THE UNIVERSITY *of* EDINBURGH

Robust Spatial Perception with 4D Radar for Mobile Autonomy

Fangqiang Ding

Doctor of Philosophy

CDT Robotics and Autonomous Systems

School of Informatics

The University of Edinburgh

2025

Abstract

Mobile autonomous systems, such as self-driving cars, field robots, and drones, have made significant progress in recent years, with the potential to transform domains like transportation, logistics, inspection, and disaster response. However, achieving reliable and generalizable mobile autonomy in complex, real-world environments remains a major challenge. A fundamental prerequisite for realizing mobile autonomy is robust spatial perception, namely the ability to interpret the surrounding environment accurately and to localise the ego-agent precisely across a variety of complex scenarios. Conventional spatial perception approaches, largely dependent on optical sensors such as cameras and LiDARs, are compromised under adverse weather conditions or in poor lighting. This vulnerability not only poses safety concerns but also impedes the long-term, large-scale deployment of mobile autonomous agents. By contrast, 4D radar, with its resilience to challenging weather, cost-effectiveness, inherent velocity measurements, and privacy-preserving characteristics, presents a promising alternative. Nevertheless, fully capitalising on its potential remains a significant challenge, owing to the sensor's unique data characteristics and the absence of a well-established research infrastructure.

Against this backdrop, the present thesis systematically investigates 4D radar-based spatial perception across three pivotal tasks: scene flow estimation, moving object detection and tracking, and 3D occupancy prediction. To address sensor-specific constraints and the intricacies inherent in each task, we propose bespoke 4D radar-based methods. First, we introduce self-supervised and cross-modal supervised learning frameworks for scene flow estimation, targeting issues of data sparsity, noise, and the lack of point-wise scene flow annotations. Our self-supervised approach, `RaFlow` adopts novel architectures and loss functions to facilitate label-free estimation, while our cross-modal technique `CMFlow`, leverages cues extracted from other sensor modalities (e.g., LiDAR and cameras) to enhance performance further. Crucially, the estimated scene flow underpins downstream tasks such as motion segmentation and ego-motion estimation. Next, we present `RaTrack`, a novel 4D radar-centric method for moving object detection and tracking. In contrast to traditional approaches reliant on object classification and 3D bounding boxes, `RaTrack` emphasises motion segmentation and clustering for improved robustness against noise and point sparsity. Lastly, for 3D occupancy prediction, we advocate the use of 4D radar tensors (4DRTs) instead of point clouds, preserving critical information ordinarily lost during point cloud generation. Our method `RadarOcc`, introduces an efficient pipeline capable of handling large,

noisy 4D radar data while mitigating interpolation errors arising from direct coordinate transformations.

Extensive experiments on multiple real-world datasets confirm that our methods achieve competitive or superior performance compared to state-of-the-art techniques. In the scene flow task, they surpass LiDAR-based approaches via radar-specific designs and outperform fully supervised models when sufficient unannotated data is incorporated during training. For moving object detection and tracking, our approach significantly outperforms existing tracking-by-detection paradigms, offering enhanced accuracy in tracking moving objects. As for 3D occupancy prediction, our pipeline achieves leading results among radar-based methods and demonstrates strong competitiveness relative to LiDAR- and camera-based alternatives. Collectively, these findings highlight the potential of 4D radar not just as a supplement but as a primary sensing modality for robust spatial perception. Nonetheless, this thesis acknowledges current limitations, both task-specific and systemic, and outlines prospective directions to further advance this emerging field.

Lay summary

Imagine a car that drives itself through a busy city, in the rain, surrounded by moving traffic. Or a robot that searches for survivors in a smoke-filled building after a fire. These are no longer science fiction. But to make them safe and reliable, these machines need to understand the world around them, just like we do. Most robots and self-driving cars use cameras and a laser-based sensor called LiDAR to "see." These sensors work well in clear conditions, but they can struggle in heavy rain, fog, or smoke. That is a serious problem because it limits where and when these machines can be trusted to operate. Would you rely on a driverless car that cannot cope with poor visibility?

My research explