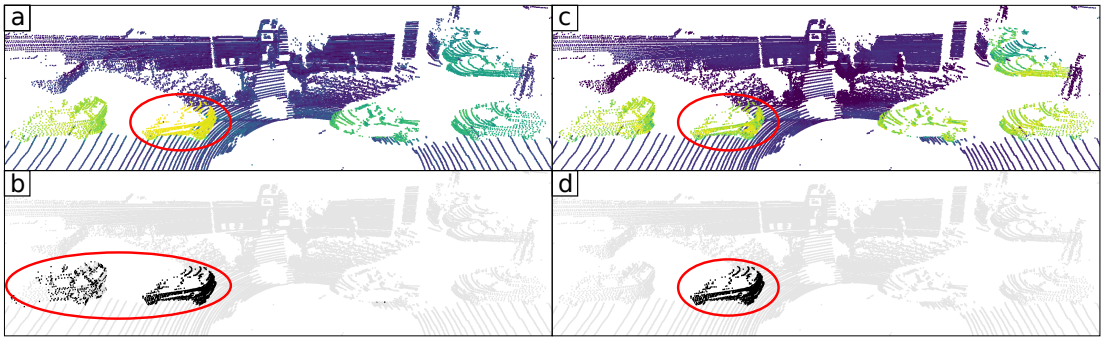


Figure 5.6: Influence of the proposed mask embeddings: (a) is the cosine similarity between the mean feature of the circled car and all the other point features. The point features provide semantic and spatial information: all car points have high similarity, and nearby points have more similar features. The generated mask (b) includes part of a second car. Splitting the mask embeddings, the features from the backbone focus mainly on semantic information, giving all cars similar features (c), while the spatial information is given by the positional encodings. As a result, the mask can accurately segment a single car (d).

information. We add the positional encodings to obtain our mask embedding, and the generated masks better capture single instances, as depicted in Figure 5.6.d.

5.2.7 Ablation Studies

We perform experiments to show the influence of each design choice and the number of decoder blocks. To reduce the total wall-clock training time, we perform the experiments in a smaller model by reducing the feature dimensions of the backbone blocks and the decoder blocks by a factor of 4. We use the same training schedule as in Section 5.2.4 except that we train the models with a fixed learning rate for 50 epochs. We show the results on the SemanticKITTI validation set.

5.2.7.1 Design Choices

In Table 5.4, we show the influence of each design choice on the final performance and runtime. In experiment (A), we follow Mask2Former [22] but using a feature extractor with 3D sparse convolutions and voxel features, achieving a panoptic quality of 51.9%. In (B), we add the auxiliary semantic segmentation loss on the full resolution point features. We seek to get more distinctive features, which are used to generate the mask predictions, and the overall performance highly relies on their quality. We boost the performance by 0.4 percent points without affecting runtime. In (C), we use point features instead of voxel features by copying voxel features to all the points within that voxel as v2p function and obtain a gain of 0.4 percent points, and the runtime decreases to 283 ms. The reason is that we do not need to interpolate the masks for the masked-attention since we are using full-resolution masks and features. In (D), we perform kNN interpolation between voxel features and point features. Each point feature is the weighted sum of the k nearest voxels. We use $k = 3$ neighbors and improve 0.7 percent points while increasing runtime to 313 ms. Finally, in experiment (E), we add the positional encoding to the full-scale point features to obtain the so-called mask embeddings and generate the mask predictions with them. This boosts the performance by 1.5 percent points to the final PQ of 54.9% and increases the inference time to 315 ms.

5.2.7.2 Number of Decoder Blocks

We show the influence of the number of decoder blocks in Table 5.5. Using one block, we achieve a panoptic quality of 53%. With two blocks, the performance increases by 0.8 percent points and the addition of a third block boosts the performance by 1.1 percent points, reaching a PQ of 54.9%. The mIoU remains approximately the same, but extra blocks improve the instance segmentation.

5.2.8 Learnable Queries

In this experiment, we evaluate whether the learnable queries can act as mask proposals. We visualize the proposals by generating the masks before the first decoder layer. In Figure 5.7, we compute the mask scores for each point via the dot product and visualize the proposals. For "stuff" classes, the learnable query covers all the points belonging to that class without differentiating the point coordinates. In other words, the queries that generate "stuff" masks are position invariant. In contrast, for "thing" classes, the queries must separate between instances and account for the spatial information, and the mask proposals focus on individual instances.

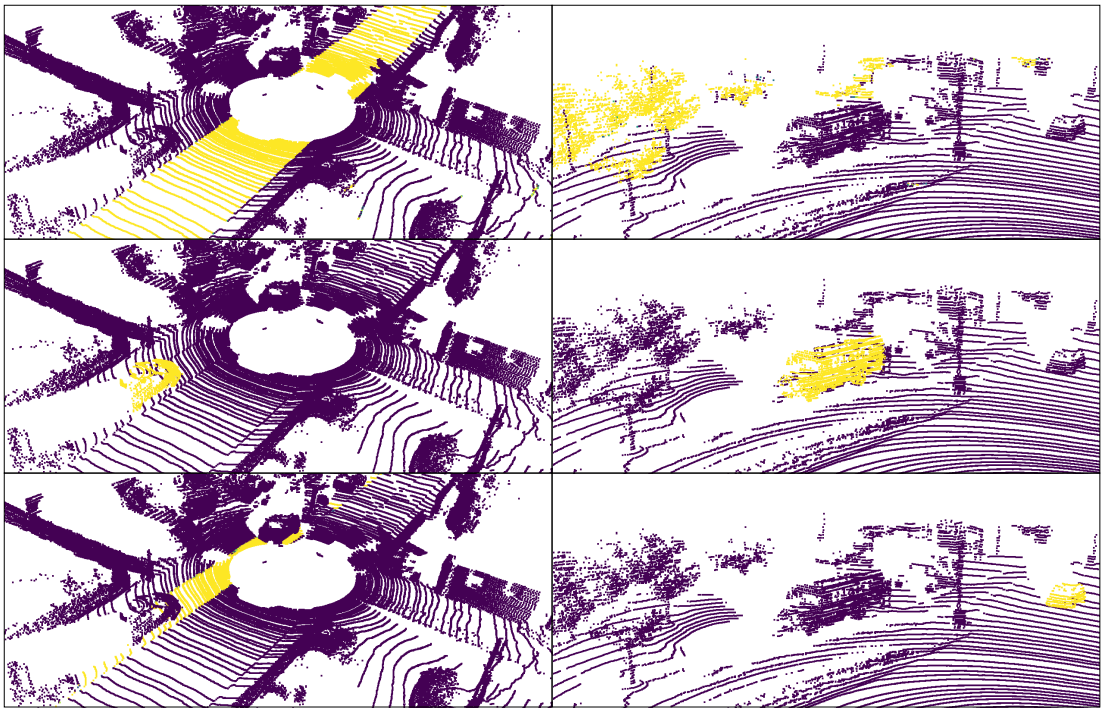


Figure 5.7: Mask proposals from some learnable queries showing "thing" instances and "stuff" classes. We obtain the mask scores via the dot product between point features and queries before the transformer decoder.

5.3 Conclusion

In this chapter, we presented an approach to perform panoptic segmentation of 3D LiDAR point clouds by directly predicting a set of binary masks along with their semantic classes. Our method allows for optimizing the model end-to-end, as we can fully avoid the need for any clustering algorithm as post-processing. This enables us to successfully use our method on different datasets without changing any hyperparameters and makes our model potentially more scalable. It gives us the possibility to incorporate additional training data without the need for hyperparameter tuning to adapt to the new data. We implemented and evaluated our approach on different datasets and compared it against baselines and techniques to achieve panoptic segmentation. We performed experiments to support all our claims and showed that our approach achieves competitive performance even with a semantic segmentation network without any modification as feature extractor.

As mentioned in previous chapters, we aim to learn from the data how to segment the whole LiDAR scan, identify instances, and how to fuse appearance and motion information to associate them over time. In Chapter 4, we proposed a method to extend a 3D panoptic segmentation network to 4D panoptic segmentation by associating per-scan instance predictions. The drawback of the method

# decoder blocks	PQ	RQ	SQ	mIoU
1	53.0	63.2	73.7	57.7
2	53.8	64.2	78.7	**58.5**
3	**54.9**	**64.9**	**79.7**	58.1

Table 5.5: Influence of the number of decoder blocks in panoptic quality on the validation set of SemanticKITTI. All metrics are reported in percent (%).

was that we cannot jointly optimize for segmentation and association and learn all aspects of the task from the data. It requires hand-tuning parameters for the clustering step in the segmentation and the association step. With the proposed approach in this chapter, we remove the clustering step and are able to learn from the data how to segment instances of different semantic classes, which encompass different shapes and sizes. This is the first step towards an end-to-end model for 4D panoptic segmentation. In the next chapter, we investigate how to build on top of the presented approach and extend it to obtain end-to-end 4D panoptic segmentation by also learning the instance associations from the data.

Chapter 6

Mask-Based 4D Panoptic Segmentation

In the context of autonomous navigation with robots or self-driving cars, spatio-temporal scene understanding is crucial to navigate the environment, since it is necessary to segment the scene and identify other agents and track their motion over time. In Chapter 4, we proposed a method for 4D panoptic segmentation by associating the instance predictions of individual scans over time. This method, however, requires hand-tuning parameters for the segmentation of the scan and for the association of instances over time. This does not allow our model to train end-to-end and learn the complete task from the data, which makes it less scalable. In Chapter 5, we went a step back and tackled the task of 3D panoptic segmentation (without considering the temporal dimension) and proposed a method that learns to identify instances of different semantic classes directly from data, removing the need for hand-tuning parameters and a clustering step. This allows our model to learn the segmentation aspect of 4D panoptic segmentation from the data and is the first step into an end-to-end model. In this chapter, we investigate the problem of 4D panoptic segmentation for 3D LiDAR scans, which requires a semantic annotation of each LiDAR scan, but also information about the evolution of the individual instances throughout the whole sequence. As illustrated in Figure 6.1, existing approaches use post-processing steps like clustering or association between predictions to output the final 4D predictions. This does not allow for end-to-end training to learn all the aspects of the task from the available data. Existing approaches can be divided into two groups. The first group follows the tracking-by-detection paradigm [61]. They perform 3D panoptic segmentation of the individual scans and later associate the current scan-wise instance predictions with the instances detected in previous timesteps. The approach presented in Chapter 4 follows this paradigm. The second group aggregates a few consecutive LiDAR scans to ob-

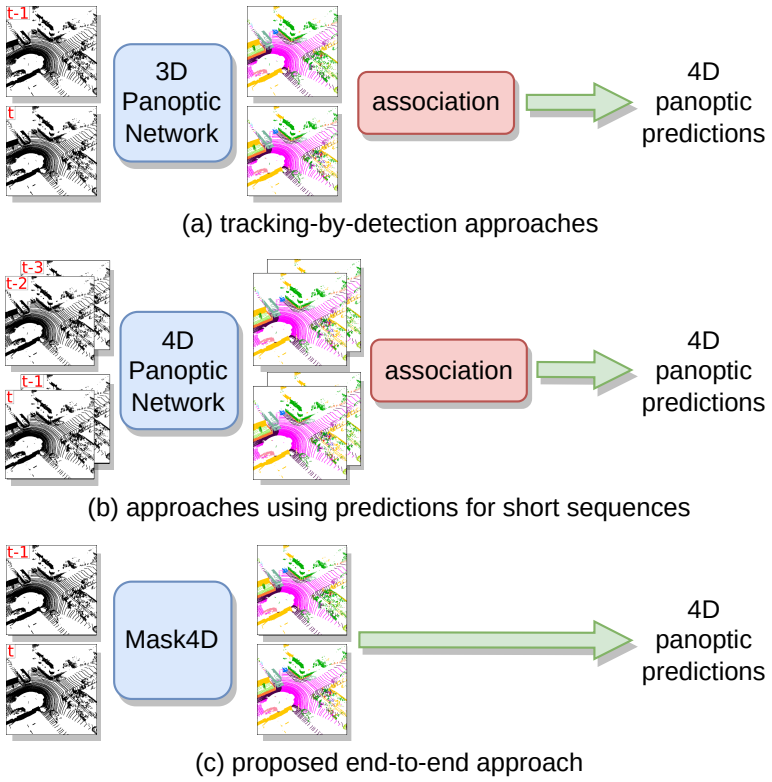


Figure 6.1: Different ways to obtain 4D Panoptic Segmentation: (a) tracking-by-detection: segment individual scans and associate predictions over time [61] as described in Chapter 4. (b) predictions for a few scans: use aggregated scans as input and clusters over the 4D point cloud. To get consistent instances over the whole sequence, associate the short sequence predictions [2, 52, 73, 177]. Other approaches rely on post-processing steps like clustering and association, while our method (c) is end-to-end trainable, operates on a scan-to-scan basis, and directly outputs 4D predictions.

tain a 4D point cloud and perform clustering over space and time simultaneously to obtain predictions for short sequences. The instance IDs are consistent only within these short sequences and therefore they need to associate these short sequences to obtain predictions consistent over time [2, 52, 73, 177]. As mentioned before, we investigate tackling this task in an end-to-end manner. This allows us to jointly optimize for segmentation and association and learn both aspects of the task simultaneously from the data, which cannot be achieved with previous methods, which use non-differentiable steps. Our approach uses the network proposed in Chapter 5 to perform both 3D and 4D panoptic segmentation without relying on any post-processing step like clustering to obtain instance segmentation or an association step between predicted instances. By removing the post-processing steps, we also remove the necessary hand-tuning of hyperparameters or the manual design of a cost matrix. Furthermore, we modify the cross-attention operation in the decoder to add spatial prior information of the instance position given pre-

vious detections. The way we can combine appearance and spatial information is crucial and has a critical influence on the performance. By proposing a fully end-to-end trainable approach, we allow our model to learn from the data and choose the best way of combining these different types of information for this particular task in different contexts.

The main contribution of this chapter is a model that performs 4D panoptic segmentation that can be trained end-to-end without any post-processing step. Inspired by MaskPLS presented in Chapter 4, we combine learnable queries and point features from a backbone to obtain binary masks and semantic classes for each scan. To perform tracking, the queries that predicted instances in the previous scan are used in the subsequent steps to decode the same instances over time and obtain consistent instance IDs throughout the whole 3D LiDAR sequence. We propose a loss function that provides negative samples during training to enforce dissimilar feature vectors for different instances and improves the tracking performance. Query-based detection networks tend to focus mainly on the appearance of the instances and often do not include important position information. To alleviate this problem, we leverage the sequential nature of commonly recorded outdoor scenes using 3D LiDAR scans and propose position-aware mask attention. This modification allows us to increase the attention weights on areas where the instance is located based on previous detections and motion estimation. This leads to an approach that allows us to directly train for the desired task and jointly optimize segmentation and association.

In sum, we make three key claims: First, our method achieves competitive performance in 4D panoptic segmentation while being end-to-end trainable without the need for any post-processing step. Second, our proposed loss function improves the tracking performance by providing negative samples. Third, our position-aware mask attention improves the performance by including spatial prior information from the LiDAR sequence. The implementation of our approach is publicly available at `https://github.com/PRBonn/Mask4D`.

6.1 Our Approach

We base our work on MaskPLS, introduced in Chapter 5, a mask-based 3D panoptic segmentation network, which is end-to-end trainable and directly outputs a set of binary masks and their corresponding semantic classes. It consists of a feature extractor and a transformer decoder to combine learnable queries with the extracted point-wise features. The learnable queries act as mask proposals and allow us to obtain the mask predictions after refining the queries with the transformer decoder. We extend this work to the temporal domain by reusing the queries that detected instances in previous scans to track the same instance

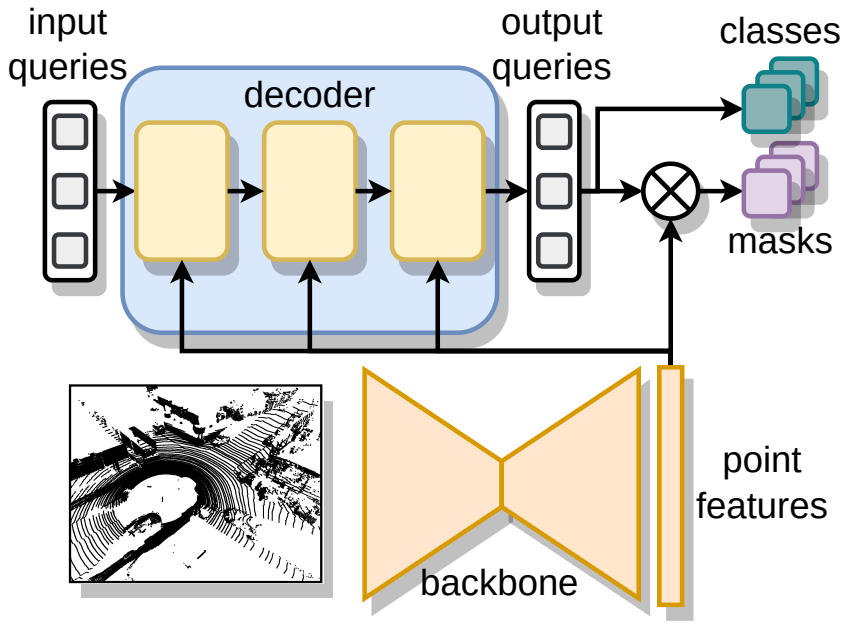


Figure 6.2: MaskPLS overview. The features from the backbone are combined in each decoder layer with N learnable queries. We obtain the mask predictions via the dot product between queries and point features and the semantic class from the queries.

over time and modify cross-attention to include spatial information about the instance position given the detections in previous scans.

6.1.1 Brief MaskPLS Review

Before presenting our approach, we first review the concepts of the 3D model MaskPLS, also illustrated in Figure 6.2. Given a point cloud $\mathcal{P} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_N\}$ with point coordinates $\boldsymbol{p}_i \in \mathbb{R}^3$, we represent it as a matrix

$$P = \begin{bmatrix} \boldsymbol{p}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{p}_N^\mathsf{T} \end{bmatrix}, \tag{6.1}$$

where each row i represents a point $\boldsymbol{p}_i \in \mathbb{R}^3$. They use a backbone to extract point-wise features $F \in \mathbb{R}^{N \times C}$ and a transformer decoder with mask attention and learnable queries $Q \in \mathbb{R}^{M \times C}$. Each of the M queries $\boldsymbol{q}_i \in \mathbb{R}^C$ is a learnable feature vector used as mask proposal, which are input to the decoder. The queries are refined through consecutive decoder layers to predict the masks in the scan, either an instance of a thing class or a full stuff class. Each of the D decoder layers follows the original transformer decoder [142], replacing cross-attention with mask attention between the queries and point features from the backbone, followed by self-attention and a feedforward network (FFN). The mask scores for

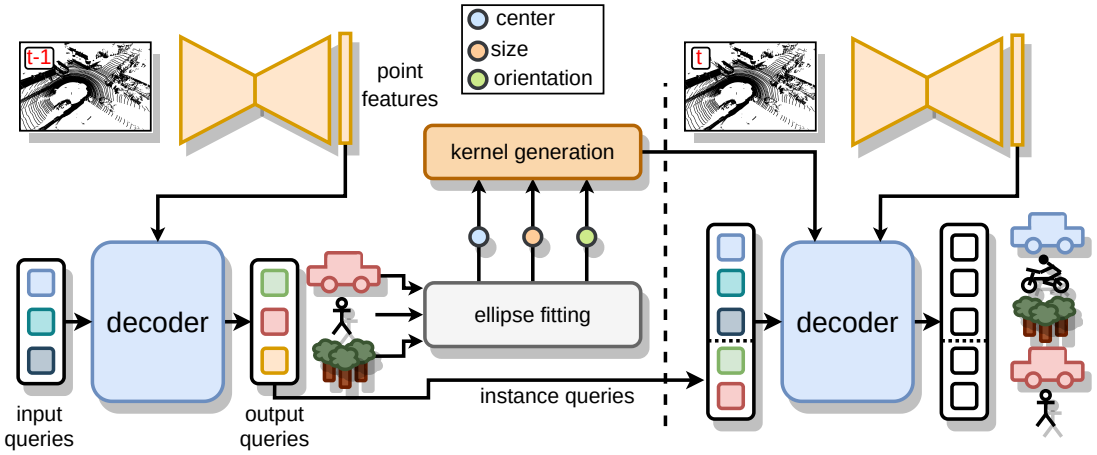


Figure 6.3: Overview of our method. We extract point features and combine them with the learnable queries to perform panoptic segmentation by predicting masks and semantic classes. We reuse the output queries that decoded instances as tracking queries in the next scans. After each detection, we fit an ellipse into the instances to get its center, size, and orientation, and generate a kernel to modify the attention weights. The tracking queries carry the identity of the instances, allowing us to keep consistent instance IDs over time.

each output query, i.e., the scores of each point for each of the M predicted masks $\boldsymbol{m}_i \in [0,1]^N$, are obtained from the output query $\boldsymbol{q}_i$ and the point features $\boldsymbol{F}$, as follows:

$$\hat{\boldsymbol{m}}_i = \text{sigmoid}(\boldsymbol{F}\boldsymbol{q}_i^\top). \tag{6.2}$$

To obtain the final semantic class for the predicted mask, one selects the most likely class based on the output class probabilities. They assign the mask with the highest score to each point to obtain our mask predictions. For further information, refer to MaskPLS presented in Chapter 5.

In MaskPLS, a fixed number of M queries at the input must decode all classes and objects in the scene. These queries get refined through the cross and self-attention layers in the decoder, and the output queries are combined with the point features to decode a single mask.

6.1.2 Mask4D for 4D Panoptic Segmentation

In this section, we explain how we extend MaskPLS from 3D to 4D panoptic segmentation. In MaskPLS, the output queries that detected instances have information about the instance's appearance and position. In our approach, we assume that instances do not change too much their appearance or position in two consecutive LiDAR scans, and therefore we can use the output queries to

always detect the same instance over time. This way, we leverage the existing 3D model and modify it to perform 4D panoptic segmentation.

In our proposed approach Mask4D, we use two groups of queries as input: detection queries Q_{det} and tracking queries Q_{tr}, which we concatenate and input simultaneously into the network at each step, together with the LiDAR scan. New instances and the stuff classes are decoded by the fixed M detection queries Q_{det} while the already tracked instances are decoded by their corresponding tracking queries Q_{tr} and thus keep a consistent instance ID. The number of Q_{tr} varies over time depending on the number of instances being tracked. This way, we do not perform any association or post-processing, and our approach directly outputs for each point a semantic class and instance IDs which are consistent over time.

Tracking using queries: We depict the inference of our model in Figure 6.3. First, we detect all the instances and the stuff classes with Q_{det}. Each time we decode a new instance, we take its corresponding output query to initialize a new tracking query, which carries the instance identity across the whole LiDAR sequence. We use the probability of the most likely class as a classification score to decide when to initialize a new tracking query and when to perform re-identification. We initialize a new tracking query when the classification score is larger than a threshold τ_{new}. In the next scan, we concatenate Q_{det} and Q_{tr} and input them to the decoder, along with the point features from the backbone. We decode the stuff classes and the newly appearing instances with Q_{det} and the already tracked instances with Q_{det} and assign consistent instance IDs. To adapt to the changing appearance of the instance, we replace the tracking query with the corresponding output query after each detection. The self-attention applied to all queries allows us to detect new instances and avoid the re-detection of the tracked objects.

Re-identification: To handle occlusions or instances that temporally leave the scene, we keep queries of all tracked instances, even if they have not been detected (inactive) for a maximum of τ_{life} scans. If an inactive query decodes an instance with a classification score larger than a threshold τ_{re}, we perform re-identification and assign the ID of the inactive track to the instance. This way, we preserve the identity of the instances even in the case of short-term occlusions.

6.1.3 Training Setup

We illustrate our training procedure in Figure 6.4. For simplicity, we only show the predicted mask and the paired ground truth mask for each output query, to depict how we compute the loss functions. We train our model by sequentially providing S scans randomly sampled from a sequence of length L. The aim of the training is for the model to segment the whole scan and each instance over time with the same query, giving a consistent instance ID. We segment the first scan

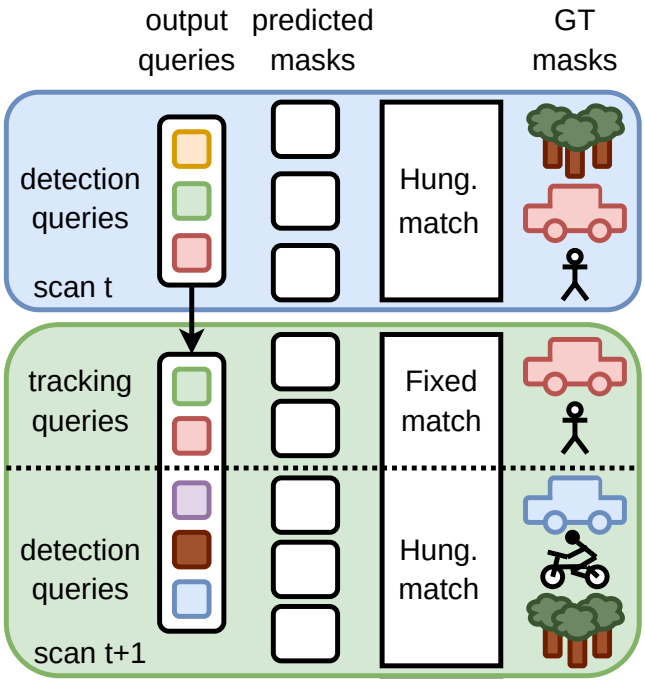


Figure 6.4: Training setup of Mask4D. We run the inference for scan t and compute the losses on the pairs matched using the Hungarian algorithm. At $t+1$ we input detection and tracking queries and compute the losses over two sets. First, with a fixed matching for the tracking queries, and second, with the Hungarian algorithm for the detection queries to match with new instances and stuff classes.

using only Q_{det} to decode all instances and stuff classes, and use the Hungarian algorithm to match them with the ground truth masks and compute the losses as in MaskPLS. For the subsequent $S-1$ scans, we use the same set Q_{det} but add the output queries that decoded instances in the previous step as Q_{tr}. We want detection queries to decode only the new instances and stuff masks, and tracking queries to decode the already tracked instances. To do that, we perform a fixed matching between the prediction of Q_{tr} and their ground truth masks, and then the Hungarian algorithm for predictions given by Q_{det}.

6.1.4 Loss Function

MaskPLS loss review. In MaskPLS, the Hungarian algorithm matches all predicted masks $\mathcal{Z}$ and ground truth masks $\mathcal{Y}$ to compute the cross-entropy loss for the class and a mask loss over the K matched pairs:

$$\mathcal{L}_m = \sum_{j=1}^{K} -\lambda_{\text{cls}} \log \hat{p}_j(c_i) + \mathcal{L}_{\text{mask}}, \tag{6.3}$$

where $\hat{p}_j(c_i)$ is the probability of class c_i and $\mathcal{L}_{\text{mask}}$ is the combination of dice loss Dice and binary cross-entropy CE:

$$\mathcal{L}_{\text{mask}} = \sum_{k=1}^{M} \lambda_{\text{d}} \text{Dice}(\boldsymbol{m}_i(k), \hat{\boldsymbol{m}}_j(k)) + \lambda_{\text{b}} \text{CE}(\boldsymbol{m}_i(k), \hat{\boldsymbol{m}}_j(k)), \tag{6.4}$$

where $\boldsymbol{m}_i$ is the ground truth mask i, $\hat{\boldsymbol{m}}_j$ the mask scores for query j and M is the number of points. For the non-matched masks $\mathcal{Z}_n$, the model is enforced to predict the "no_object" class $\emptyset$ by computing only the class loss with $\alpha < 1$:

$$\mathcal{L}_n = \sum_{j \in \mathcal{Z}_n} -\alpha \log \hat{p}_j(\emptyset). \tag{6.5}$$

Their final loss $\mathcal{L}_d$ is given by the sum of both losses:

$$\mathcal{L}_d = \mathcal{L}_m + \mathcal{L}_n. \tag{6.6}$$

The loss supervises the mask predictions obtained with the input queries and after each of the D decoder layers.

Detection and tracking loss: We enforce tracking queries Q_{tr} to decode the same instance over time and detection queries Q_{det} to decode new instances and stuff classes.

In the first scan, we are not tracking any instance and therefore use the same procedure as MaskPLS to compute our detection loss $\mathcal{L}_d$ following Equation (6.6). For the following steps, the set of predictions $\mathcal{Z} = \mathcal{Z}_{\text{det}} \cup \mathcal{Z}_{\text{tr}}$ consists on the predictions $\mathcal{Z}_{\text{det}}$ made by Q_{det} and the predictions $\mathcal{Z}_{\text{tr}} = \mathcal{Z}_{\text{ac}} \cup \mathcal{Z}_{\text{in}}$ made by Q_{tr}, which correspond to the tracked instances present in the current scan (active) $\mathcal{Z}_{\text{ac}}$ and the tracked instances not present in the scan (inactive) $\mathcal{Z}_{\text{in}}$. We first perform a fixed matching between the active tracking predictions $\mathcal{Z}_{\text{ac}}$ and their corresponding ground truth masks and compute the tracking loss $\mathcal{L}_t$ over the matched pairs using the same function as in Equation (6.3).

Second, we remove tracking predictions and their ground truth and compute $\mathcal{L}_d$ using Equation (6.6) over the rest of the predictions $\mathcal{Z}_{\text{h}} = \mathcal{Z}_{\text{det}} \cup \mathcal{Z}_{\text{in}}$ for the new appearing instances and stuff masks. We also supervise intermediate outputs with the detection loss, which gives $D + 1$ terms for each tracking loss term. In addition, the first scan is only supervised for detection, which creates an imbalance between the losses that we compensate for by down weighting $\mathcal{L}_d$ with $\alpha_d < 1$ and increase the contribution of $\mathcal{L}_t$ by $\alpha_t > 1$ in our detection and tracking loss $\mathcal{L}_{dt}$:

$$\mathcal{L}_{dt} = \alpha_d \mathcal{L}_d + \alpha_t \mathcal{L}_t. \tag{6.7}$$

Matching loss. During training, the fixed matching enforces the tracking predictions to resemble the right instances. During inference, however, a detection

query could decode a tracked instance, and a tracking query could decode a wrong instance. To account for that, we use the Hungarian algorithm between all predictions and ground truth and compute two losses. We select the K'_{det} instances predicted by Q_{det} that matched tracked instances and compute the loss in Equation (6.5). This way, we optimize these queries to predict "no_object" to avoid decoding a tracked instance at inference.

Then, we select the K'_{tr} pairs of prediction and ground truth corresponding to instances predicted by Q_{tr} that matched a wrong instance and perform a dissimilarity loss to enforce that the masks are different. We invert the ground truth mask $\boldsymbol{m}$ to generate the target as follows: $\boldsymbol{m}_i^* = |1 - \boldsymbol{m}_i|$. We only compute the loss over the foreground points $\mathcal{F} = \{i \mid \boldsymbol{m}(i) = 1\}$ of the ground truth mask $\boldsymbol{m}$. The dissimilarity loss $\mathcal{L}_{\text{dis}}$ is the same function in Equation (6.4) but only applied to the subset of foreground points $\mathcal{F}$:

$$\mathcal{L}_{\text{dis}} = \sum_{k \in \mathcal{F}} \lambda_{\text{d}}^* \text{Dice}(\boldsymbol{m}_i^*(k), \hat{\boldsymbol{m}}_j(k)) + \lambda_{\text{b}}^* \text{CE}(\boldsymbol{m}_i^*(k), \hat{\boldsymbol{m}}_j(k)), \tag{6.8}$$

where λ_{d}^* and λ_{b}^* are weighting factors, $\boldsymbol{m}_i^*$ is the inverted version of $\boldsymbol{m}$ and $\hat{\boldsymbol{m}}_j$ are the mask scores for query j. This enforces low scores for predictions obtained by Q_{tr} that match another instance, which can be thought of as negative samples.

The complete matching loss $\mathcal{L}_{\text{match}}$ is:

$$\mathcal{L}_{\text{match}} = \sum_{j=1}^{K'_{\text{det}}} -\alpha \log \hat{p}_j(\varnothing) + \sum_{j=1}^{K'_{\text{tr}}} \mathcal{L}_{dis}. \tag{6.9}$$

Our final loss $\mathcal{L}$ is the summation: $\mathcal{L} = \mathcal{L}_{dt} + \mathcal{L}_{\text{match}}$.

6.1.5 Position-aware Mask Attention

Analysing the network predictions, we observe ID switches between instances in different positions in the scan, like cars in different lanes. We argue that the problem might be that the queries encode mainly appearance information and lack important spatial knowledge.

To alleviate this problem, we propose position-aware mask attention. We leverage predictions of previous scans by adding information about the position and size of the instances to the cross-attention. Each time we detect an instance, we fit an ellipse to infer its position, size, and orientation, and generate a Gaussian-like kernel to modify the binary mask and the attention weights in the next step. The kernel increases the weights of points around the instance center to provide prior information from previous detections.

Ellipse fitting. We calculate the instance 2D center $\mu = (\mu_x, \mu_y)^\top$ as the mean of all instance point coordinates and fit a 2D ellipse, as shown in Figure 6.5

(top), using Singular Value Decomposition over the set of instance points $\mathcal{I}$. We compute $U\Sigma V^* = \text{SVD}(\mathcal{I})$, where $\Sigma = \text{diag}(s_1, s_2)$. We use the singular values s_1, s_2 as the magnitude of the semiaxes and V^* as the orientation.

Kernel generation. We generate a Gaussian-like kernel $\boldsymbol{g} \in \mathbb{R}^N$ for the N points in the point cloud for each instance $\mathcal{I}$, i.e., a kernel with value one at its center and exponentially decaying. We use an anisotropic squared exponential function $G(\mathbf{x})$:

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mu')^\top K'(\mathbf{x} - \mu')\right). \tag{6.10}$$

We use $R = V^*$ as rotation matrix to match the orientation of the instance and scale the kernel according to the instance size using the singular values s_1, s_2. We compute the new mean and covariance matrix as:

$$\mu' = R\mu, \quad \text{with} \quad K' = RKR^\top, \tag{6.11}$$

where

$$K = \begin{pmatrix} s_1^2 & 0 \\ 0 & s_2^2 \end{pmatrix}. \tag{6.12}$$

We obtain our Gaussian-like kernel $\boldsymbol{g} \in \mathbb{R}^N$ by applying $G(\mathbf{x})$ to the point cloud, i.e., $\boldsymbol{g}(i) = G(i), 1 \leq i \leq N$.

Mask generation. The mask attention proposed by Cheng et al. [22] is a variation of cross-attention that only attends within the foreground region of a binary mask for each query i and its output $\boldsymbol{x}_i$ is:

$$\boldsymbol{x}_i = \text{softmax}(\mathcal{M}_i + \boldsymbol{q}_i K^\top)V, \tag{6.13}$$

where $K = V$ are the point-wise features F and the attention mask $\mathcal{M}_i$ is given by:

$$\mathcal{M}_i(j) = \begin{cases} 0 & \text{if } \bar{\boldsymbol{m}}_i(j) = 1 \\ -\infty & \text{otherwise} \end{cases}, \tag{6.14}$$

where $\bar{\boldsymbol{m}}_i(j)$ is the binarized mask prediction $\hat{\boldsymbol{m}}_i$ (threshold at 0.5) of the previous layer for each point j, as in Equation (6.2).

We compute the kernel $\boldsymbol{g}$ by applying $G(x)$ to the points and add it to the binary mask to consider the area where the instance is located. We compute the kernel $\boldsymbol{g}$ using Equation (6.10) using the points belonging to the previous detection for each instance, add it to the mask scores and normalize the weights to obtain $\hat{\boldsymbol{m}}_i' = \text{norm}(\hat{\boldsymbol{m}}_i + \boldsymbol{g})$. The new attention mask is given by:

$$\mathcal{M}_i'(j) = \begin{cases} 0 & \text{if } \bar{\boldsymbol{m}}_i'(j) = 1 \\ -\infty & \text{otherwise} \end{cases}, \tag{6.15}$$

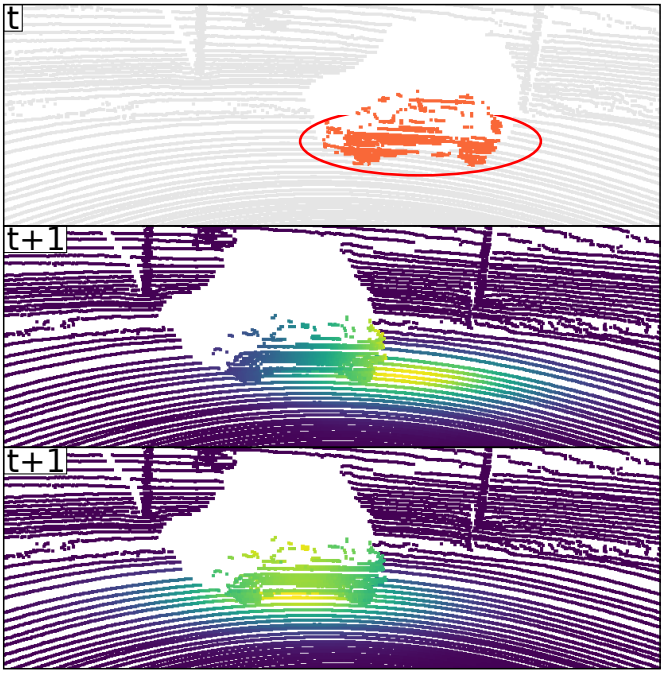


Figure 6.5: Application of the Gaussian-like kernel. At time t (top), we fit a 2D ellipse around the instance and generate our kernel. At $t+1$, we apply the kernel as a spatial prior. Compensating only the ego-motion (middle), the kernel does not consider the instance velocity, thus does not fully cover the instance, and includes other points. Updating the position of the instance using a motion model (bottom), we apply the kernel closer to the instance position. Colors depict the magnitude of the kernel for each point.

where $\tilde{\boldsymbol{m}}'_i$ is the binarized (thesholded at 0.5) version of $\hat{\boldsymbol{m}}'_i$. We generate a different kernel for each tracking query and use zeros as the kernel for the detection queries.

Attention weights. We want to use the spatial information to also modify the attention weights and emphasize the contribution of points around the instance position. Following previous works [36], we add the logarithm of the kernel to the attention weights before the softmax, as shown in Figure 6.6, and compute our position-aware mask attention as:

$$\boldsymbol{x}_i = \text{softmax}(\mathcal{M}'_i + (\boldsymbol{q}_i K^\top) + \log(\boldsymbol{g}))V. \tag{6.16}$$

6.1.6 Motion Compensation for Position-aware Mask Attention

To better estimate the instance locations, we compensate the ego-motion using the SLAM approach by Behley et al. [6] to help in the association process. After each detection, we update previous instance centers μ_{t-1} by applying the relative transformation ${}^{t}\mathsf{T}_{t-1}$, between the previous and current scan, and apply the kernel

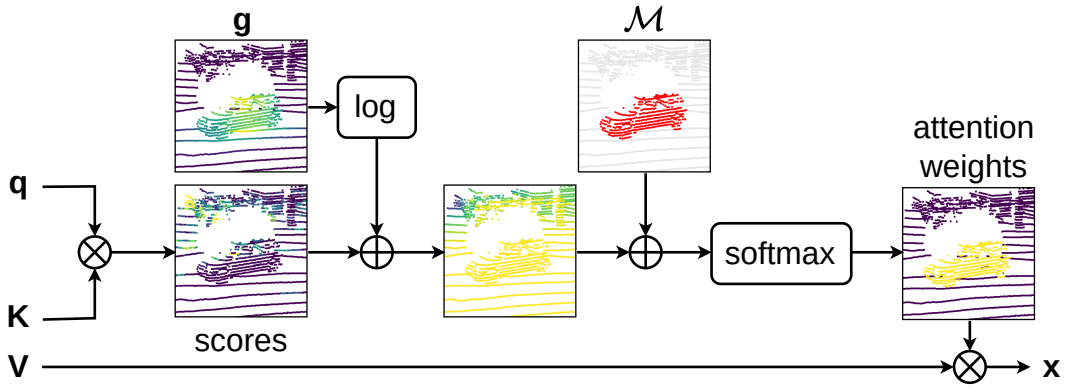


Figure 6.6: Structure of position-aware mask attention. We add the logarithm of the kernel as a spatial prior to the attention scores and apply the attention mask to consider a subset of points within the mask. The resulting attention weights focus on the desired instance.

around the new instance center $\mu_t = {}^t\mathsf{T}_{t-1}\mu_{t-1}$. However, the compensation does not consider the motion of the instances, as observed in Figure 6.5 (middle). To account for the motion, we predict the new positions of the instances with a constant velocity motion model and apply the kernel around the new instance centers, as depicted in Figure 6.5 (bottom). We update the inactive queries as well for re-identification.

6.2 Experimental Evaluation

The main focus of this chapter is an approach for 4D panoptic segmentation that does not require any post-processing step and can be trained in an end-to-end fashion. In this section, we present our experiments to show that (i) our method achieves competitive performance in 4D panoptic segmentation while being end-to-end trainable without the need for any post-processing step, (ii) our proposed loss function improves the tracking performance by providing negative samples, and (iii) our position-aware mask attention improves the performance by including spatial prior information from the LiDAR sequence.

6.2.1 Experimental Setup

We evaluate our method using the 4D Panoptic Segmentation benchmark [2] of SemanticKITTI [3, 4]. It provides point-wise annotations for 22 sequences of the KITTI odometry dataset [38]. We use the LiDAR Segmentation and Tracking Quality (LSTQ) metric [2] given by $LSTQ = \sqrt{S_{\text{cls}} \cdot S_{\text{assoc}}}$, which is the geometric mean of the classification score S_{cls} and the association score S_{assoc}. We further

report the intersection over union for stuff IoU^{St} and things classes IoU^{Th}.

6.2.2 Implementation Details

We build on top of MaskPLS using $D = 6$ decoder layers, and use SphereFormer [75] as backbone and $N = 100$ detection queries. We keep inactive tracks for $\tau_{\text{life}} = 5$ scans and use $\tau_{\text{new}} = \tau_{\text{re}} = 0.8$ for new detected instances and re-identification. We use the same cost as MaskPLS for the matching and the same weights to build $\mathcal{L}_d$ and $\mathcal{L}_t$, which we combine in $\mathcal{L}_{dt}$ using $\alpha_d = 0.5$, $\alpha_t = 50$ to empirically equalize the losses. For the disimilarity loss $\mathcal{L}_{\text{dis}}$, we use $\lambda_{\text{d}}^* = \lambda_{\text{b}}^* = 20$. We train for 50 epochs using AdamW [92] optimizer and 0.0001 as a fixed learning rate. We use $S = 3$ scans randomly picked from a sequence of $L = 10$ as input. When submitting to the test set, we use random rotation and translation as test-time augmentations.

6.2.3 4D Panoptic Segmentation Performance

The first experiment evaluates the performance of our approach on the SemanticKITTI validation set in Table 6.1 and test set in Table 6.2. Our method achieves the best performance both in the validation and the test set. Our model achieves state-of-the-art performance in 4D panoptic segmentation while being end-to-end trainable without the need for any post-processing step. Our model ranks first in the benchmark without requiring the hyperparameter tuning necessary for the post-processing steps, and allows us to optimize jointly for segmentation and association. In contrast, CA-Net, presented in Chapter 4, uses a clustering step to predict instances and later an association step, which needs a manually designed cost function. The other works [2, 52, 73, 177] follow a similar procedure and use a clustering step to obtain 4D predictions on short sequences of a few scans, and an association step to combine them into the final 4D predictions. Particularly, 4DPLS, 4D-StOP, and Eq-4D-StOP save all the predictions and run an offline association step to stitch them together, finding the best association between overlapping tracklets. This means that these methods are not online since they require predictions for future LiDAR scans. On the other hand, our method is online, as it does not rely on future observations, and we believe this is the reason for the difference in the association score between our approach and the approaches 4D-StOP and Eq-4D-StOP.

6.2.4 Ablation Studies

Next, we assess the influence of our contributions on the performance of our approach, namely the loss function, different ways of computing the Gaussian-

Method	LSTQ	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}
4DPLS[2]	62.7	65.1	60.5	65.4	61.3
4D-DS-Net[52]	68.0	71.3	<u>64.8</u>	64.5	65.3
CA-Net	68.0	72.9	63.4	64.6	62.0
4D-StOP[73]	67.0	74.4	60.3	65.3	60.9
Eq-4D-StOP[177]	<u>70.1</u>	**77.6**	63.4	**66.4**	<u>67.1</u>
Mask4D (ours)	**71.4**	<u>75.4</u>	**67.5**	<u>65.8</u>	**69.9**

Table 6.1: SemanticKITTI validation set results. All metrics are reported in percent (%). Bold numbers indicate the best results, and underlined numbers indicate the second best.

Method	LSTQ	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}
4DPLS[2]	56.9	56.3	57.4	66.9	51.6
4D-DS-Net[52]	62.3	65.8	58.9	65.6	49.8
CA-Net	63.1	65.7	60.6	66.9	52.0
4D-StOP[73]	63.9	<u>69.5</u>	58.8	67.7	53.8
Eq-4D-StOP[177]	<u>67.0</u>	**72.0**	<u>62.4</u>	<u>69.1</u>	<u>60.9</u>
Mask4D (ours)	**67.1**	66.4	**67.8**	**71.8**	**62.4**

Table 6.2: SemanticKITTI test set results. All metrics are reported in percent (%). Bold numbers indicate the best results, and underlined numbers indicate the second best.

like kernel, and the motion compensation. To reduce the total wall-clock training time, we train the model using $S = 2$ scans as input for 25 epochs.

6.2.4.1 Loss Functions

In this experiment, we evaluate the performance of the model for different loss functions in setups (A) and (B). This experiment supports our second claim, that our proposed loss function improves the tracking performance by providing negative samples. To evaluate only the influence of the loss functions, we train the models with mask attention and show the results in Table 6.3. First, in (A), we use the detection and tracking loss and obtain $LSTQ = 61.7\%$. Next, in (B), we add our matching loss to provide negative samples and improve the association S_{assoc} by 9.5 percent points but hinder S_{cls} by 0.4 percent points. This shows that adding our loss improves the segmentation and tracking performances.

#	tracking	matching	*LSTQ*	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}
A	✓	-	61.7	56.5	67.3	66.3	**68.7**
B	✓	✓	**66.5**	**66.0**	**66.9**	**66.9**	66.8

Table 6.3: Influence of the loss functions on the validation set. All metrics are reported in percent (%).

6.2.4.2 Kernel Computation

In this experiment, we illustrate how our modification of mask attention improves the performance by including spatial prior information from the LiDAR sequence. We show different ways of computing the Gaussian-like kernel in Figure 6.7 and the performance of the model in Table 6.4. In setting (C), no kernel is applied, and in setting (D), we compensate for the ego-motion and apply a fixed-size isotropic kernel (as shown in Figure 6.7-row 1) only to the binary mask and not to the attention weights. We improve S_{assoc} by 2.8 percent points and LSTQ by 1 percent point.

Next, we compensate for the ego-motion and use different kernels to adapt to the instances. In setting (E), we use the same kernel as in setting (D), and LTSQ improves by 1.4 percent points. In setting (F), the size of the kernel is proportional to the size of the instance, as seen in Figure 6.7-row 2, which improves the performance by 1.7 percent points. Third, in setting (G), we compute the instance size in the X and Y coordinates and generate an axis-aligned anisotropic kernel scaled with the size of the instance, as shown in Figure 6.7-row 3. The performance improves by 1.5 percent points but drops by 0.2 percent points related to the setting (F). Finally, in setting (H) we fit an ellipse into the instance using SVD and compute an anisotropic kernel scaled and rotated to match the instance, as seen in Figure 6.7-row 4 and improve S_{assoc} by 1.8 percent points and LSTQ reaches 69.4%. This shows the improvement given by our proposed position-aware mask attention, which makes the model slower, taking 500 ms per scan in contrast with the 300 ms needed by MaskPLS when measured on an NVIDIA RTX A5000 GPU.

6.2.4.3 Motion Compensation

In the next experiment, we evaluate how the motion compensation for the instances affects the performance of the model and show our results in Table 6.5. We use the anisotropic kernel based on the ellipse computation. The validation set (sequence 08) occurs inside a city, with the ego-car moving at a low speed and with only 11% of the instances moving. In contrast, in sequence 01 the ego-car drives on a highway and all instances are moving. The results in both sequences

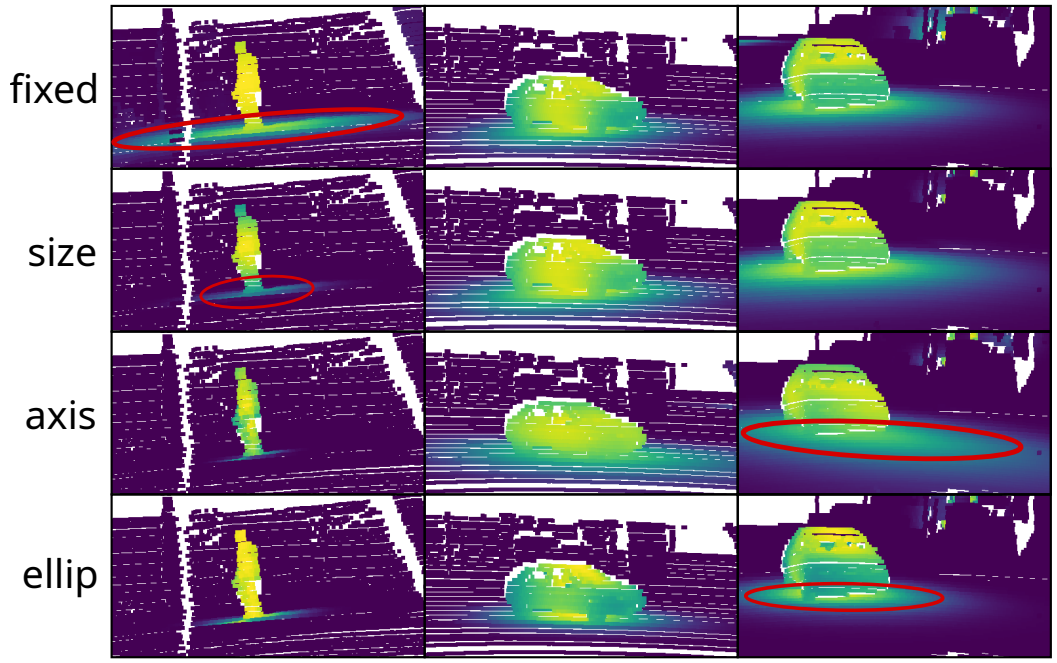


Figure 6.7: Different kernels for position-aware mask attention. Row 1 (fixed) shows an isotropic kernel with fixed size, which does not adapt to different instances like the pedestrian, and includes background points. Row 2 (size) shows an isotropic kernel scaled by the size of the instance, which adapts better to different classes. Row 3 (axis) shows an anisotropic axis-aligned kernel scaled by the instance size. It adapts to axis-oriented instances (center) but not to non-aligned instances (right). Row 4 (ellip) shows an anisotropic kernel scaled and rotated to match size and the orientation. It adapts to different classes and different orientations. Colors depict the magnitude of the kernel for each point.

depict the influence of the motion compensation also in dynamic environments.

In setting (I), we apply the Gaussian-like kernel using the instance centers in the coordinates of the previous scan. This spatial information helps for instances located further apart, as mentioned in Section 6.1.5. Next, in setting (J), we compensate the ego-motion by applying the relative transformation between scans, increasing LSTQ by 0.5 percent points for sequence 08 and 2.2 percent points for sequence 01, because the kernel is applied closer to the actual instance center, but we do not account for the motion of dynamic instances, as seen Figure 6.5-middle. Finally, in setting (K), we predict instance centers with a constant velocity motion model and compute the kernel around it, as shown at the bottom of Figure 6.5. We improve the performance by 0.3 percent points for sequence 08 and 0.7 percent points for sequence 01. This highlights how compensating for the motion of the instances when computing the kernel leads to a better localized attention and improves the segmentation performance

#	Kernel	LSTQ	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}
C	no kernel	66.5	66.0	66.9	66.9	66.8
D	only mask	67.5	68.8	66.3	66.3	66.1
E	fixed	68.9	70.9	67.0	66.6	68.7
F	size	69.2	71.7	66.7	66.6	**69.2**
G	size xy	69.0	70.4	**67.7**	**66.7**	68.9
H	ellipse	**69.4**	**71.8**	67.2	**66.7**	68.8

Table 6.4: Influence of how to calculate the kernel on SemanticKITTI validation set. All metrics are reported in percent (%). (C) not applying any kernel, (D) using the kernel only for the binary mask attention, and (E-G) applying the kernel computed in different ways: (E) fixed size, (F) scaled with the instance size, (G) scaled with the instance size in X and Y; and (H) scaled and rotated to match instance size and orientation.

#	center update	LSTQ		S_{assoc}		S_{cls}	
	sequence	08	01	08	01	08	01
I	local coords	68.9	68.7	70.5	82.1	**67.2**	57.5
J	ego-motion	69.4	70.9	71.8	82.6	**67.2**	**60.9**
K	motion model	**69.7**	**71.6**	**72.3**	**84.2**	67.1	**60.9**

Table 6.5: Influence of the different pose compensations on SemanticKITTI sequence 08 and sequence 01. All metrics are reported in percent (%).

6.2.5 Queries for Tracking

In this last experiment, we show an insight into why we add spatial information to better track instances over time. In Figure 6.8, we show the consecutive instance predictions of MaskPLS for a few scans in a highway scenario. The colors represent the query number that detected the instance as its ID. In the circled areas, we observe certain consistency in the instance IDs. On the left, the car that follows the ego-car remains in a similar relative position and gets assigned the same ID because it is decoded by the same query. On the right, the same instance ID is assigned to different cars that traverse the same spatial region. These examples show that the queries decode masks of a particular semantic class in a particular spatial region. Instead, we want the queries to follow a particular instance over time. This means that we have to modify the spatial information of the queries to decode the tracked instance in its new position. This observation motivates our proposed position-aware mask attention to include spatial information instead of relying only on the query information, which always

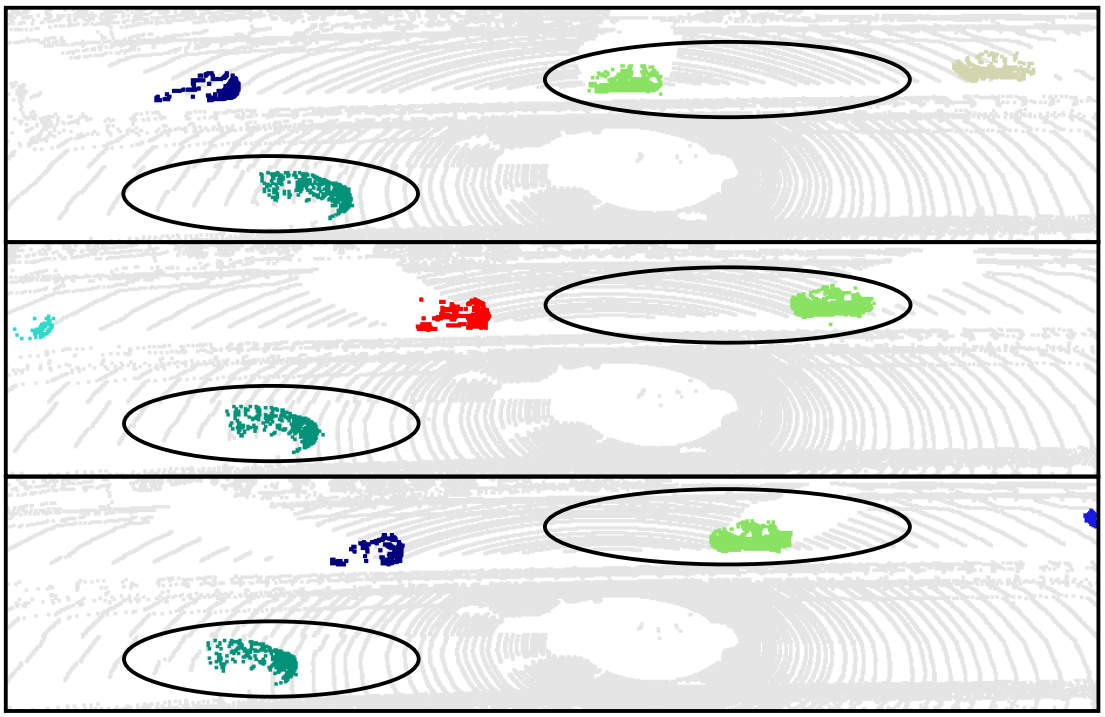


Figure 6.8: Instance segmentation results of sequential scans using MaskPLS. Colors depict instance IDs. Instances at a similar spatial location are decoded by the same query over time and therefore keep the same ID. The queries decode instances in a specific area.

points to the same relative position.

The presented experiments show how we can extend MaskPLS for 3D to 4D panoptic segmentation by reusing the output queries and achieve state-of-the-art performance. Furthermore, we show how our proposed loss function and Gaussian-like kernel improve the segmentation and tracking performance, and the rol of motion compensation.

6.3 Conclusion

In this chapter, we presented a novel approach to perform 4D panoptic segmentation without any post-processing step, like clustering or instance associations, that achieves state-of-the-art performance while being end-to-end trainable. We use the mask-based 3D panoptic segmentation network presented in Chapter 5 and extend it to 4D by using the same query to segment the same instance over time. The experiments show that our proposed loss function improves the performance by providing negative samples. Finally, our position-aware mask attention allows us to include spatial prior information in the cross-attention and improve the association. This enables us to jointly optimize for segmentation and as-

sociation and let the network choose how to combine appearance and position information.

With this approach, we have a model that learns the segmentation and association from the data without the need to hand-tune hyperparameters to adapt the approach for different environments. When modifying the dataset with additional data recorded in different scenarios where the driving conditions and traffic participants might be different, our model can potentially learn both segmentation and association from this variety of data during the training phase, without the need for manual hand-tuning. This makes our model potentially more scalable than previous approaches, which is valuable for autonomous systems that continuously collect data to include a variety of scenarios to enhance the performance of the perception systems.

In this first part of the thesis, we worked with 3D data from LiDAR scans and focused on the segmentation of the scene while identifying individual traffic participants over time. In the next part, we work with systems that count only with RGB cameras and focus on the reconstruction of the 3D surrounding scene.

Part II

Image Occupancy Prediction and Panoptic Segmentation

Chapter 7

Vision-Based 3D Semantic Occupancy Prediction

In the context of outdoor navigation, semantic scene understanding plays a crucial role in enabling safe navigation in complex environments. This entails understanding the surroundings geometrically and semantically to be able to react accordingly. 3D perception tasks often rely on costly 3D sensors like LiDAR, which provide an accurate geometric representation, whereas vision-centric scene understanding seeks to provide meaningful information about the surrounding scene using only RGB cameras. In Part I, we studied how to tackle semantic scene understanding relying on a 3D LiDAR. In this second part, we rely only on RGB cameras, which are quite orthogonal sensors, as they offer high-resolution images with rich texture and color. The generated data, however, is inherently 2D and lacks direct depth information to provide a geometric representation of the 3D surroundings. While the recent advances in computer vision in the image domain, including large-scale annotated datasets [45, 72, 86] and pretrained models [72, 91, 127] provide off-the-shelf image segmentation, several tasks have been proposed to predict the structure of the environment in an online fashion using only camera data recorded at a single timestamp. 3D semantic occupancy prediction is a vision-centric task that aims to represent the 3D geometric structure of the surrounding scene from a setup of surrounding cameras, which means extracting 3D information from images. This task involves dividing the surrounding scene into a voxel grid and predicting, for each voxel, whether it is occupied or not, and its corresponding semantic class in case of being occupied. Other tasks for geometric and semantic scene understanding from images include 3D object detection and depth estimation. Compared to 3D object detection [83, 84, 117], occupancy prediction [59, 60, 113, 139] provides a more fine-grained representation and, in contrast to monocular depth estimation [43, 153], it seeks to also reconstruct the environment in occluded areas.

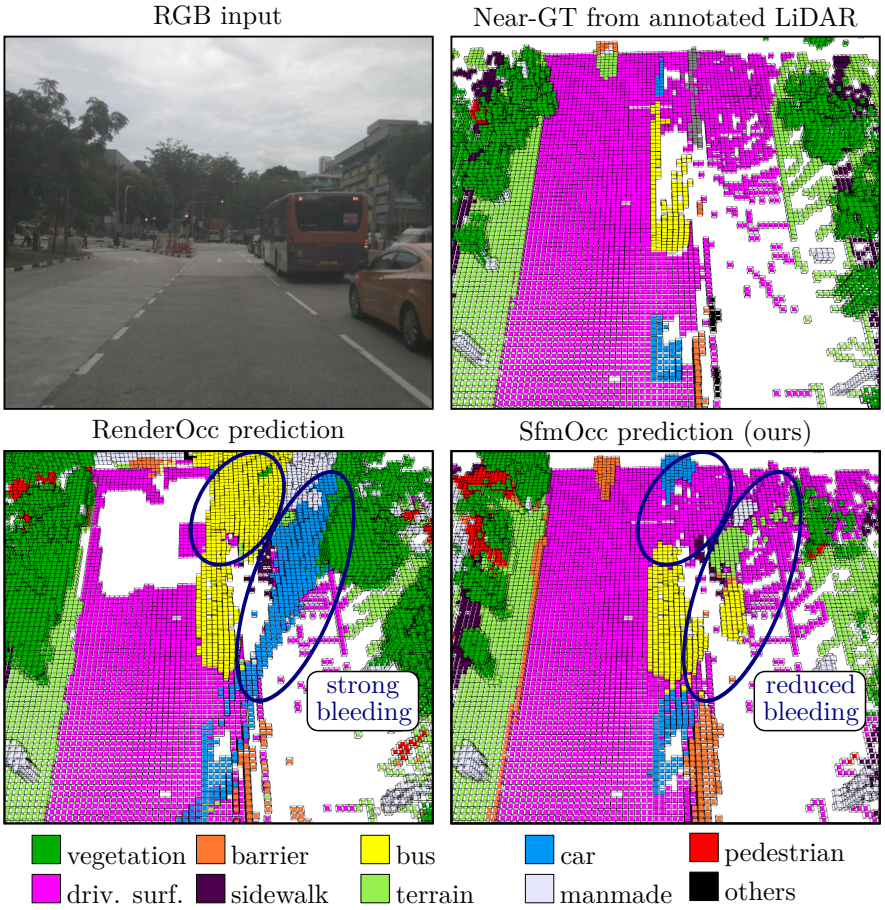


Figure 7.1: Approaches that do not rely on ground-truth occupancy labels for training, like RenderOcc [113], usually perform implicit supervision and have bleeding effects in their predictions, as highlighted by circles in the image. Using only RGB images for training, we explicitly supervise for occupancy, which allows us to predict occupancy and reduce the bleeding.

State-of-the-art approaches for 3D semantic occupancy prediction [60, 139, 172] rely on 3D voxel labels as an explicit supervision signal to learn the full occupancy of the surrounding scene. Due to the costly annotation process, some recent approaches project segmented LiDAR scans [9, 113] into the images and use these for supervision. Given the sparsity of a 3D LiDAR compared to camera images, they use multiple past and future scans to obtain denser supervision signals, but rely on the availability of a LiDAR sensor. They see the problem of occupancy estimation as depth prediction of the surrounding scene using depth images as supervision. Instead of using projected LiDAR scans, previous methods for depth estimation [153] use structure-from-motion in the small overlap between cameras to obtain scale-aware pseudo depths for pretraining. Given the availability of time-synchronized images in the training set, we propose not to rely on the small overlap between cameras but to use all available images captured by the multi-camera setup for each scene to perform bundle adjustment. Without considering

extra sensors, recent research [59, 168] investigates using only images for training. These approaches leverage multiple temporally close RGB images jointly for supervision and consider multi-view consistency to train a depth estimator for the surrounding views, but do not exploit all available images taken of the whole scene. Some methods that use LiDAR scans [9, 113] or RGB images [59, 168] provide implicit supervision to learn the occupancy by optimizing for a proxy task like volume rendering [99].

In this chapter, we tackle the task of 3D semantic occupancy prediction using only RGB images for training, without relying on extra sensors such as LiDAR to provide depth. We propose to explicitly supervise for 3D semantic occupancy prediction while relying only on image information to better predict the full geometry and semantics of the scene, also behind occluded voxels as shown in Figure 7.1.

The main contribution of this chapter is a method to generate occupancy pseudo labels relying solely on images that we can use to supervise networks for 3D semantic occupancy and predict the full occupancy of the surrounding area. Compared to previous approaches that use only RGB images [34, 59, 168], instead of using a few sequential images for supervision, we propose to leverage all available images of the training set. We use bundle adjustment to align the images of each scene and compute the camera poses. We exploit this information to generate depth images to replace the LiDAR supervision. We furthermore generate triangle meshes and use them to filter out depth values belonging to occluded objects. We exploit a foundation model [127] to generate semantics for each image, which are added to our pseudo labels to provide both geometric and semantic supervision. Training with our generated pseudo labels, we achieve state-of-the-art performance while using only unlabeled RGB images for training.

In sum, we make two key claims: (i) without any labels, our approach achieves state-of-the-art performance on 3D semantic occupancy prediction among methods using only images for training; and (ii) our depth filtering using a triangular mesh improves the performance of 3D occupancy prediction.

The implementation and generated pseudo labels used in our approach are available at `https://github.com/PRBonn/SfmOcc`.

7.1 Vision-Based 3D Semantic Occupancy Prediction

7.1.1 Task Definition

Given a set of RGB images $\mathcal{I} = \{\boldsymbol{I}_{1,t}, \ldots, \boldsymbol{I}_{N,t}\}$ from a setup with N surrounding cameras recorded at timestep t, the objective is to predict the geometry and semantics of the surroundings as a dense 3D voxel grid $\tilde{\boldsymbol{O}} \in \mathbb{R}^{S_x \times S_y \times S_z \times C}$ where

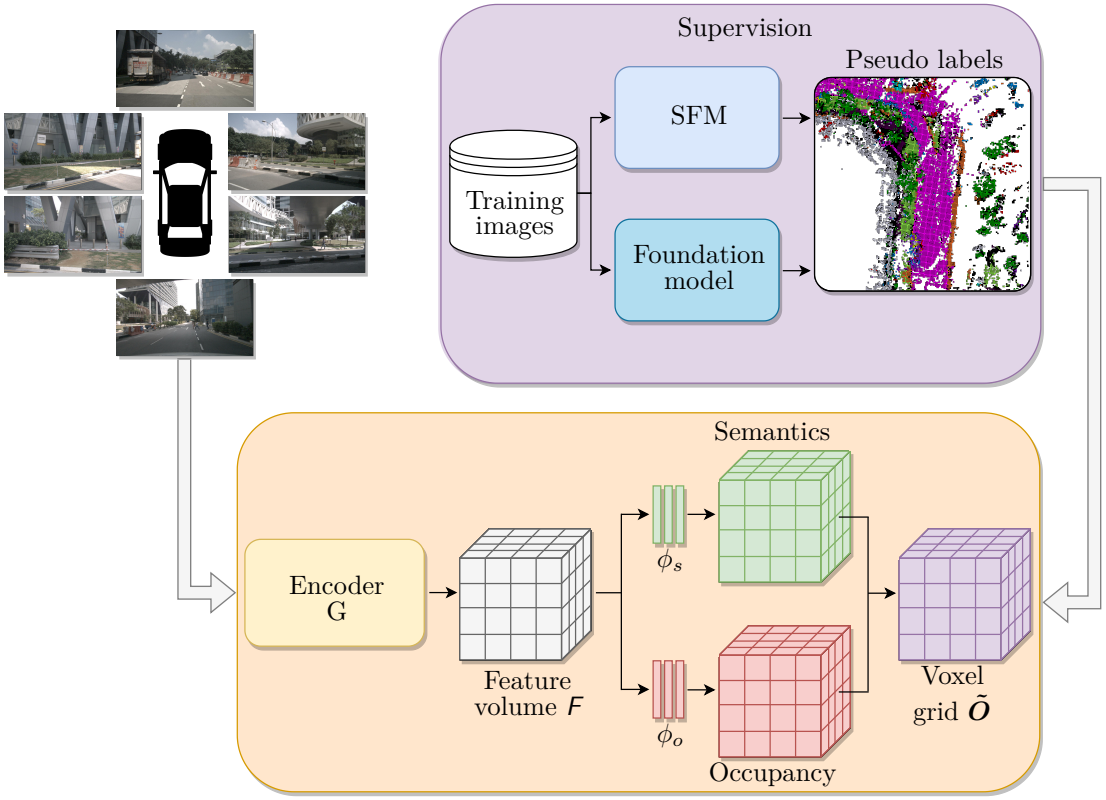
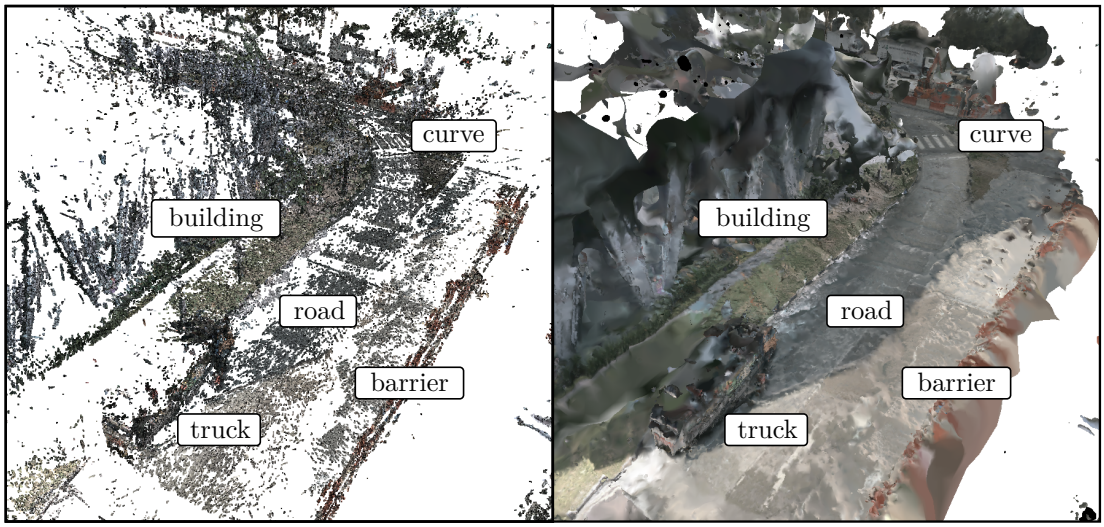


Figure 7.2: Overview of our approach called SfmOcc. Given a set of RGB images from a multi-camera setup, we extract a 3D volume feature F and use MLP heads ϕ_s and ϕ_o to predict semantic logits and occupancy probabilities for each voxel, which we fuse to obtain the final voxel grid $\tilde{O}$. For supervision, we leverage all available training images and use SFM and volume reconstruction to generate sparse occupancy pseudo labels. We use a foundation model to obtain semantic maps for the images and set the semantic class of each voxel. This way, we explicitly supervise our approach for semantic occupancy prediction while relying only on camera data.

S_x, S_y, S_z is the volume resolution and C is the number of semantic classes (including the "empty" class).

We show an overview of our approach, called SfmOcc, in Figure 7.2. We first extract 3D voxel features $F \in \mathbb{R}^{S_x \times S_y \times S_z \times D}$ from the input images $\mathcal{I}$ using a network G, where D is the feature dimension. The network G usually extracts 2D image features and projects them to 3D by either first predicting depth [113] or via attention [60]. From the voxel features F, we obtain semantic logits and occupancy probabilities using two MLP heads ϕ_s and ϕ_o and supervise the model using sparse occupancy pseudo labels. We obtain the final semantic occupancy predictions by assigning the predicted semantic class to the voxels predicted as occupied. Note that our method is not bound to a particular network G to perform occupancy prediction and could be used to train different models by replacing the supervision.



(a) Point cloud from sfm (b) Mesh from sfm

Figure 7.3: Point cloud (a) and triangle mesh (b) obtained from structure-from-motion for one scene. The scene depicts a parked truck on the left side of the road, construction barriers on the right, a building on the left, and a curvature at the end of the road.

To obtain pseudo labels for each scene, we use all images recorded by the camera setup at all timesteps $t \in \{t_1, \ldots, t_M\}$. We use bundle adjustment to align all $N \cdot M$ images in the scene, obtain camera poses, and then generate depth images as explained in the next section. We further use these depth images to perform volume reconstruction and obtain occupancy pseudo labels. We use a foundation model [127] to obtain 2D semantic maps for each image and generate semantic occupancy pseudo labels.

7.1.2 Depth Image Generation

We leverage all the available image information to obtain depth images that we can further use to generate occupancy pseudo labels. For each recorded scene, we use bundle adjustment [140] to align the RGB images captured by the N cameras at the M timesteps in the scene $\{\{I_{1,1}, \ldots, I_{N,1}\}, \ldots, \{I_{1,M}, \ldots, I_{N,M}\}\}$.

Given that the N cameras are rigidly attached, we optimize the poses $\mathsf{T}_{i,t} \in \mathbb{R}^{4\times4}$ of a single camera i for each timestep t and, for each other camera j, the transformation ${}^{j}\mathsf{T}_{i} \in \mathbb{R}^{4\times4}$ to camera i for the whole sequence. Furthermore, we fix the intrinsics K_i for each camera and do not optimize them, i.e., assuming a fixed, not-changing camera calibration. These two modifications allow us to optimize a smaller number of parameters and better constrain the adjustment, which in turn leads to more reliable results and is more robust to outliers from dynamic objects. This way, we obtain the pose of each camera, i.e., $\mathsf{T}_{j,t} = {}^{j}\mathsf{T}_{i}\mathsf{T}_{i,t}$ at each given timestep t in the scene. To simplify notation, when referring to a

camera i at a timestep t, we drop the subscript t.

Given the optimized camera poses T_i, we use multi-view stereo reconstruction [33, 119] to generate a sparse point cloud $\mathcal{P}$ for the given scene as shown in Figure 7.3 (a). We use the camera intrinsics K_i to project the point cloud $\mathcal{P}$ into each camera i and obtain a depth image $D_i^{\mathcal{P}} = \pi(\mathcal{P}, \mathsf{T}_i, \mathsf{K}_i)$, where π represents the projection of each point in the point cloud to the depth image, as shown in Figure 7.4.

Each depth image $D_i^{\mathcal{P}}$ provides depth for the reconstructed regions. Due to the sparsity of the point cloud $\mathcal{P}$, the obtained depth images $D_i^{\mathcal{P}}$ often have pixels with wrong depth belonging to occluded objects or areas. This can be seen in the point cloud obtained when unprojecting a depth image $D_i^{\mathcal{P}}$ to 3D, as shown in Figure 7.4.

7.1.3 Depth Filtering

As our point cloud $\mathcal{P}$ is sparse, points belonging to occluded objects are projected into the depth image $D_i^{\mathcal{P}}$, generating wrong depth values. If we obtain a closed surface without holes, we can avoid the projection of occluded points into the depth image. We can obtain such a surface by generating [50, 120] a triangle mesh $\mathcal{T}$ of the given scene using the poses from bundle adjustment to align the predicted depths from the multi-view stereo reconstruction. We show such a mesh in Figure 7.3 (b). Projecting the mesh we obtain the depth image $D_i^{\mathcal{T}} = \pi(\mathcal{T}, \mathsf{T}_i, \mathsf{K}_i)$ as shown in Figure 7.4. We can observe that the mesh generation introduces blob-like artifacts and problems in the boundaries, which prevents us from directly using the depth image $D_i^{\mathcal{T}}$ for supervision. However, we can use this depth image $D_i^{\mathcal{T}}$ to filter out pixels with wrong depth in $D_i^{\mathcal{P}}$ belonging to occluded areas.

We compute the residual between both depth images $\hat{D}_i = |D_i^{\mathcal{P}} - D_i^{\mathcal{T}}|$ and filter out pixels with residual larger than a threshold τ, which results in the filtered depth map D_i^f for image I_i:

$$D_i^f(u,v) = \begin{cases} D_i^{\mathcal{P}}(u,v) & \text{, if } \hat{D}_i(u,v) \leq \tau \\ 0 & \text{, otherwise} \end{cases}. \tag{7.1}$$

The depth image from the mesh $D_i^{\mathcal{T}}$ represents the closest surface to the camera and therefore the pixels (u, v) belonging to occluded objects have different depth values and a large residual in $\hat{D}_i$, which allows us to filter them out.

We show an example of the obtained depth image D_i^f in Figure 7.4. Given the filtering, the result is less dense than $D_i^{\mathcal{P}}$ but pixels with the wrong depth are discarded (as seen in Figure 7.4) by unprojecting the depth images into 3D.

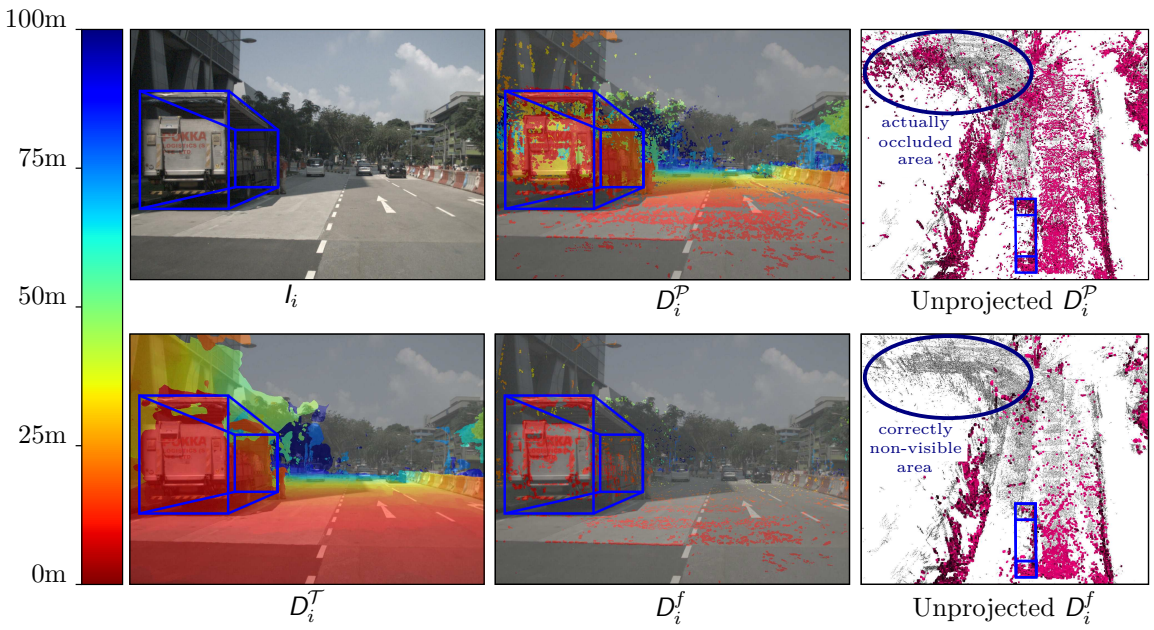


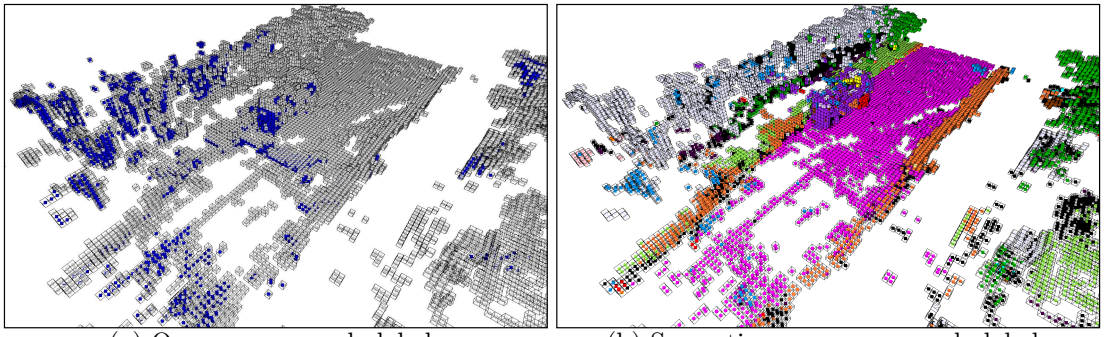
Figure 7.4: Filtering of depth images. We project the point cloud $\mathcal{P}$ into the image I_i and obtain a depth image $D_i^{\mathcal{P}}$. Due to the sparsity of the input point cloud, pixels associated with occluded points have wrong depth values. This is better visible when unprojecting the depth image to 3D. We show the full SFM point cloud for reference. We project the triangle mesh $\mathcal{T}$ into the image I_i to obtain a depth image $D_i^{\mathcal{T}}$ and use it to filter out pixels with wrong depth. The result is D_i^{f}. We unproject it to 3D to better show the filtering. We draw a bounding box around the truck for better visualization.

It is important to mention that during this process, also pixels with the correct depth are removed, making the supervision even sparser. This filtering helps the network to better learn the depth and to generate better occupancy pseudo labels, as discussed in Section 7.2.6.1.

7.1.4 Occupancy Pseudo Labels

The obtained depth images D_i^{f} can be used to replace the supervision in methods that use projected LiDAR for supervision [9, 113]. To explicitly supervise our method for occupancy, we can use the depth images along with the camera poses from bundle adjustment to generate sparse occupancy pseudo labels. Since we have depth images for the whole scene, we use the $N \cdot M$ depth images D_i^{f} to obtain a denser supervision, also behind occluded voxels in the current view.

We build a global voxel grid $\mathcal{V}_g \in \mathbb{R}^{H' \times W' \times D' \times B}$, where H', W', D' are the dimensions of the complete sequence and perform a similar operation to occupancy mapping [53]. For each ray of each image, we assume that the endpoint corresponds to a surface and that the line of sight between the camera center and endpoint is free space. To determine the voxels that need to be updated, we per-



(a) Occupancy pseudo labels (b) Semantic occupancy pseudo labels

Figure 7.5: Generated occupancy pseudo labels. We show in (a) the occupancy labels generated from a single (blue) and from all depth images in the scene (gray), and in (b), the corresponding semantic classes for each occupied voxel.

form a ray-casting operation to determine voxels along the ray from the camera center to the endpoint. We use a 3D variant of the Bresenham's algorithm [13] to approximate the ray and step through the voxel grid from the camera center to the endpoint of the ray. We set all traversed voxels as "empty" and the voxel at the end of the ray as "occupied". We show examples of our generated pseudo labels from one or all timesteps in Figure 7.5. With this approach, we combine all the different viewpoints of all cameras and obtain sparse occupancy pseudo labels for the whole scene, including free space and occluded voxels behind objects. This allows us to directly supervise occupancy probabilities instead of relying on proxy tasks like volume rendering [9, 59, 113].

7.1.5 Semantic Maps

To supervise for semantic occupancy, we extract semantic maps using a pre-trained open-vocabulary model [127]. This way, we obtain 2D semantic maps for each image without any 2D or 3D ground-truth labels. To add semantic information in the pseudo label generation, instead of setting voxels as "occupied" at the end of a ray, we accumulate in a histogram the class of the ray, which we obtain from the corresponding semantic map. At the end of the generation, we perform majority voting for each voxel to obtain the final semantic class. Given the inconsistency of the semantics predicted by the pre-trained model for the different views, the obtained labels may not be consistent for the different objects or surfaces, but still provide reasonable classes for the whole scene, as shown in Figure 7.5.

7.2 Experimental Evaluation

The main focus of this work is a method to perform 3D semantic occupancy prediction, which is trained using only RGB images. We leverage all available training images to generate sparse pseudo labels and explicitly supervise for occupancy. We present our experiments to show the capabilities of our method, and support our key claims: (i) our approach achieves state-of-the-art performance on 3D semantic occupancy prediction among methods using only images for training, and (ii) our depth filtering using a triangular mesh improves the performance of 3D occupancy.

7.2.1 Experimental Setup

We evaluate our method on Occ3D-nuScenes [139] dataset, which provides 3D semantic occupancy ground-truth for the images of nuScenes [14]. nuScenes provides 1000 driving scenes with six surrounding-view cameras. The occupancy ground-truth covers a range of $[-40, 40]$ m in x and y direction and $[-1, 5.4]$ m in z-direction with voxel size $\{0.4 \times 0.4 \times 0.4\}\, m^3$ meters and contains 17 semantic classes, the 16 classes of nuScenes plus an extra "empty" class. We use IoU to evaluate occupancy performance without considering semantics and mIoU to get the average across all non-empty semantic classes. Occ3D-nuScenes only provides labels for the training and validation set of nuScenes, but not for the test set. To avoid running experiments in the same set of 150 scenes (validation) where we evaluate our performance, we separate 60 scenes from the training set and use this set as our validation set. This way, we run our experiments and evaluate our performance in different subsets of the data.

7.2.2 Implementation Details

We use BEVStereo [83] as the network G to obtain the 3D voxel features $\boldsymbol{F}$. We predict 3D semantic occupancy for a voxel grid with resolution $S_x = 200$, $S_y = 200$, $S_z = 16$ and voxel size $\{0.4 \times 0.4 \times 0.4\}\, m^3$. We keep the original architecture and only add the semantic and occupancy heads ϕ_s, ϕ_o, which both consist of two linear layers and softplus activation layers. We follow the training strategy of RenderOcc [113], resize the image to 512×1408 pix and use AdamW [92] optimizer to train for 12 epochs with learning rate of 10^{-4} and batch size 8. We supervise both the semantic logits and the occupancy probabilities using cross entropy loss between predictions and pseudo labels. All experiments are conducted on 8 NVIDIA A40 GPUs.

We filter static sequences where the car moved less than 1 m and discard complete scenes where less than 75% of the images were successfully aligned. We

use the camera intrinsics K_i provided by the dataset and do not optimize them. To obtain semantic maps, we use Grounded SAM [127] and follow OccNerf [168] to prompt the model with multiple synonyms of each class name of the nuScenes dataset.

7.2.3 Pseudo Labels

We generate pseudo labels only for the training set. However, given that we use bundle adjustment, we do not consider scenes where the image alignment fails, like scenes where the vehicle does not move, too dark scenes, or scenes with too many dynamic objects. We obtain depth images and pseudo labels for 584 out of the 700 scenes in the training set and use 60 of the remaining scenes as validation set. The bundle adjustment and depth image generation take around 25 min per scene, while the occupancy pseudo-label generation takes around 2 min per scene. We generate a total of $127,458$ depth images, which we use to generate $21,231$ voxel-level sparse pseudo labels covering the surroundings of the vehicle with the same voxel grid and range as Occ3D-nuScenes. The generated pseudo labels contain 18 semantic classes, including the 17 of Occ3D-nuScenes, and an extra "uncertain" class, which represents occupied voxels with no class given by the pre-trained semantic model. We consider the voxels with this class for occupancy but not for semantic supervision.

7.2.4 3D Semantic Occupancy Prediction Performance

The first experiment evaluates the performance of our approach in the Occ3D-nuScenes [139] dataset and shows that our approach achieves state-of-the-art performance among methods supervised using only camera data. In Table 7.1, we compare against methods that use as supervision the ground-truth 3D labels (3D), LiDAR (L), and only cameras (C). Several methods concentrate on the semantic reconstruction and therefore do not report the IoU.

As can be seen, our approach SfmOcc surpasses previous camera-only methods by at least 5.8 percent points both in terms of IoU and mIoU. We are able to predict better 3D geometry, shown by our higher IoU compared to the baselines, and we even outperform methods trained with LiDAR supervision both in IoU and mIoU. We also train our same architecture with ground truth labels and denote it as BEVStereoOcc. Relying on images only, our approach SfmOcc is able to learn the free space, but drops in terms of mIoU, probably because of wrong semantic classes in the generated pseudo labels.

Method	Mode	GT Sem.	IoU [%]	mIoU [%]
OccFormer [172]	3D	✓	-	21.9
TPVFormer [60]	3D	✓	-	27.8
CTF-Occ [139]	3D	✓	-	**28.5**
BEVStereoOcc	3D	✓	**51.6**	27.7
TPVFormer [60]	L	✓	17.2	13.6
RenderOcc [113]	L	✓	**45.9**	23.9
OccFlowNet [9]	L	✓	-	**26.1**
SimpleOcc [139]	C	-	-	7.1
SelfOcc [59]	C	-	45.0	9.3
OccNeRF [168]	C	-	45.0	9.5
GaussianOcc [34]	C	-	-	9.9
LangOcc [8]	C	-	51.8	11.8
SfmOcc (ours)	C	-	**57.7**	**17.7**

Table 7.1: 3D semantic occupancy prediction performance on occ3D-nuScenes. In the column mode: 3D are methods trained with occupancy ground truth labels, L trained with LiDAR supervision, and C trained only with cameras. GT Sem. indicates the requirement of ground-truth semantic labels for supervision. We denote our model trained with ground-truth labels as BEVStereoOcc.

7.2.5 Performance in Scenes with Dynamic Objects

We generate occupancy pseudo labels relying on bundle adjustment with the static scene assumption, where we do not reconstruct dynamic objects. In this experiment, we compare the performance of different image-based methods on scenes with many dynamic objects. We select 30 scenes with more dynamic objects in the validation set of Occ3D-nuScenes and evaluate image-based methods in Table 7.2. Our approach outperforms other methods relying on images for training for all dynamic classes and on the average across all classes. Other methods that supervise with a few consecutive images also rely on multi-view geometry for supervision and therefore face the same problem.

7.2.6 Ablation Studies

We analyse the influence of our design choices on the performance of the approach, namely the depth filtering and different ways of generating pseudo labels. We evaluate our model in the 60 scenes that we separated from the training set. We indicate each experimental setup with capital letters (A), (B), etc. in Tabs. 7.3, 7.4 and 7.5.

Method	mIoU	bicycle	bus	car	cons. veh.	motorcycle	pedestrian	trailer	truck
SelfOcc [59]	10.5	0.1	6.6	13.2	0.0	0.4	2.4	0.0	7.7
GaussianOcc [34]	11.0	3.8	14.6	17.2	0.8	2.9	10.1	0.14	10.6
SfmOcc (ours)	**19.6**	**8.2**	**14.8**	**20.9**	**7.1**	**12.0**	**12.2**	**1.6**	**15.9**

Table 7.2: 3D semantic occupancy prediction performance on scenes with many dynamic objects in Occ3D-nuScenes. All metrics are reported in percent (%).

#	Depth supervision	IoU [%]	mIoU [%]
A	LiDAR	53.8	16.4
B	Depth images $\boldsymbol{D}^{\mathcal{P}}$	40.5	12.3
C	Filtered depth images $\boldsymbol{D}^{f}$	**46.6**	**14.0**

Table 7.3: Semantic occupancy prediction performance supervising only with depth images. We compare using depth images obtained by projecting LiDAR scans and by projecting the SFM point cloud $\mathcal{P}$ before $\boldsymbol{D}^{\mathcal{P}}$ and after $\boldsymbol{D}^{f}$ filtering.

7.2.6.1 Depth Filtering

In this experiment, we show the importance of the depth filtering to obtain depth supervision and generate the occupancy pseudo labels.

We first supervise only with depth images and the corresponding semantic maps. We follow RenderOcc [113] and randomly sample camera rays and use volume rendering to obtain their semantic and depth values. We show our results in Table 7.3. In (A), we get depth images by projecting the LiDAR using the calibration between sensors. Due to the sparsity of the LiDAR, we follow a common practice [113] and use a window of 7 consecutive LiDAR scans for a denser supervision. We achieve 16.4% mIoU and 53.8% IoU. In (B), we use the depth images $\boldsymbol{D}^{p}$ obtained by projecting the point cloud $\mathcal{P}$ into the image plane. These depth images contain pixels with wrong depth due to the sparsity of the obtained point cloud, as explained in Section 7.1.2 and as shown in Figure 7.4. Given the density of the supervision, we can train using the depth images for a single timestep, resulting in 12.3% mIoU and 40.5% IoU. In (C), we use the depth images $\boldsymbol{D}^{f}$ obtained after the depth filtering with the triangle mesh. We filter out pixels with wrong depth, as explained in Section 7.1.3 and shown in Figure 7.4. This filtering allows us to achieve 14.0% mIoU and 46.7% IoU, improving 1.7 percent points and 6.1 percent points respectively, and shows that we improve the performance.

For the next experiments, we train using only the sparse occupancy pseudo labels generated with the depth images as explained in Section 7.1.4 and show the results in Table 7.4. In (D), we use the depth images $\boldsymbol{D}^{\mathcal{P}}$ obtained by projecting

#	Pseudo labels	IoU [%]	mIoU [%]
D	Depth images $\boldsymbol{D}^{\mathcal{P}}$	60.7	18.7
E	Filtered depth images $\boldsymbol{D}^{f}$	**68.3**	**20.5**

Table 7.4: Semantic occupancy prediction performance supervising with pseudo labels from unfiltered and filtered depth images.

the point cloud $\mathcal{P}$ into the image plane, and in (E), we use the filtered depth images $\boldsymbol{D}^{f}$. Compared with Table 7.3, training with pseudo labels instead of only using depth images improves the IoU by at least 14 percent points and the mIoU by around 4 percent points. As shown in Table 7.4, our depth filtering helps to remove invalid depth values due to occlusions and improves the performance from 60.7 to 68.3 in terms of IoU and from 18.7 to 20.5 in terms of mIoU. This highlights the importance of filtering wrong depth values before generating the occupancy pseudo labels.

7.2.6.2 Pseudo Label Generation

In this experiment, we assess how the different ways of generating pseudo-labels presented in Section 7.1.4 influence the model performance.

In (F), we only use depth images for a given timestep, unproject them into a point cloud, and voxelize them. This way, we get supervision only for the occupied voxels as shown in Figure 7.5 (a) and only up to the first occupied voxel. The IoU reaches 38.44 due to the sparsity of the supervision and the fact that we can only supervise the occupied voxels.

In (G), we aggregate the point clouds from all the depth images for the scene and voxelize them. In this case, we obtain supervision for occupied voxels in the complete scene. This includes voxels occluded by objects, but which are visible from a different view, as shown in Figure 7.5 (b). Here, we only supervise occupied voxels and therefore the occupancy prediction performance is similar to (F), as shown by the IoU. However, since we have more voxels with semantic classes as supervision, the mIoU improves from 13.9 to 14.5.

In (H), we use the depth images from a single timestep to perform volume reconstruction and also set all the traversed voxels as "empty", as explained in Section 7.1.4. Here, we only have values for the voxels up to the first object. While the mIoU improves around 3 percent points with respect to (G), the IoU improves by 20 percent points with respect to (F) and (G). We argue that this is due to the supervision of empty voxels, which represent around 90% of the scene and help the model better understand the 3D geometry.

Finally, in (I), we perform volume reconstruction using all the depth images

#	Single image	All images	Volume rec.	IoU [%]	mIoU [%]
F	✓	-	-	38.4	13.9
G	-	✓	-	38.5	14.5
H	✓	-	✓	58.6	17.2
I	-	✓	✓	**68.3**	**20.5**

Table 7.5: Influence in semantic occupancy prediction performance supervising with different sparse pseudo labels. Either using a single or multiple images to generate them, and whether or not to use volume reconstruction to obtain supervision for the free space.

Pseudo labels	IoU [%]	mIoU [%]
Whole voxel grid	49.6	8.7
Visible area only	**50.4**	**15.7**

Table 7.6: Evaluation of our generated pseudo labels vs. the ground-truth voxel labels for the training set of Occ3D-nuScenes.

in the scene. This way, we obtain supervision for all voxels along the rays of each depth image in the scene. The IoU reaches 68.25 and mIoU 20.52 due to the supervision of both occupied and free voxels observed from the different points of view of the different cameras.

7.2.6.3 Evaluation of Pseudo Labels

To show the quality of our generated pseudo labels, we evaluate them against the ground-truth labels for the training set and show the results in Table 7.6. If we evaluate the whole voxel grid, our labels have an IoU of 49.6% and mIoU of 8.7% with respect to the ground-truth labels. During training, we do not consider the parts of the scene not seen by any camera for supervision, in which case our pseudo labels have an IoU of 50.4% and mIoU of 15.7%.

7.2.7 Qualitative Results

Finally, we compare qualitatively the semantic occupancy predictions of our approach SfmOcc with different state-of-the-art methods. Namely RenderOcc [113] trained using LiDAR and SelfOcc [59] supervised only with images [59]. We show predictions for each method in Figure 7.6.

Training only with LiDAR depth, RenderOcc [113] shows bleeding at the boundaries of the objects, which is usually observed in methods that perform

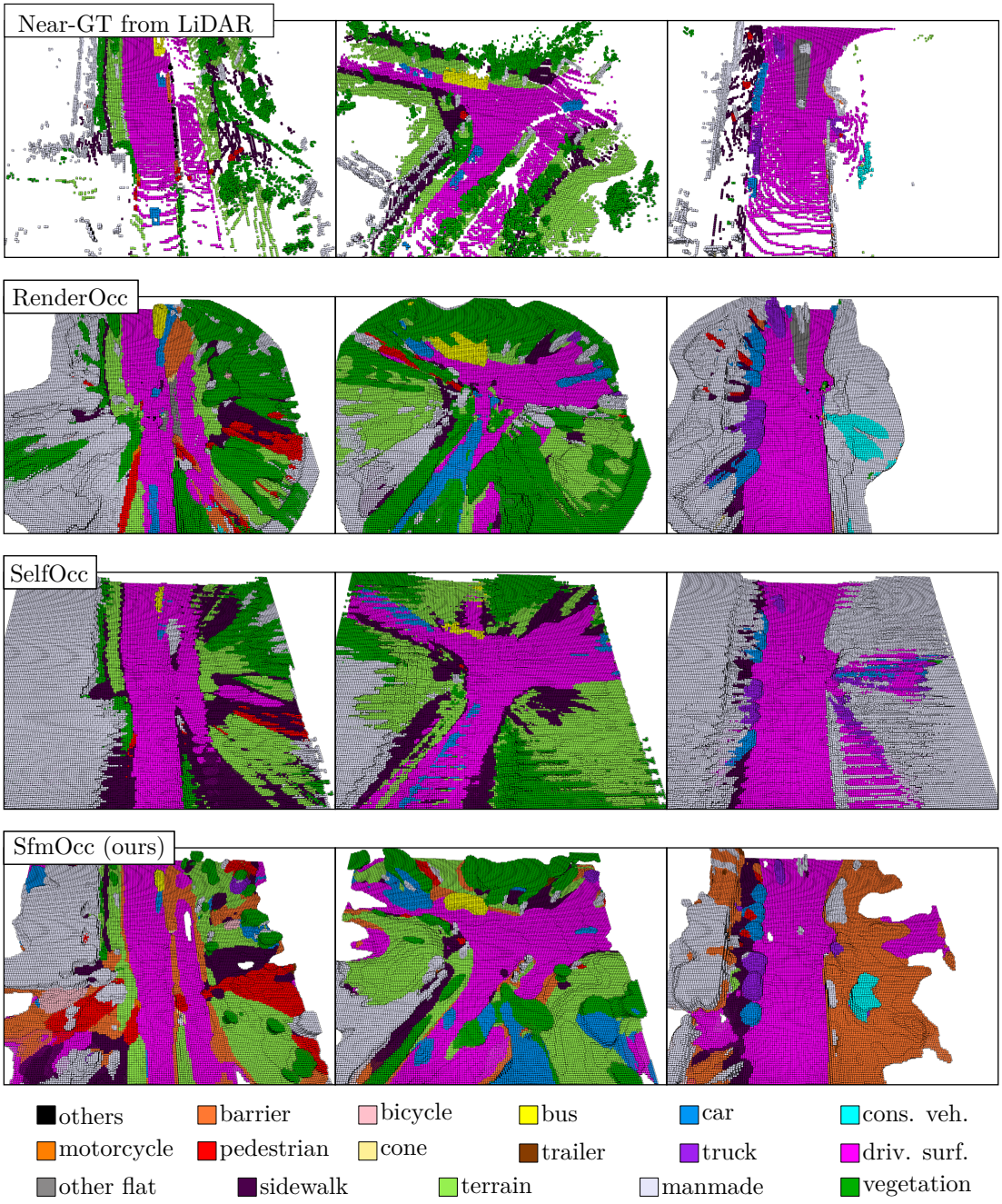


Figure 7.6: Qualitative results of our approach SfmOcc. We compare against approaches that use depth images (RenderOcc) or multiple RGB images (SelfOcc) as supervision. They show a bleeding effect similar to monocular depth estimation predictions, while our approach better learns the full shape of the objects, including the parts that are not visible from the camera.

mono-depth estimation. This is because they only supervise each camera ray up to the first object and therefore are not able to learn the whole shape of the objects. This is similar to methods that supervise only with RGB images, like SelfOcc [59], which provide, for each ray, the corresponding ray in other views

and leverage multi-view consistency, and supervise using photometric constraints. On the other hand, our approach relies only on images for training but provides sparse supervision for multiple voxels in the scene, including occluded voxels. This allows the model to better learn the geometry of the scene and the shape of objects, not only the depth, and leads to more complete results. Furthermore, although the semantics of the pseudo labels are sometimes not consistent within an object, the model is able to learn a single class for the whole object. Because we do not use ground truth semantics, our method sometimes predicts wrong semantic classes.

In summary, our evaluation shows that our method achieves state-of-the-art performance on semantic occupancy prediction among methods trained only with camera data, and that we are able to predict full occupancy instead of only performing depth estimation. However, due to the static surrounding assumption in our bundle adjustment system, we are currently not able to reconstruct dynamic objects in the pseudo label generation.

7.3 Conclusion

In this chapter, we presented a novel approach to generate occupancy pseudo labels for 3D semantic occupancy prediction using only vision data. Our method allows us to supervise using only camera data without the need for extra sensors such as LiDAR, and exploits the information from all the input images during training jointly. Without manual labeling and relying only on a foundation model for semantics, our approach achieves state-of-the-art performance among methods trained only with camera information and even competitive performance among methods trained with LiDAR. We implemented and evaluated our approach and provided comparisons to other existing techniques. The experiments suggest that we can use only images and structure-from-motion to generate supervision for occupancy estimation. Furthermore, we proposed a depth filtering method using a triangle mesh, which eliminates wrong depth values and improves the performance. In this chapter, we investigated how to tackle 3D spatial and semantic scene understanding relying only on RGB camera data. While most approaches assume the availability of a LiDAR sensor to generate occupancy labels or to provide accurate depth for supervision, autonomous systems might count on a limited sensor suite due to cost or hardware constraints. In this work, we explore how to exploit all the recorded RGB images only for supervision and output a segmented 3D representation of the scene.

With this approach, we are able to reconstruct the surrounding scene geometrically and semantically using only vision data. As a next step towards a more complete representation of the scene, we can include not only geometric

and semantic information but also instance-level information. In the next chapter, we investigate the task of 3D panoptic occupancy prediction, which requires reconstructing the surrounding scene and including semantic and instance information.

Chapter 8

Vision-Based 3D Panoptic Occupancy Prediction

As discussed in previous chapters, one of the key requirements for safe outdoor navigation is semantic scene understanding, which includes both 3D geometric and semantic information about the scene. In Chapter 7 we analyse how usually 3D perception tasks for scene understanding rely on 3D sensors like LiDAR, which intrinsically provide geometric information, and how alternatively vision-centric approaches aim to tackle scene understanding by relying only on RGB cameras. The goal of this part of the thesis is to tackle semantic scene understanding relying only on RGB images. RGB data provides rich color and texture information that helps in the semantic reasoning and foundation models [72, 91, 127] trained on large-scale annotated datasets [45, 72, 86] provide off-the-shelf solutions for image segmentation. However, this data modality lacks depth information to geometrically describe the scene, one of the requirements for semantic scene understanding. In the previous chapter, we focused on 3D semantic occupancy prediction using only RGB data, which provides a representation of the scene but lacks individual object information. As a next step into a more complete understanding of the scene, 3D panoptic occupancy prediction represents a vision-centric task that aims to represent the 3D geometry of the surroundings and provide semantic and instance information using only a setup of cameras covering the surroundings of the vehicle. State-of-the-art methods for 3D panoptic occupancy prediction [91, 150, 165] assume the availability of 3D voxel labels for supervision, which are challenging and expensive to obtain. As discussed in the previous chapter, this is also the case for a large number of methods that focus on 3D semantic occupancy prediction [60, 139, 172] while other methods [34, 113] rely on costly 3D sensors like LiDAR and use segmented scans for training. To avoid relying on 3D sensors, recent research [34, 35, 59, 168] focuses on using only RGB images for training.

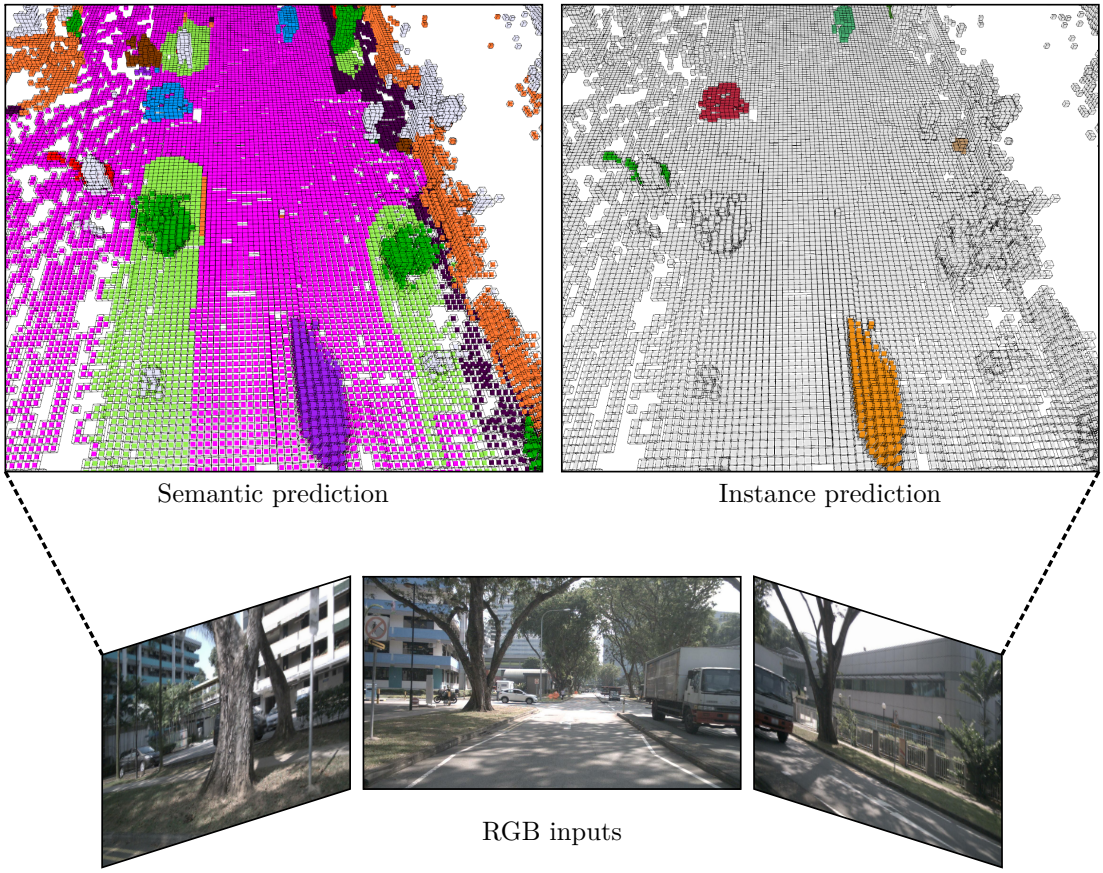


Figure 8.1: Our approach allows us to predict, from a set of RGB images, the occupancy of the surrounding scene along with the semantic classes and instances. We show with different colors the semantic classes on the left and the instance IDs on the right. See Figure 8.6 for the legend of class colors.

They use a few consecutive images from the same camera to enforce multi-view consistency and train a depth estimator. This means that they only provide information up to the first surface (and thus not for occluded areas) and they supervise for proxy tasks such as volume rendering [99]. To provide a more direct and efficient solution, we proposed, in the previous chapter, to not rely only on a few consecutive frames but to leverage all the available training images and generate semantic occupancy pseudo labels for explicit supervision. This allowed us to outperform all previous methods for 3D semantic occupancy prediction, relying only on images. However, this method only provides semantic information and does not include information about the individual traffic participants, which is fundamental for safe navigation of autonomous driving. Furthermore, given the static world assumption made in the bundle adjustment, the moving objects in the scene are not included in the pseudo labels, and therefore, the model is not able to learn dynamic objects in the environment.

In this chapter, we tackle the task of 3D panoptic occupancy prediction using

only RGB images for training. We provide explicit supervision while relying only on image information to predict the geometry of the whole scene alongside its semantic and instance segmentation, as shown in Figure 8.1.

The main contribution of this chapter is a novel method to generate panoptic occupancy pseudo labels relying on image data and foundation models. We can use this data to supervise neural networks for 3D panoptic occupancy. We leverage all images from the training set and run bundle adjustment to estimate camera poses and compute scale-aware depth images. However, in contrast to the work presented in Chapter 7, we use a pre-trained open-vocabulary segmentation model to obtain *panoptic* segmentation of each RGB image and combine them with the depth images to generate sparse *panoptic* occupancy pseudo labels, which we can use as explicit supervision. Furthermore, we additionally use a 3D foundation model to obtain dense depth predictions and combine them with the panoptic predictions to detect and add dynamic objects into the pseudo labels. This way, our model is able to predict not only semantics but can also predict instances and reconstruct dynamic objects in the scene. To the best of our knowledge, we provide the first method for 3D panoptic occupancy prediction using only unlabeled RGB images for training, and furthermore, we achieve state-of-the-art performance for vision-based 3D semantic occupancy prediction.

Our experiments suggest that: (i) we present the first image-based approach for 3D panoptic occupancy prediction, (ii) our approach achieves state-of-the-art performance on 3D semantic occupancy prediction among methods using only images for training, and (iii) our approach is able to predict dynamic objects in the scene. The implementation and generated pseudo labels used in our approach are available at `https://github.com/PRBonn/SfmPanOcc`.

8.1 Vision-Based 3D Panoptic Occupancy Prediction

8.1.1 Overview

The goal is to predict a semantic class and instance ID for each voxel in a dense 3D voxel grid $\tilde{\boldsymbol{O}} \in \mathbb{Z}^{S_x \times S_y \times S_z \times 2}$, where S_x, S_y, S_z is the volume size in x, y, and z directions, and we predict two values for each voxel. One value corresponds to the semantic class $c \in \{1, \ldots, C\}$, where C is the number of semantic classes (including the "empty" class), same as in the previous chapter. This value indicates whether the voxel is occupied and its semantic class, which can belong to both "stuff" or "things" [69]. The other predicted value is the instance ID, only valid for voxels with a predicted "thing" class. This way, we predict the geometry, semantics, and instances of the surrounding scene. Figure 8.2 shows the overview

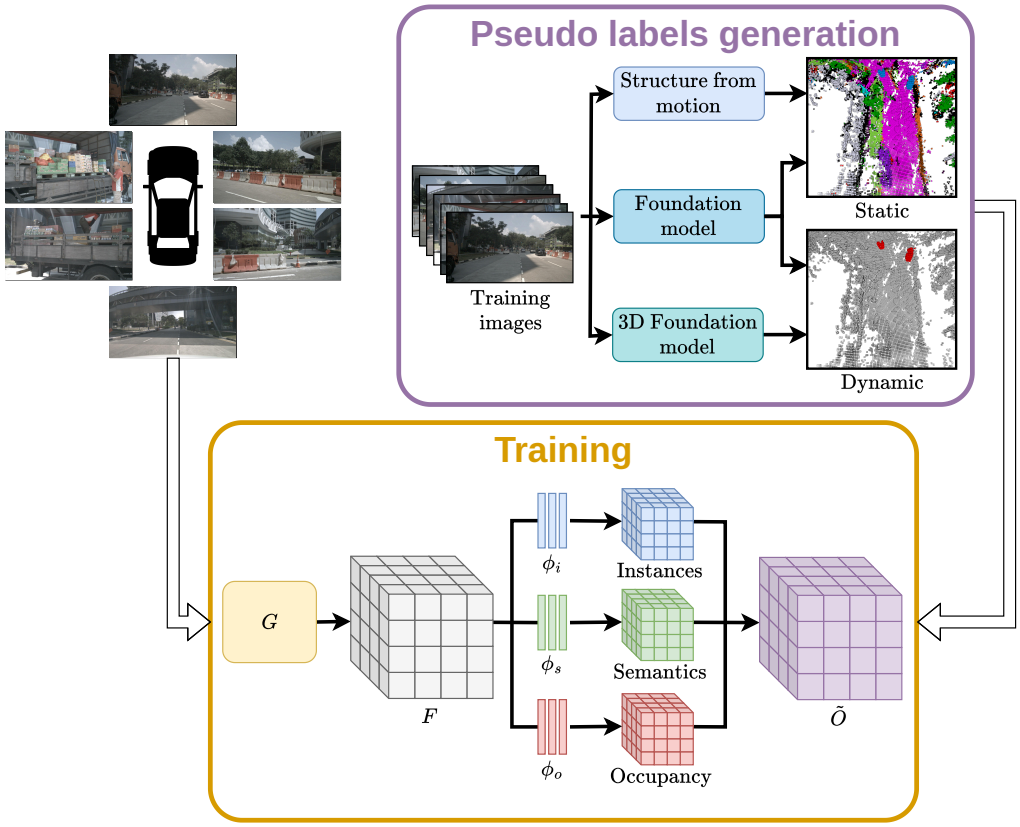


Figure 8.2: Overview of our method. At each timestep, the multi-camera setup captures a set of input RGB images from which we extract 3D voxel features F. Using three MLP heads ϕ_o, ϕ_s, and ϕ_i, we predict an occupancy value, semantic class, and instance ID for each voxel, which we fuse to obtain the final voxel grid $\tilde{\boldsymbol{O}}$. We use all available training images and use SFM and volume reconstruction to generate occupancy pseudo labels for the static scene. We leverage a 3D foundation model to add dynamic objects to the pseudo labels. We use a foundation model to segment the images and add the semantic class and instance ID to each voxel. We explicitly supervise for panoptic occupancy prediction while relying only on camera data.

of our approach called SfmPanOcc.

We use as input RGB images $\mathcal{I} = \{I_{1,t}, \dots, I_{N,t}\}$ captured at timestep t by a sensor setup consisting of N cameras. We employ a network G to extract 3D voxel features $\boldsymbol{F} \in \mathbb{R}^{S_x \times S_y \times S_z \times D}$ from the input images $\mathcal{I}$, where D is the feature dimension.

For each voxel, we predict occupancy probabilities, semantic logits, and offsets to the instance center by applying three different MLP heads ϕ_s, ϕ_o, and ϕ_i over the voxel features $\boldsymbol{F}$. We threshold the occupancy probabilities to obtain occupancy values for each voxel and use arg max over the logits to obtain semantic classes for the occupied voxels. To obtain instance IDs, we follow a

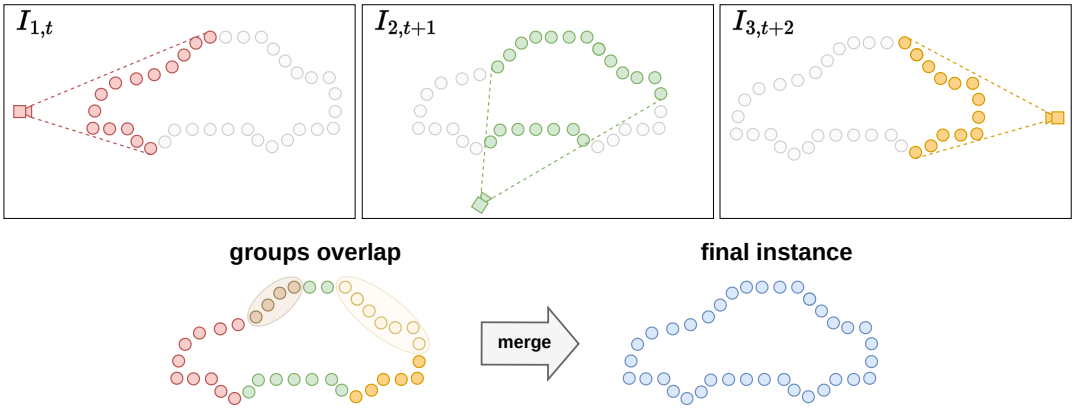


Figure 8.3: Instance merging process. We project the depth images from the different cameras and assign instance IDs to the projected points (top). Because IDs are not consistent across cameras, a single 3D instance has points with multiple IDs. We merge all groups of points based on their overlap and the distance between their centroids to obtain an instance with a single ID (bottom).

bottom-up approach by filtering voxels with predicted "stuff" class, shifting the 3D coordinates of the remaining voxels using the predicted offsets and using a clustering algorithm [26] to group voxels into instances. Note that the proposed method can use different networks G to extract 2D features and project them to 3D via attention [60] or via depth prediction [113].

To train the approach, we obtain pseudo labels for each scene using all the images captured by the N cameras at all M timesteps. We first use bundle adjustment to align the $N \cdot M$ images and then generate depth images (Section 8.1.2), with which we perform volume reconstruction and obtain occupancy pseudo labels. We use a foundation model [127] to segment the 2D images and obtain semantic classes and instance IDs. We utilize this information to generate *panoptic* occupancy pseudo labels (Section 8.1.3). Additionally, we use a 3D foundation model to obtain dense depth images and leverage them to detect and add dynamic objects into our pseudo labels (Section 8.1.4).

8.1.2 Depth Image Generation

To generate our occupancy pseudo labels, we first need to compute depth images. Following the same procedure described in Chapter 7, we use all RGB images captured by the N cameras at all M timesteps for a given scene and use bundle adjustment [140] to align these RGB images $\{\{I_{1,1}, \dots, I_{N,1}\}, \dots, \{I_{1,M}, \dots, I_{N,M}\}\}$. We assume rigidly attached cameras and constant intrinsic parameters K_i of each camera i.

With the obtained camera poses for each camera at each given timestep in the scene, we use multi-view stereo reconstruction [33, 119] to generate a sparse

point cloud that we project into each camera i using the intrinsic parameters K_i to obtain depth images. Furthermore, we generate a triangle mesh and use it for depth filtering as described in the previous chapter. Given the extrinsic parameters provided by the dataset, this allows us to obtain, for each camera i at timestep t, a scale-aware sparse depth image $\mathsf{D}^f_{i,t}$.

8.1.3 Panoptic Occupancy Pseudo Labels

We use the sequence of images $\{\mathsf{I}_{i,1}, \dots \mathsf{I}_{i,M}\}$ recorded by each camera i as the input to a foundation model, in our case, GroundedSAM2 [127], to obtain the semantic and instance segmentation for each image, where the instance IDs are consistent over time. This allows us to obtain panoptic segmentation for the M images captured by each camera. We project the depth images to 3D using the camera parameters K_i to obtain point clouds and assign to each point, the semantic class and an instance ID from the panoptic segmentation prediction, as shown in Figure 8.3 (top). We do the same for the N cameras and aggregate all the point clouds from all $N \cdot M$ images in the sequence to obtain a single point cloud $\mathcal{P} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_J\}$ where each point $\boldsymbol{p}_j = (\boldsymbol{x}_j, l_j)$ consists of 3D coordinates $\boldsymbol{x}_j$ and instance ID l_j and J is the total number of points.

The instance IDs of points belonging to the same instance might be different for points projected from different cameras, as shown in Figure 8.3 (bottom). To obtain a single instance ID, we merge groups of points based on their overlap. Given a group of points with the same instance ID $\mathcal{P}_l = \{\boldsymbol{p}_j \in \mathcal{P} \mid l_j = l\}$, we count for each point $\boldsymbol{p}_j \in \mathcal{P}_l$ the number of neighbors (nn) in the other group:

$$\mathrm{nn}(\mathcal{P}_l, \mathcal{P}_{l'}) = \left|\left\{\boldsymbol{p}_j \in \mathcal{P}_l \;\middle|\; \exists\, \boldsymbol{p}_k \in \mathcal{P}_{l'} \text{ s.t. } \|\boldsymbol{x}_j - \boldsymbol{x}_k\|_2 < \tau_n\right\}\right|. \quad (8.1)$$

We define the symmetric overlap as:

$$\mathrm{overlap}(\mathcal{P}_l, \mathcal{P}_{l'}) = \frac{\mathrm{nn}(\mathcal{P}_l, \mathcal{P}_{l'}) + \mathrm{nn}(\mathcal{P}'_l, \mathcal{P}_l)}{|\mathcal{P}_l| + |\mathcal{P}_{l'}|}. \quad (8.2)$$

We merge groups $\mathcal{P}_l$ and $\mathcal{P}_{l'}$ if the symmetric overlap between them is larger than a threshold τ_o. This allows us to remap the instance IDs in each image to be consistent across all cameras and timesteps in the scene.

To generate pseudo labels, we build a global voxel grid and perform a similar operation to occupancy mapping [53]. We perform a ray-casting operation to determine the voxels along each ray of each camera and use a 3D variant of the Bresenham's algorithm [13] to step through the voxel grid from the camera center to the endpoint of the ray. We set all traversed voxels as "empty" and the voxel at the end of the ray as "occupied". We accumulate in a histogram the different semantic classes and instance IDs for each voxel and obtain the final values via majority voting. As a result, we obtain panoptic occupancy pseudo labels from images, as shown in Figure 8.4, relying on a foundation model.

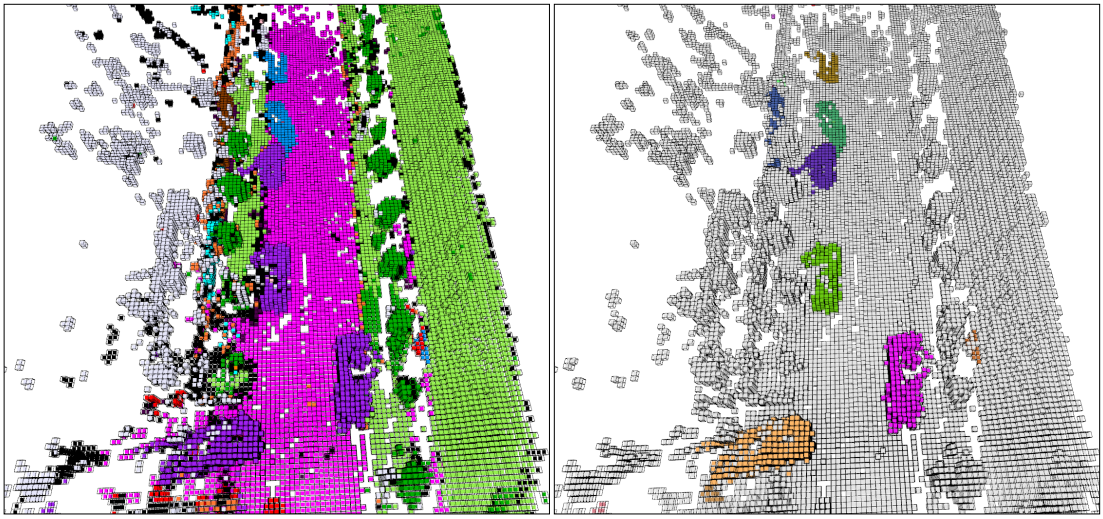


Figure 8.4: Generated panoptic occupancy pseudo labels. We show the semantic classes for each voxel with different colors on the left. On the right, we show with colors the different instance IDs.

8.1.4 Pseudo Labels for Dynamic Objects

Given the static scene assumption made in the bundle adjustment, the depth images $\boldsymbol{D}^f_{i,t}$ do not contain depth for pixels belonging to dynamic objects and therefore dynamic objects are not present in the pseudo labels.

To predict depth also for moving objects, we use a 3D foundation model, in our case, MonST3R [170], and input the sequences of images $\{I_{i,1}, \ldots, I_{i,M}\}$ captured by each camera i. Given two input images, MonST3R can produce dense, per-pixel 3D point clouds along with confidence maps in a single feed-forward pass, and further estimate depth images, camera intrinsics, and camera poses based on the inferred geometry. From these outputs, we use the dense up-to-scale depth predictions $D^p_{i,t}$ and confidence maps $C^p_{i,t}$. The predicted depth images $D^p_{i,t}$ include artifacts and are noisier than $\boldsymbol{D}^f_{i,t}$ due to the low resolution of the predictions. However, they contain depth for the dynamic objects, as shown in Figure 8.5, which we can use to add moving objects into our pseudo labels.

We scale the depth images $D^p_{i,t}$ by estimating a scale factor s^* and scale bias b^* using the depth from structure-from-motion $D^f_{i,t}$, given by:

$$s^*, b^* = \arg\min_{s^*,b^*} M^p_{i,t} \cdot M^f_{i,t} \| D^p_{i,t} - s^* D^f_{i,t} + b^* \|_2, \tag{8.3}$$

where $\cdot$ is the per-element product per matrices, $M^p_{i,t} = \mathbb{I}[C^p_{i,t} > \tau_c]$ is the binary mask of pixels with confidence larger than τ_c, and $M^f_{i,t} = \mathbb{I}[D^f_{i,t} > 0]$ is the valid sparse depth mask. In this setup, Equation (8.3) represents a least squares problem that can be solved in closed form.

For each input image, we obtain the scaled depth prediction $D^{p'}_{i,t} = s^* D^p_{i,t} + b^*$ and use the panoptic segmentation to filter out "stuff" pixels. We project $D^{p'}_{i,t}$

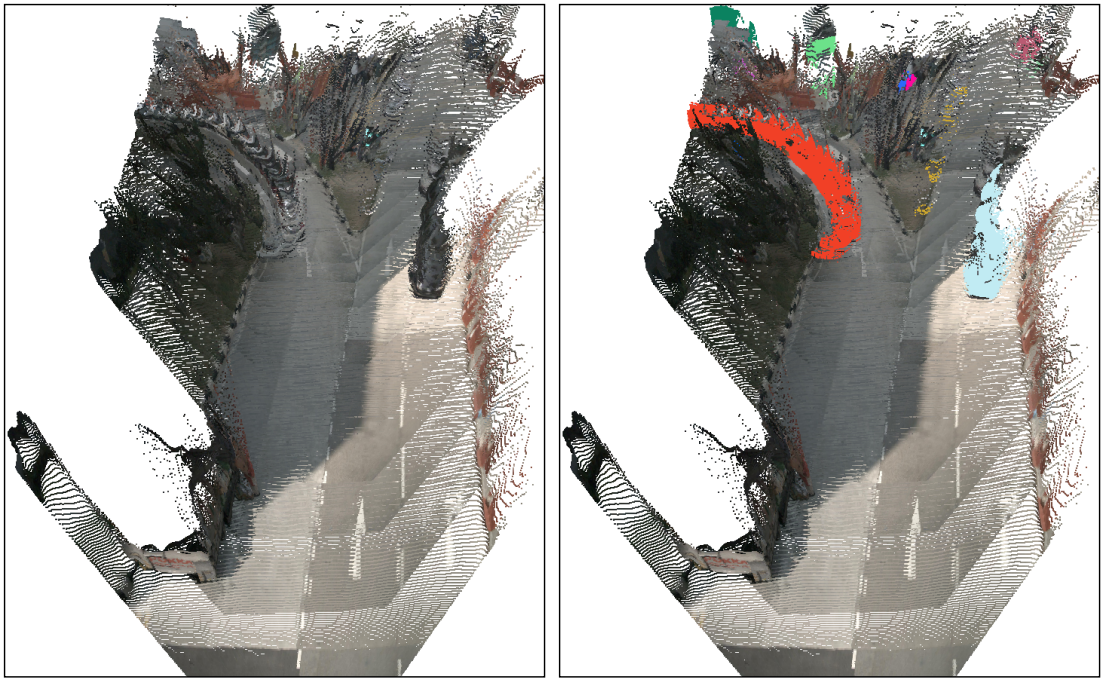


Figure 8.5: Depth predictions using 3D foundation model and dynamic objects detection. We show the projected point clouds from the depth predictions for a sequence of images. On the left, we show the aggregated point cloud of the scene from above. On the right, we further highlight the detected dynamic objects.

into a 3D point cloud where each point has a corresponding instance ID. For each group of points with the same instance ID $\mathcal{P}_l = \{\boldsymbol{p}_j \in \mathcal{P} | l_j = l\}$, we compute its centroid $\boldsymbol{c}_{l,t} \in \mathbb{R}^3$ at each time step t as the mean of the 3D point coordinates.

After processing all M timesteps, we classify instances as moving if the centroid displacement exceeds a threshold τ_m:

$$\max_t \|\boldsymbol{c}_{l,t} - \boldsymbol{c}_{l,t-1}\|_2 > \tau_m. \tag{8.4}$$

This way, we detect moving objects in each individual image and use this information together with the scaled depth prediction $D^{p'}_{i,t}$ to add dynamic objects to the panoptic occupancy pseudo labels at each individual timestep as described in Section 8.1.3. With this procedure, we add dynamic objects into our generated pseudo labels using image data and relying on a 3D foundation model for depth prediction.

8.2 Experimental Evaluation

The main focus of this work is a method for 3D panoptic occupancy prediction, which is trained relying only on image data. We generate sparse pseudo labels using all the available training images and a foundation model to add semantics

and instances. We leverage a 3D foundation model to detect and add dynamics to the pseudo labels. We present our experiments to show the capabilities of our methods, and show that: (i) we present the first image-based approach for 3D panoptic occupancy prediction, (ii) our approach achieves state-of-the-art performance on 3D semantic occupancy prediction among methods using only images for training, and (iii) our approach is able to predict dynamic objects in the scene.

8.2.1 Experimental Setup

In this section, we evaluate our model using the same dataset as in Chapter 7, Occ3D-nuScenes [139], which builds on top of nuScenes [14] and provides 3D voxel labels for semantic occupancy prediction. nuScenes consists of 1000 driving scenes captured by six cameras covering the complete 360° of the vehicle. The occupancy ground-truth covers a range of $[-40, 40]$ m in x and y direction and $[-1, 5.4]$ m in z-direction with a voxel size of $\{0.4 \times 0.4 \times 0.4\}\, m^3$ and containing the 16 semantic classes of nuScenes plus an extra "empty" class. Compared to the previous chapter, we need voxel-level panoptic labels to evaluate our method. We obtain those labels, following SparseOcc [91] to combine the bounding boxes provided by nuScenes and the voxel-level semantic labels. We evaluate the occupancy performance using IoU and consider also the semantics by computing the mIoU across all classes. For panoptic segmentation performance, we use voxel-level panoptic quality (PQ) [69]. Given that Occ3D-nuScenes only provides labels for two sets: training and validation sets of nuScenes. We use the same split defined in the previous chapter and separate 60 scenes from the training set to use as our validation set. This allows us to run experiments in 60 scenes and evaluate performance in 150 scenes, therefore using different subsets of data.

8.2.2 Implementation Details

The voxel grid has a size $S_x = 200, S_y = 200, S_z = 16$ with voxel size $\{0.4 \times 0.4 \times 0.4\}\, m^3$. We use BEVStereo [83] as the network G to extract the 3D voxel features F and keep most of the original architecture, only adding the occupancy, semantic, and instance heads. The occupancy and semantic heads ϕ_o and ϕ_s consist of a single linear layer and softplus activation function. The instance head ϕ_i consists of a stack of 3D convolutions and ReLU activation functions followed by a linear layer. We follow RenderOcc [115] and use AdamW [92] optimizer with learning rate of 10^{-4} and train for 12 epochs with batch size 8. We use the cross entropy loss between predictions and pseudo labels for occupancy and semantics, and L2 loss for the offset predictions. We conduct all experiments on 8 NVIDIA A40 GPUs.

We use Grounded SAM2 [127] to obtain semantic maps and also obtain consistent instance IDs for the whole sequence of images. We prompt the model with synonyms of the class names of the nuScenes dataset, as proposed in OccNerf [168]. To generate the panoptic labels, we use $\tau_n = 0.1\,m$ to count neighbors, and $\tau_o = 0.1$ for the overlap. To obtain the dense depth predictions for the dynamic objects, we use MonST3R [170] with default parameters. For the scaling, we use $\tau_c = 0.5$ to filter out points with low confidence and $\tau_m = 1\,m$ to detect moving objects.

8.2.3 Pseudo Labels

We generate pseudo labels to train our approach for panoptic occupancy prediction and therefore use only images from the training set of nuScenes. For some scenes, bundle adjustment fails to align the images due to several reasons, such as too dark scenes, scenes with too many dynamic objects, or sequences where the vehicle does not move enough. In total, we generate depth images and pseudo labels for 584 out of the 700 scenes in the training set. We produce 127,458 depth images with which we generate 21,231 sparse panoptic occupancy pseudo labels with the same voxel grid and range as Occ3D-nuScenes. Our generated labels contain the 17 classes included in Occ3D-nuScenes and an extra "uncertain" class for pixels with no given semantic class by the pre-trained open-vocabulary segmentation model.

8.2.4 3D Panoptic Occupancy Prediction Performance

In this experiment, we evaluate our approach in the task of 3D panoptic occupancy prediction and compare it with previous approaches in Table 8.1. Given that we propose the first method for this task relying only on camera data for supervision, we compare it against existing state-of-the-art methods that train using ground-truth voxel labels, namely SparseOcc [91] and Panoptic-FlashOcc [165]. Our approach achieves 58.2% IoU, better than previous methods that even rely on voxel labels. Regarding the semantic segmentation performance, our approach achieves 17.9% compared with 31.6% of the state-of-the-art approach, but without manual annotations for semantic classes. In terms of panoptic quality, our approach achieves 11.5% against the 15.8% of the best-performing method. Overall, our method shows strong performance given the fact that it is not trained with ground truth depth, occupancy, instances, or semantic classes.

Method	Supervision	IoU [%]	mIoU [%]	PQ[%]
SparseOcc [91]	3D	37.7	30.9	13.0
Panoptic-FlashOcc [165]	3D	43.3	**31.6**	**15.8**
SfmPanOcc (ours)	C	**58.2**	**17.9**	**11.5**

Table 8.1: 3D *panoptic* occupancy prediction performance on occ3D-nuScenes. In the column mode, 3D are methods trained with occupancy ground truth labels, and C trained only with cameras.

8.2.5 3D Semantic Occupancy Prediction Performance

In this experiment, we evaluate the performance of our approach in the task of 3D semantic occupancy prediction. We show the results in Table 8.2, where we compare against methods that supervise using ground-truth 3D voxel labels (3D), LiDAR (L), and only camera data (C). Several works presented in the table focus only on semantic reconstruction and therefore do not report IoU. Our approach outperforms previous approaches that rely only on camera data, both in terms of IoU and mIoU, even outperforming a method using LiDAR for supervision. While relying only on images, our approach is able to better learn the geometry of the scene, depicted by the high IoU, compared to the previous camera-based methods, but has a lower performance in terms of mIoU compared with methods that use ground-truth semantics. We believe that the reason for this drop in performance might be the wrong semantic classes predicted by the foundation model, and later assigned to our pseudo labels.

8.2.6 Performance in Scenes with Dynamic Objects

We generate our occupancy pseudo labels in two steps. First, we use bundle adjustment assuming a static scene and do not reconstruct dynamic objects, as done in the previous chapter. In a second step, we use a 3D foundation model to obtain dense depth predictions and add dynamic objects to the pseudo labels. To show the advantage of our approach, we evaluate the semantic occupancy prediction performance of image-based methods on 30 scenes with many dynamic objects, as done by SfmOcc. We show the results in Table 8.3. Previous methods either supervise using a few consecutive images [34, 59] or all the available images in the scene, relying on multi-view geometry. Therefore, their supervision does not consider dynamic objects. Our approach, in contrast, includes dynamic objects and outperforms all previous methods in terms of mIoU. Compared with SfmOcc, which provides labels only for static objects, our approach improves the performance for commonly seen semantic classes such as bus, car, and pedestrian, but

Method	Supervision	GT Sem.	IoU [%]	mIoU [%]
OccFormer [172]	3D	✓	-	21.9
TPVFormer [60]	3D	✓	-	27.8
CTF-Occ [139]	3D	✓	-	**28.5**
TPVFormer [60]	L	✓	17.2	13.6
RenderOcc [113]	L	✓	**45.9**	23.9
OccFlowNet [11]	L	✓	-	**26.1**
SimpleOcc [35]	C	-	-	7.1
SelfOcc [59]	C	-	45.0	9.3
OccNeRF [168]	C	-	45.0	9.5
GaussianOcc [34]	C	-	-	9.9
LangOcc [10]	C	-	51.8	11.8
SfmOcc	C	-	57.7	17.7
SfmPanOcc (ours)	C	-	**59.2**	**18.4**

Table 8.2: 3D *semantic* occupancy prediction performance on occ3D-nuScenes. In the column supervision: 3D are methods trained with occupancy ground truth labels, L trained with LiDAR supervision, and C trained only with cameras. GT Sem. indicates the usage of ground-truth semantic labels for supervision.

Method	mIoU	bicycle	bus	car	cons. veh.	motorcycle	pedestrian	trailer	truck
SelfOcc [59]	10.5	0.1	6.6	13.2	0.0	0.4	2.4	0.0	7.7
GaussianOcc [34]	11.0	3.8	14.6	17.2	0.8	2.9	10.1	0.14	10.6
SfmOcc	19.6	**8.2**	14.8	20.9	**7.1**	12.0	12.2	**1.6**	**15.9**
SfmPanOcc (ours)	**20.3**	7.2	**25.5**	**23.6**	6.2	**15.0**	**13.3**	0.8	15.0

Table 8.3: 3D semantic occupancy prediction performance on scenes with many dynamic objects in Occ3D-nuScenes. All metrics are reported in percent (%).

slightly decreases the performance for the classes that are easier to confuse, such as construction vehicles, trailers, and trucks. We argue that the problem lies in the semantic classes predicted by the foundation model. For the static parts, we obtain the class via majority voting over the semantic maps for multiple images and even from different cameras. In contrast, we rely on a single image and its semantic map to add moving objects, which is more susceptible to wrong semantic predictions.

8.2.7 Ablation Studies

In this section, we conduct experiments to evaluate different strategies to generate pseudo labels and to compare the performance against using the ground truth

#	Depth images	IoU [%]	mIoU [%]
A	Sfm	**68.3**	20.5
B	MonST3R (static)	51.4	11.0
C	MonST3R (static) + dynamics	52.7	11.4
D	Sfm + dynamics	66.5	**21.2**

Table 8.4: Comparison of generated labels using different depth images from (A) bundle adjustment, (C) MonST3R considering only the static parts, (D) MonST3R static scene and adding dynamics, and (E) static part from bundle adjustment and dynamics

labels. We evaluate our model in the 60 scenes that we separate as the validation set and indicate each experimental setup as (A), (B), etc. in Table 8.4 and Table 8.5. Furthermore, we compare the generated pseudo labels against the ground truth labels.

8.2.7.1 Different Depth Images

In this experiment, we train our model with labels generated using different depth images and show the semantic occupancy prediction performance in Table 8.4 on the validation set.

In setup (A), we use the pseudo labels generated using the depth images from bundle adjustment. The achieved performance is 68.3% IoU and 20.5% mIoU. In our approach, we use a 3D foundation model to obtain dense depth predictions that are up-to-scale. As another option, we scale them with the depth images from bundle adjustment as described in Section 8.1.4 and generate occupancy pseudo labels. However, using these depth images would generate a trace of moving objects as shown in Figure 8.5. Therefore, in setup (B), we first detect dynamic objects as described in Section 8.1.4 and remove them to consider only the static background. The achieved performance is 51.4% IoU and 11.0% mIoU, which shows that although the 3D foundation model provides denser depth predictions, the quality of the bundle adjustment images is better suited to generate pseudo labels. In setup (C), we add the dynamic objects into the pseudo labels of setup (B) and increase the IoU and mIoU by 2.3 and 0.4 percent points, respectively, which shows the advantage of adding moving objects. Finally, in setup (D), we combine the static scene from bundle adjustment and the dynamic objects from the 3D foundation model prediction to generate pseudo labels. The performance is 66.5% IoU and 21.2%mIoU, which shows, despite the drop in IoU, how combining the best of both methods to generate pseudo labels leads to better learning the semantics of the scene.

#	Labels	IoU [%]	mIoU [%]	PQ [%]
E	Pseudo labels	**66.5**	21.2	15.0
F	Ground truth	61.4	**48.9**	**33.4**

Table 8.5: Performance comparison when training with ground truth labels vs. our generated pseudo labels on the validation set.

8.2.7.2 Comparison with Ground Truth Labels

In this experiment, we compare the performance of our model when using ground truth labels vs. our generated pseudo labels on the validation set and show the performance in Table 8.5. The model trained with pseudo labels (E) achieves a better occupancy prediction in terms of IoU. However, the difference in performance of 27.7 and 18.4 percent points in mIoU and PQ in (F) shows that despite relying only on image information, the quality of our pseudo labels is not enough to close the gap with methods trained with ground truth labels.

8.2.7.3 Quality of the Pseudo Labels

To show the quality of our generated panoptic occupancy pseudo labels, we compute the IoU, mIoU, and PQ to compare our pseudo labels with the ground truth labels for the training set of Occ3D-nuScenes. Our pseudo labels achieve 51.8% IoU, 15.6% mIoU, and 9.1% PQ. As can be seen, even the labels present a performance gap compared to the methods trained using the ground truth labels, particularly in the mIoU, with a gap of 16 percent points and 6.7 percent points in PQ. However, our labels surpass previous methods by 8.5 percent points in terms of IoU, which evaluates the occupancy estimation regardless of the semantic classes. This shows how our approach for pseudo label generation using only images can effectively recover the geometry of the scene, but struggles with assigning the corresponding semantic information. Improving the semantics in the pseudo label generation would enhance the occupancy prediction performance and would be an interesting direction for future work.

8.2.8 Qualitative Results

In Figure 8.6, we provide qualitative results of our proposed approach and SfmOcc (proposed in the previous chapter) in comparison. Due to the supervision of voxels in the whole scene, our approach is able to reconstruct the whole shape of objects, including occluded voxels. Different from SfmOcc presented in Chapter 7, our method is able to predict dynamic objects in the scene, as highlighted in the images. Furthermore, our method is able to predict not only the semantics of

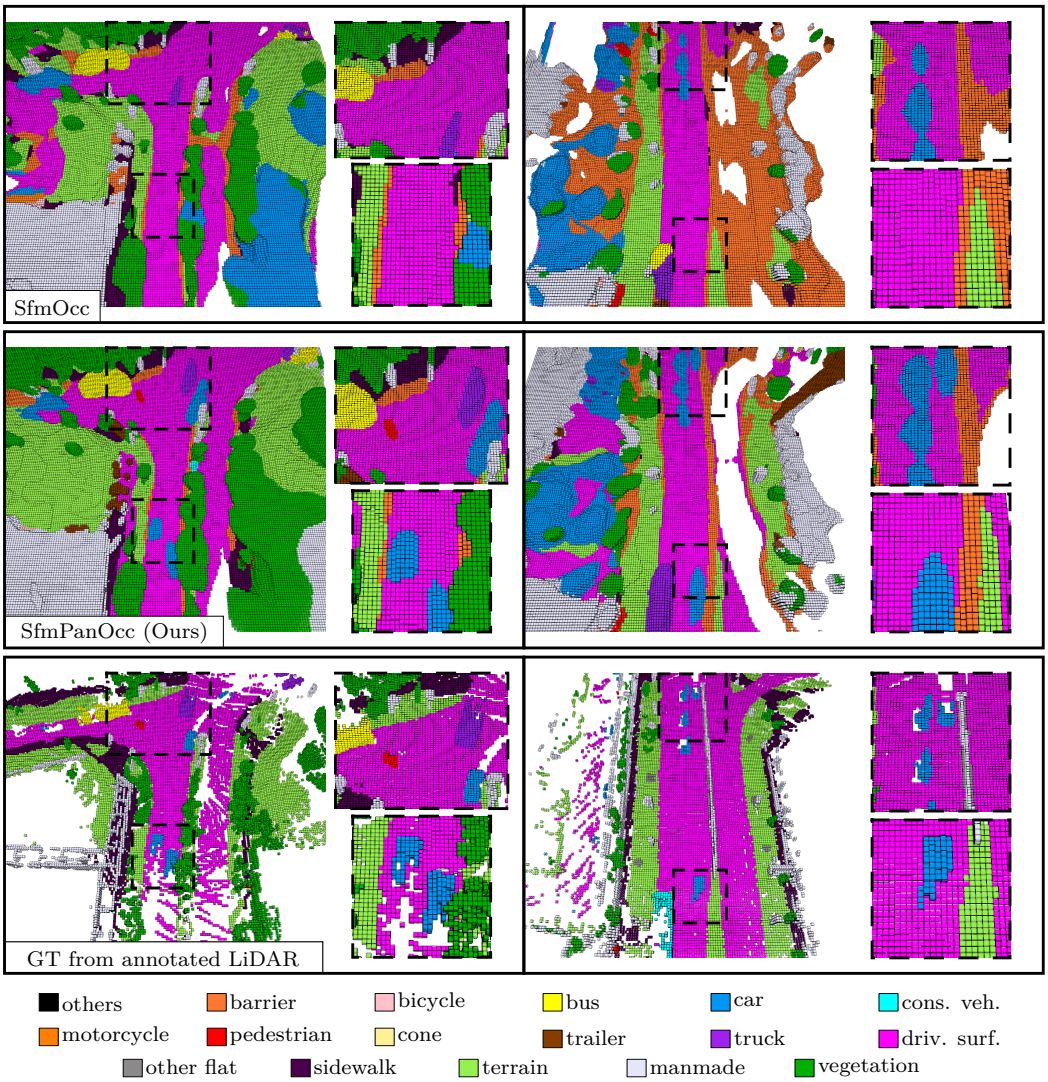


Figure 8.6: Qualitative results of our approach against SfmOcc, the best previous approach using images for supervision. We highlight areas containing dynamic objects. SfmOcc does not include dynamic objects in the supervision, and it is able to predict only a few of them. In comparison, our approach is better able to predict dynamic objects, showing the advantage of our method to generate pseudo labels.

the scene but also to identify individual instances, which we show in Figure 8.7. Given that we do not use ground-truth semantics and rely on a foundation model, our method sometimes predicts wrong semantic classes, which can lead to wrong instance predictions.

In summary, our experiments show that while relying only on images and foundation models, our method is the first to perform 3D panoptic occupancy estimation, being able to reconstruct the scene and predict semantic classes and instances. Furthermore, our approach achieves state-of-the-art performance on semantic occupancy prediction among methods using only camera data for su-

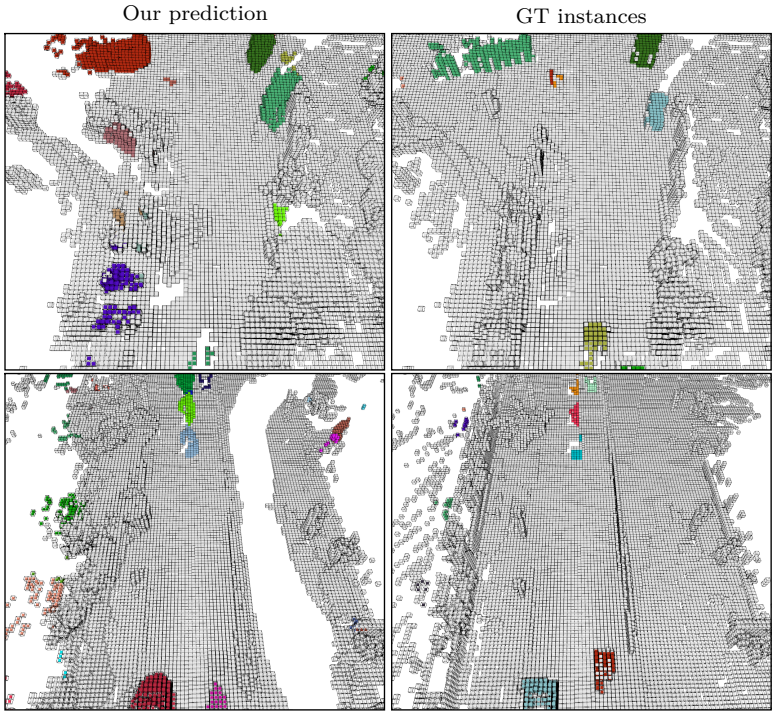


Figure 8.7: Instance prediction results. We compare the ground truth and our instance predictions. Although our approach is able to differentiate between different instances, wrong semantic class predictions can lead to wrong instance predictions.

pervision. With the inclusion of dynamic objects in our pseudo labels, our model is able to predict not only static but also moving objects.

8.3 Conclusion

In this chapter, we presented a novel approach to generate pseudo labels for 3D panoptic occupancy prediction relying only on camera data. We propose to rely only on foundation models for image segmentation and depth prediction, to generate pseudo labels and train our model using image information only. This yields, to the best of our knowledge, the first model for 3D panoptic occupancy prediction trained using image information.

Our model achieves state-of-the-art performance in 3D semantic occupancy prediction among methods that rely only on images for supervision and is able to predict dynamic objects in the scene. Our experiments suggest that we can generate pseudo labels by leveraging all the available images and adding dynamic objects using a 3D foundation model. We can use those labels to train our model for 3D panoptic occupancy prediction and estimate occupancy, semantics, and instances, also for dynamic objects in the scene.

In this second part of the thesis, we worked with RGB cameras as a single

data modality to tackle the task of semantic scene understanding. We used off-the-shelf solutions for the semantic interpretation of the images and focused on predicting the structure of the surrounding scene, including semantic and instance information. As a result, our proposed methods relying only on RGB cameras advance the state-of-the-art toward a more holistic vision-centric understanding of the surrounding scene.

This part of the thesis showcases how we can leverage camera data to train scene understanding methods and extract 3D information essential for autonomous vehicles' navigation. The resulting reconstruction performance is similar to that produced by systems that rely on 3D sensors such as LiDAR, underscoring the potential of vision-only approaches. However, the current method still faces limitations regarding the semantic recognition of the different elements and objects in the scene. This aspect is critical to achieving a more reliable understanding of the surroundings, which is key for robust planning and decision-making in autonomous vehicles.

Chapter 9

Conclusion

Safe autonomous navigation of robotic systems and self-driving vehicles in dynamic environments requires them to understand their surroundings both geometrically and semantically to make decisions and plan ahead in a safe way. Perception is a critical component of autonomy, as it serves as the foundation for planning and control, and the safety and effectiveness of decision-making directly depend on its quality. To achieve such an interpretation of the scene, autonomous vehicles are usually equipped with different complementary sensors, such as RGB cameras and LiDAR scanners, with their inherent strengths and challenges. On the one hand, RGB cameras provide high-resolution colored images with rich texture, which makes it easier to understand details in the scene. However, they only provide 2D information, lack depth, and do not work in low-light conditions. On the other hand, LiDAR sensors provide 3D geometric information about the surrounding scene, but without color information and at a lower resolution, which makes the interpretation of the scene more challenging.

In this thesis, we investigated methods to tackle spatio-temporal semantic scene understanding developed for individual data modalities, namely LiDAR and RGB cameras. We work with each modality independently to explore the capabilities and challenges of each one of them without the complexity of sensor fusion. Fusing multiple sensor modalities leads to sensor redundancy, which enhances safety by ensuring continuous operation if one sensor fails. Furthermore, the complementary nature of the obtained information helps achieve a more robust and reliable scene understanding, improving perception in challenging conditions like low light and rain. However, this fusion increases the complexity of the perception system, the price, and the computational demands, while the fusion of potential inconsistencies between sensor data requires dedicated algorithms.

In Part I, we tackle 3D semantic scene understanding using only a LiDAR scanner. This sensor intrinsically provides 3D geometric information about the

scene, and therefore, we focus on semantic segmentation and instance tracking to provide a more complete understanding of the scene. This first part of the thesis enables end-to-end trainable LiDAR segmentation and tracking models without requiring hand-tuning of parameters or heuristics to tackle scene understanding. We advance state-of-the-art towards more scalable approaches that can learn all needed parameters from the data directly, and can include data continuously collected by autonomous systems from a variety of scenarios to enhance the performance and generability.

In Part II, we focus on 3D semantic scene understanding using only data from RGB cameras. The captured RGB images contain rich texture and color information that makes the reasoning easier. Furthermore, vision foundation models provide out-of-the-shelf segmentation. In this scenario, we focus on the reconstruction of the surrounding 3D scene by predicting the occupancy values for a voxel grid around the vehicle. We use structure-from-motion to generate 3D labels using all the available images and use them to train methods for semantic occupancy prediction, relying only on image information and without assuming the availability of a LiDAR sensor. This second part of the thesis enables the usage of image data along with foundation models to extract 3D information and tackle scene understanding, which is essential for autonomous vehicles' navigation. We improve the reconstruction of vision-only systems and close the performance gap with methods relying on 3D sensors like LiDAR. However, the presented methods face limitations when semantically segmenting the scene, which is critical to achieve a more reliable understanding of the surroundings and make decisions accordingly.

9.1 Short Summary of the Key Contributions

In this thesis, we investigate how to achieve spatio-temporal and semantic scene understanding, which is a cornerstone in enabling autonomous vehicles to safely navigate complex environments. To analyze the challenges of individual data modalities, we independently study this task relying only on LiDAR data in Part I and RGB cameras in Part II. This section summarizes the main contributions of the individual parts of this thesis.

In Part I, we focus on semantically segmenting the scene and tracking individual traffic participants using LiDAR scans, and in Part II on reconstructing the 3D geometry of the scene using only RGB cameras. For safe navigation, multiple sensors provide complementary information, and sensor redundancy ensures perception even if one sensor fails. We investigate each data modality separately to investigate the challenges of each one of them without dealing with sensor fusion.

In Chapter 4 we present our first contribution, a method to extend 3D panop-

tic segmentation networks and obtain instance IDs consistent over time via the tracking-by-detection paradigm. We first perform panoptic segmentation of each new LiDAR scan and then combine appearance and motion information to associate instances over time and obtain 4D panoptic segmentation. At the core of our method is a CNN that generates instance-wise feature vectors to encode appearance information. We train such networks using contrastive learning and provide as positive and negative samples, the same and other instances. We enforce this way the feature vectors of the same instance to be similar over time and dissimilar to the ones of other instances, which allows us to use this appearance information for instance association.

Since the proposed method relies on post-processing steps such as clustering and instance associations, and relies on hand-tuning parameters, it is not possible to train the model end-to-end and learn segmentation and association from data. As a first step in that direction, the contribution of Chapter 5 is a method to learn 3D panoptic segmentation in an end-to-end fashion. This approach does not rely on post-processing steps like clustering and can learn segmentation directly from the provided data without relying on external parameter tuning. We first reformulate the task of panoptic segmentation as the prediction of a set of non-overlapping binary masks along with their semantic classes. To obtain these masks and classes, we combine point-wise features extracted from a sparse convolutional backbone with learnable feature vectors that act as mask proposals. These mask proposals, also called queries, are randomly initialized and optimized with the rest of the model. We refine these mask proposals using a transformer decoder in which we use cross-attention and self-attention mechanisms to allow the interaction between queries and point features and between all queries. Given our formulation and the fact that the model is end-to-end trainable, it has the advantage that it can learn to segment instances of different shapes and sizes directly from the data and this makes it more scalable to potentially a larger training dataset.

In Chapter 6, we tackle again the task of 4D panoptic segmentation. The contribution of this chapter is an end-to-end trainable method that does not need post-processing steps, such as clustering and associations like in Chapter 6 and without the need of hand-tuning parameters. We take the method presented in Chapter 5 and extend it by using two sets of learnable queries: detection and tracking queries. We use the detection queries to segment all newly appearing instances and stuff classes, and use the tracking queries to segment the same instance over time, adding instance tracking. For successful tracking, we keep updating the changes in the appearance of each instance by using information from the last time we segmented it. We furthermore use this last segmentation to leverage spatial information and guide the segmentation in the next time step.

To include prior spatial information, we propose position-aware mask attention. We first compute a Gaussian-like kernel around the tracked instance and use it to modify the attention weights to lead the query to focus around the position of the instance. With these modifications, our model performs 4D panoptic segmentation in an end-to-end manner without the need for post-processing steps or parameter tuning. This allows us to jointly optimize for segmentation and association, and learn how to combine these cues directly from the data, which makes our model more scalable to new data.

In Part II we focus on 3D semantic scene understanding using only RGB cameras as sensors. Since cameras provide rich texture and color information which facilitates semantic segmentation, we use an off-the-shelf segmentation model and concentrate on reconstructing the 3D geometry of the scene. In Chapter 7 we focus on the supervision of networks for 3D semantic occupancy prediction without relying on the availability of a LiDAR sensor or manual annotations. The contribution of this chapter is a method to generate sparse occupancy pseudo labels relying solely on RGB images and a vision foundation model to obtain semantic segmentation of those images. We leverage all the available images used for training and use bundle adjustment to align them and obtain camera poses. Given the computed camera poses, we use multi-view stereo reconstruction to generate a sparse point cloud, which we project into each camera to obtain sparse scale-aware depth images. We can use these depth images to supervise depth without assuming that the system includes a LiDAR sensor. Because the generated point cloud is sparse, projecting it into the images leads to wrong depth values belonging to occluded objects. To filter out these values, we use the estimated depth values and camera parameters to compute a triangle mesh, which allows us to identify pixels belonging to occluded objects. To supervise directly with occupancy values, we build a voxel grid and perform ray casting using the filtered depth images to set each voxel as occupied or free, and obtain sparse occupancy pseudo labels. We furthermore leverage a vision foundation model to segment each RGB image and add semantics to the voxel grid to obtain sparse semantic occupancy pseudo labels. This allows us to train our model without relying on LiDAR scans or occupancy ground-truth labels.

In Chapter 8 we focus on 3D panoptic occupancy prediction relying only on RGB cameras as sensors. The contribution of this chapter is a method to generate pseudo labels using all the available RGB images and leveraging foundation models for image segmentation and depth prediction. We obtain scale-aware depth images following the work presented in Chapter 7. We use a vision foundation model to segment the RGB images and obtain semantic and instance predictions. We perform ray casting using the depth images to obtain pseudo labels depicting the geometry of the scene and use the segmented images to add not only semantic

but also instance information. Given the static scene assumption made during bundle adjustment, the generated pseudo labels do not include dynamic objects. We obtain dense depth predictions using a 3D foundation model and utilize them to detect dynamic objects in the scene, which we add to the pseudo labels. This allows us to train a model using only RGB images and predict the geometry, semantics, and instances of the scene, including the moving objects.

9.2 Potential for Future Work

In this work, we focus on spatio-temporal and semantic scene understanding using a single sensor modality at a time. In Part I, we worked with LiDAR point clouds, which already provide a 3D representation of the scene, and therefore we focused on segmentation and instance tracking. In Chapter 4, we started by adding tracking into existing 3D panoptic segmentation methods and discovered that to exploit large amounts of data, we need models that are end-to-end trainable and do not rely on hand-tuning hyperparameters and post-processing operations. This led to the method presented in Chapter 5 for 3D panoptic segmentation and in Chapter 6 for 4D panoptic segmentation. The advantage of these methods is that we can potentially extend the training dataset and learn from it how to segment scans and associate instances over time. The key challenges to bring the proposed methods in Part I to real self-driving cars are mainly the reliance on large amounts of annotated data and the generalization to different sensors.

At the core of the presented methods is the transformer architecture, which benefits from large amounts of data. However, annotating 3D LiDAR scans is a laborious and expensive task due to their 3D nature and sparsity. To address this requirement, one could first segment lots of images by following works in image-based segmentation [71] and generate large amounts of labeled 3D data by using a model-in-the-loop data engine to, at the same time, label 3D point clouds and train the model. A first step into that direction was given by Ošep et al. [112], who proposed to use SAM [71] to segment images and project the labels to the point cloud to label large amounts of data and train a transformer-based model for open-vocabulary LiDAR segmentation. Leveraging 2D foundation models to generate labels for 3D data is also investigated by Huang et al. [58], who project 2D labels generated with SAM to train a class-agnostic 3D scene segmentation model and further fine-tune it using the most confident predictions of the model. This way, they are able to boost the segmentation performance without the need for manual training labels. To enable the interactive labeling of LiDAR point clouds, one could use the work by Fradlin et al. [**?**], which proposes a new paradigm to allow segmenting multiple objects on multiple LiDAR scans simultaneously by superimposing consecutive scans and emulating clicks.

Second, different LiDAR sensors produce point clouds with different resolutions and different patterns, and there is no unified architecture to process all these kinds of point clouds. Our methods rely on the point-wise features extracted by encoder-decoder architectures with sparse convolutions to segment the scene and separate it into different instances of the same class. The choice of the feature extractor is critical for the segmentation performance and, therefore, should be investigated in depth to generate descriptive and meaningful features for each point. Moreover, since point clouds are unstructured, there is no consensus in the community about which data representation to use for point cloud processing, and which advantages and disadvantages each one offers. Therefore, a thorough study should be conducted to understand the weaknesses and strengths of each data representation and the architectures for feature extractors, and potentially propose a unified representation alongside an architecture that generates meaningful point-wise features that can be used for the segmentation task.

In Part II, we focused on the 3D reconstruction of the surrounding scene using only RGB images. We generated pseudo labels to train models for occupancy prediction, relying on only on camera data and leveraging foundation models. In Chapter 7 we perform bundle adjustment using all the available RGB images to generate sparse scale-aware depth images. We built a 3D voxel grid and used ray casting to set the occupancy values of voxels along each ray of each depth image. Using a vision foundation model, we added semantic information to the pseudo labels. This allowed us to generate sparse pseudo labels to supervise methods for semantic occupancy prediction. In Chapter 8 we added instance information to the pseudo labels by performing panoptic segmentation of the RGB images using a vision foundation model. However, given the static scene assumption made in the bundle adjustment, the pseudo labels still did not include dynamic objects. We leveraged a 3D foundation model to obtain dense depth predictions and used them to detect and add these dynamic objects to the pseudo labels. This allowed us to predict, from a set of RGB images, the geometry of the scene along with the semantic and instance information.

In the task occupancy prediction, much emphasis is made on the add-ons to enhance performance without a thorough revision of the basics of depth estimation, which is a core technique for the wide variety of methods that build on top of LSS [117] principles. First, considering multiple consecutive RGB images as input would provide the depth prediction network with valuable information about the surrounding scene, making it easier to understand at inference time, where objects in the world are, leading to better performance.

Second, we note that one of the key challenges to bringing the proposed methods to real self-driving cars is the low segmentation performance, which lies in the quality of the generated pseudo labels. While training with our generated

pseudo labels using only images and structure-from-motion achieves IoUs compared to methods supervised with segmented LiDAR scans, the segmentation errors in the images lead to poor quality in the semantic classes and instances of the occupied voxels, and therefore the mIoU and the PQ. Improving the image segmentation using a fine-tuned model or manually annotating images would drastically improve the generation of pseudo labels using structure-from-motion

Third, in our work, we add moving objects to the pseudo labels, relying on the dense depth predictions made by a 3D foundation model. However, these predictions are up-to-scale, and we rely on the bundle adjustment results to obtain scale-aware predictions. Given that the scale of the prediction is not constant for the whole image, this step introduces errors in the detection of moving instances and their addition to the pseudo labels in the right position. Therefore, we believe that improving the scale of the predicted depth, be it the real scale or at least uniform, would improve the quality of the pseudo labels and consequently the prediction of moving objects in the scene, which is crucial for safe autonomous driving navigation.

Lastly, in this thesis, we investigated scene understanding using individual data modalities to understand the individual strengths and challenges without the complexities of sensor fusion. Achieving complete scene understanding in real self-driving cars requires the complementary information that both sensor modalities contribute, and the fusion of both data modalities should be investigated. A starting point is the methods proposed in Chapter 5 and Chapter 6, which provide a framework for seamlessly fusing camera and LiDAR data similar to the multi-modality capabilities of the Perceiver [64]. With this approach, information extracted from LiDAR point clouds and RGB images can be merged to enhance performance using the asymmetric attention of the transformer decoder. This architecture provides an end-to-end framework that benefits from large amounts of data. Furthermore, this approach could be trained not only with data from a system with a sensor suite that includes synchronized LiDAR and RGB cameras but also with pseudo labels generated only from segmented RGB images and structure-from-motion.

Bibliography

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

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

[8] S. Boeder, F. Gigengack, and B. Risse. Langocc: Self-supervised open vocabulary occupancy estimation via volume rendering. *arXiv preprint*, arXiv:2407.17310, 2024.

[9] S. Boeder, F. Gigengack, and B. Risse. Occflownet: Towards self-supervised occupancy estimation via differentiable rendering and occupancy flow. *arXiv preprint*, arXiv:2402.12792, 2024.

[10] S. Boeder, F. Gigengack, and B. Risse. LangOcc: Open Vocabulary Occupancy Estimation via Volume Rendering. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2025.

[11] S. Boeder and B. Risse. OccFlowNet: Occupancy Estimation via Differentiable Rendering and Occupancy Flow. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2025.

[12] B.D. Brabandere, D. Neven, and L.V. Gool. Semantic Instance Segmentation with a Discriminative Loss Function. In *Proc. of the CVPR Workshop on Deep Learning for Robotic Vision*, 2017.

[13] J.E. Bresenham. Algorithm for computer control of a digital plotter. In *Seminal graphics: pioneering efforts that shaped the field*, pages 1–6. 1998.

[14] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

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

[16] A. Cao and R.D. Charette. MonoScene: Monocular 3D Semantic Scene Completion. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[17] H. Cao and S. Behnke. SLCF-Net Sequential LiDAR-Camera Fusion for Semantic Scene Completion Using a 3D Recurrent U-Net. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[18] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[19] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

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

[21] 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.

[22] B. Cheng, I. Misra, A.G. Schwing, A. Kirillov, and R. Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[23] B. Cheng, A.G. Schwing, and A. Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[24] R. Cheng, C. Agia, Y. Ren, X. Li, and L. Bingbing. S3cnet: A sparse semantic scene completion network for lidar point clouds. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2021.

[25] 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.

[26] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):603–619, 2002.

[27] 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.

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

[29] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2021.

[30] B. Duisterhof, Z. Lojze, W. Philippe, L. Vincent, C. Yohann, and R. Jerome. MASt3R-SfM: A Fully-Integrated Solution for Unconstrained Structure-from-Motion. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2025.

[31] D. Eigen, C. Puhrsch, and R. Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *Proc. of the Conf. Neural Information Processing Systems (NIPS)*, 2014.

[32] W. Fong, R. Mohan, H. Valeria, L. Zhou, H. Caesar, O. Beijbom, and A. Valada. Panoptic nuScenes A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[33] S. Galliani, K. Lasinger, and K. Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2015.

[34] W. Gan, F. Liu, H. Xu, N. Mo, and N. Yokoya. Gaussianocc: Fully self-supervised and efficient 3d occupancy estimation with gaussian splatting. *arXiv preprint*, arXiv:2408.11447, 2024.

[35] W. Gan, N. Mo, H. Xu, and N. Yokoya. A simple attempt for 3d occupancy estimation in autonomous driving. *arXiv preprint*, arXiv:2303.10076, 2023.

[36] P. Gao, M. Zheng, X. Wang, J. Dai, and H. Li. Fast Convergence of DETR With Spatially Modulated Co-Attention. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

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

[38] 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.

[39] 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.

[40] C. Godard, O. Mac Aodha, M. Firman, and G.J. Brostow. Digging into self-supervised monocular depth estimation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[41] 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.

[42] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Trans. on Robotics (TRO)*, 23(1):34–46, 2007.

[43] V. Guizilini, I. Vasiljevic, R. Ambrus, G. Shakhnarovich, and A. Gaidon. Full surround monodepth from multiple cameras. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):5397–5404, 2022.

[44] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 43:4338–4364, 2020.

[45] A. Gupta, P. Dollar, and R. Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5356–5364, 2019.

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

[47] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[48] 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.

[49] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint*, arXiv:1703.07737, 2017.

[50] V.H. Hiep, R. Keriven, P. Labatut, and J.P. Pons. Towards high-resolution large-scale multi-view stereo. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

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

[52] F. Hong, H. Zhou, X. Zhu, H. Li, and Z. Liu. LiDAR-based 4D Panoptic Segmentation via Dynamic Shifting Network. *arXiv preprint*, arXiv:2203.07186, 2022.

[53] 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.

[54] P. Hu, D. Held, and D. Ramanan. Learning to Optimally Segment Point Clouds. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):875–882, 2020.

[55] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[56] J. Huang and G. Huang. Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint*, arXiv:2203.17054, 2022.

[57] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint*, arXiv:2112.11790, 2021.

[58] R. Huang, S. Peng, A. Takmaz, F. Tombari, M. Pollefeys, S. Song, G. Huang, and F. Engelmann. Segment3d: Learning fine-grained class-agnostic 3d segmentation without manual labels. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 278–295, 2024.

[59] Y. Huang, W. Zheng, B. Zhang, J. Zhou, and J. Lu. SelfOcc: Self-Supervised Vision-Based 3D Occupancy Prediction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[60] Y. Huang, W. Zheng, Y. Zhang, J. Zhou, and J. Lu. Tri-Perspective View for Vision-Based 3D Semantic Occupancy Prediction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[61] J.V. Hurtado, R. Mohan, W. Burgard, and A. Valada. Mopt: Multi-object panoptic tracking. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

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

[63] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint*, arXiv:1502.03167, 2015.

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

[65] H. Jiang, F. Yan, J. Cai, J. Zheng, and J. Xiao. End-to-end 3D Point Cloud Instance Segmentation without Detection. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[66] L. Jiang, H. Zhao, S. Shi, S. Liu, C. Fu, and J. Jia. PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[67] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised Contrastive Learning. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

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

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

[70] A. Kirillov, Y. Wu, K. He, and R. Girshick. PointRend: Image Segmentation As Rendering. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[71] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A.C. Berg, W.Y. Lo, P. Dollar, and R. Girshick. Segment Anything. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023.

[72] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A.C. Berg, W.Y. Lo, et al. Segment anything. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[73] 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.

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

[75] X. Lai, Y. Chen, F. Lu, J. Liu, and J. Jia. Spherical Transformer for LiDAR-based 3D Recognition. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

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

[77] V. Leroy, Y. Cabon, and J. Revaud. Grounding Image Matching in 3D with MASt3R. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.

[78] E. Li, R. Razani, Y. Xu, and B. Liu. Cpseg: Cluster-free panoptic segmentation of 3d lidar point clouds. *arXiv preprint*, arXiv:2111.01723, 2021.

[79] E. Li, R. Razani, Y. Xu, and B. Liu. Smac-seg: Lidar panoptic segmentation via sparse multi-directional attention clustering. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[80] J. Li, X. He, Y. Wen, Y. Gao, X. Cheng, and D. Zhang. Panoptic-phnet: Towards real-time and high-precision lidar panoptic segmentation via clustering pseudo heatmap. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[81] P. Li, R. Zhao, Y. Shi, H. Zhao, J. Yuan, G. Zhou, and Y. Zhang. LODE Locally Conditioned Eikonal Implicit Scene Completion from Sparse LiDAR. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[82] S. Li, X. Chen, Y. Liu, D. Dai, C. Stachniss, and J. Gall. Multi-scale Interaction for Real-time LiDAR Data Segmentation on an Embedded Platform. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):738–745, 2022.

[83] Y. Li, Z. Ge, G. Yu, J. Yang, Z. Wang, Y. Shi, J. Sun, and Z. Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. In *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, 2023.

[84] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2022.

[85] Z. Li, Z. Yu, D. Austin, M. Fang, S. Lan, J. Kautz, and J.M. Alvarez. Fb-occ: 3d occupancy prediction based on forward-backward view transformation. *arXiv preprint*, arXiv:2307.01492, 2023.

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

[87] 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.

[88] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(10):2024–2039, 2015.

[89] J. Liu, L. Kong, J. Yang, and W. Liu. Towards better data exploitation in self-supervised monocular depth estimation. *IEEE Robotics and Automation Letters (RA-L)*, 9(1):763–770, 2023.

[90] M. Liu, Z. Qiang, H. Zhao, J. Li, Y. Du, K. Keutzer, L. DU, and S. Zhang. Prototype-Voxel Contrastive Learning for LiDAR Point Cloud Panoptic Segmentation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[91] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 38–55, 2024.

[92] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2019.

[93] Y. Lu, X. Zhu, T. Wang, and Y. Ma. Octreeocc: Efficient and multi-granularity occupancy prediction using octree queries. *arXiv preprint*, arXiv:2312.03774, 2023.

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

[95] F. Magistri, R. Marcuzzi, E. Marks, M. Sodano, J. Behley, and C. Stachniss. Efficient and Accurate Transformer-Based 3D Shape Completion and Reconstruction of Fruits for Agricultural Robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[96] E. Marks, M. Sodano, F. Magistri, L. Wiesmann, D. Desai, R. Marcuzzi, J. Behley, and C. Stachniss. High Precision Leaf Instance Segmentation in

Point Clouds Obtained Under Real Field Conditions. *IEEE Robotics and Automation Letters (RA-L)*, 8(8):4791–4798, 2023.

[97] D. Meagher. Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer. *Technical Report*, Image Processing Laboratory, Rensselaer Polytechnic Institute (IPL-TR-80-111), 1980.

[98] T. Meinhardt, A. Kirillov, L. Leal-Taixé, and C. Feichtenhofer. TrackFormer: Multi-Object Tracking With Transformers. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[99] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

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

[101] 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.

[102] I. Misra, R. Girdhar, and A. Joulin. An End-to-End Transformer Model for 3D Object Detection. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

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

[104] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. *Advances in Neural Information Processing Systems*, 26, 2013.

[105] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 1985.

[106] D. Neven, B.D. Brabandere, M. Proesmans, and L.V. Gool. Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[107] L. Nunes, R. Marcuzzi, J. Behley, and C. Stachniss. Towards Generating Realistic 3D Semantic Training Data for Autonomous Driving. *arXiv preprint*, arXiv:2503.21449, 2025.

[108] 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.

[109] 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.

[110] L. Nunes, L. Wiesmann, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. Temporal Consistent 3D LiDAR Representation Learning for Semantic Perception in Autonomous Driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[111] L. Nunes, X. Chen, R. Marcuzzi, A. Osep, L. Leal-Taixé, C. Stachniss, and J. Behley. Unsupervised Class-Agnostic Instance Segmentation of 3D LiDAR Data for Autonomous Vehicles. *IEEE Robotics and Automation Letters (RA-L)*, 7(4):8713–8720, 2022.

[112] A. Ošep, T. Meinhardt, F. Ferroni, N. Peri, D. Ramanan, and L. Leal-Taixé. Better call sal: Towards learning to segment anything in lidar. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 71–90, 2024.

[113] M. Pan, J. Liu, R. Zhang, P. Huang, X. Li, B. Wang, H. Xie, L. Liu, and S. Zhang. RenderOcc Vision-Centric 3D Occupancy Prediction with 2D Rendering Supervision. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[114] M. Pan, L. Liu, J. Liu, P. Huang, L. Wang, S. Zhang, S. Xu, Z. Lai, and K. Yang. Uniocc: Unifying vision-centric 3d occupancy prediction with geometric and semantic rendering. *arXiv preprint*, arxiv:2306.09117, 2023.

[115] S. Pan, L. Jin, H. Hu, M. Popović, and M. Bennewitz. How Many Views Are Needed to Reconstruct an Unknown Object Using NeRF? In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[116] V. Patil, W. Gansbeke, D. Dai, and L.V. Gool. Don't Forget the Past: Recurrent Depth Estimation from Monocular Video. *IEEE Robotics and Automation Letters (RA-L)*, 5(4):6813–6820, 2020.

[117] J. Philion and S. Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[118] L. Piccinelli, Y.H. Yang, C. Sakaridis, M. Segu, S. Li, L. Van Gool, and F. Yu. Unidepth: Universal monocular metric depth estimation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[119] T. Pock, L. Zebedin, and H. Bischof. *TGV-fusion*. Springer, 2011.

[120] N. Poliarnyi. Out-of-core surface reconstruction via global TGV minimization. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[121] 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.

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

[123] 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.

[124] R. Razani, R. Cheng, E. Li, E. Taghavi, Y. Ren, and L. Bingbing. GP-S3Net: Graph-Based Panoptic Sparse Semantic Segmentation Network. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[125] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[126] Reid, Donald B. An algorithm for tracking multiple targets. *IEEE Trans. on Automatic Control*, 24(6):843–854, 1979.

[127] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint*, arXiv:2401.14159, 2024.

[128] M. Schreiber, S. Hoermann, and K. Dietmayer. Long-term occupancy grid prediction using recurrent neural networks. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[129] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[130] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe. Mask3D: Mask Transformer for 3D Semantic Instance Segmentation. 2023.

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

[132] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada. EfficientLPS Efficient LiDAR Panoptic Segmentation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[133] M. Sodano, F. Magistri, E. Marks, F. Hosn, A. Zurbayev, R. Marcuzzi, M. Malladi, J. Behley, and C. Stachniss. 3D Hierarchical Panoptic Segmentation in Real Orchard Environments Across Different Sensors. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2025.

[134] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in Neural Information Processing Systems*, 29, 2016.

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

[136] 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.

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

[138] 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.

[139] X. Tian, T. Jiang, L. Yun, Y. Mao, H. Yang, Y. Wang, Y. Wang, and H. Zhao. Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

[140] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proc. of the Intl. Workshop on Vision Algorithms: Theory and Practice*, 1999.

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

[142] 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. Neural Information Processing Systems (NIPS)*, 2017.

[143] 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.

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

[145] T. Vu, K. Kim, T.M. Luu, T. Nguyen, and C.D. Yoo. SoftGroup for 3D Instance Segmentation on Point Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[146] G. Wang, H. Wang, Y. Liu, and W. Chen. Unsupervised learning of monocular depth and ego-motion using multiple masks. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[147] H. Wang, Y. Zhu, H. Adam, A. Yuille, and L.C. Chen. MaX-DeepLab: End-to-end panoptic segmentation with mask transformers. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[148] K. Wang, C. Liu, Z. Liu, F. Xiao, Y. An, X. Zhao, and S. Shen. Multi-view depth estimation by using adaptive point graph to fuse single-view depth probabilities. *IEEE Robotics and Automation Letters (RA-L)*, 9(7):6400–6407, 2024.

[149] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. DUst3R: Geometric 3D Vision Made Easy. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[150] Y. Wang, Y. Chen, X. Liao, L. Fan, and Z. Zhang. PanoOcc: Unified Occupancy Representation for Camera-based 3D Panoptic Segmentation. In

Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), 2024.

[151] Z. Wang, S. Li, M. Cao, H. Chen, and Y. Liu. Pole-like Objects Mapping and Long-Term Robot Localization in Dynamic Urban Scenarios. *arXiv preprint*, arXiv:2103.13224, 2021.

[152] M. Weber, J. Xie, M. Collins, Y. Zhu, P. Voigtlaender, H. Adam, B. Green, A. Geiger, B. Leibe, D. Cremers, A. Osep, L. Leal-Taixé, and L.C. Chen. Step: Segmenting and tracking every pixel. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[153] Y. Wei, L. Zhao, W. Zheng, Z. Zhu, Y. Rao, G. Huang, J. Lu, and J. Zhou. Surrounddepth: Entangling surrounding views for self-supervised multi-camera depth estimation. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2023.

[154] Y. Wei, L. Zhao, W. Zheng, Z. Zhu, J. Zhou, and J. Lu. Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023.

[155] K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal on Machine Learning Research (JMLR)*, 10(2), 2009.

[156] 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.

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

[158] H. Xu, S. Peng, F. Wang, H. Blum, D. Barath, A. Geiger, and M. Pollefeys. Depthsplat: Connecting Gaussian Splatting and Depth. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[159] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu. RPVNet: A Deep and Efficient Range-Point-Voxel Fusion Network for LiDAR Point Cloud Segmentation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[160] S. Xu, R. Wan, M. Ye, X. Zou, and T. Cao. Sparse cross-scale attention network for efficient lidar panoptic segmentation. In *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, 2022.

[161] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[162] X. Yang, H. Zou, X. Kong, T. Huang, Y. Liu, w. li, F. Wen, and H. Zhang. Semantic Segmentation-Assisted Scene Completion for LiDAR Point Clouds. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

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

[164] Z. Yu, C. Shu, J. Deng, K. Lu, Z. Liu, J. Yu, D. Yang, H. Li, and Y. Chen. Flashocc: Fast and memory-efficient occupancy prediction via channel-to-height plugin. *arXiv preprint*, arXiv:2311.12058, 2023.

[165] Z. Yu, C. Shu, Q. Sun, Y. Bian, X. Wei, J. Yu, Z. Liu, D. Yang, H. Li, and Y. Chen. Panoptic-flashocc: An efficient baseline to marry semantic occupancy with panoptic via instance center. *arXiv preprint*, arXiv:2406.10527, 2024.

[166] F. Zeng, B. Dong, Y. Zhang, T. Wang, X. Zhang, and Y. Wei. MOTR: End-to-End Multiple-Object Tracking with TRansformer. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2022.

[167] A. Zhang, Z.C. Lipton, M. Li, and A.J. Smola. Dive into Deep Learning. *arXiv preprint*, arXiv:2106.11342, 2021.

[168] C. Zhang, J. Yan, Y. Wei, J. Li, L. Liu, Y. Tang, Y. Duan, and J. Lu. Occnerf: Self-supervised multi-camera occupancy prediction with neural radiance fields. *arXiv preprint*, arXiv:2312.09243, 2023.

[169] F. Zhang, C. Guan, J. Fang, S. Bai, R. Yang, P. Torr, and V. Prisacariu. Instance Segmentation of LiDAR Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[170] J. Zhang, C. Herrmann, J. Hur, V. Jampani, T. Darrell, F. Cole, D. Sun, and M.H. Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2025.

[171] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh. PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[172] Y. Zhang, Z. Zhu, and D. Du. OccFormer: Dual-path Transformer for Vision-based 3D Semantic Occupancy Prediction. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023.

[173] Y. Zheng, G. Wang, J. Liu, M. Pollefeys, and H. Wang. Spherical frustum sparse convolution network for lidar point cloud semantic segmentation. *Advances in Neural Information Processing Systems*, 37, 2024.

[174] T. Zhou, M. Brown, N. Snavely, and D.G. Lowe. Unsupervised Learning of Depth and Ego-Motion From Video. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[175] X. Zhou, D. Wang, and P. Krähenbühl. Objects as Points. *arXiv preprint*, arXiv:1904.07850, 2019.

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

[177] M. Zhu, S. Han, H. Cai, S. Borse, M.G. Jadidi, and F.M. Porikli. 4D Panoptic Segmentation as Invariant and Equivariant Field Prediction. *arXiv preprint*, arXiv:2303.15651, 2023.

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

[179] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint*, arXiv:2010.04159, 2020.

[180] H. Zou, X. Yang, T. Huang, C. Zhang, Y. Liu, w. li, F. Wen, and H. Zhang. Up-To-Down Network Fusing Multi-Scale Context for 3D Semantic Scene Completion. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

List of Figures

List of Tables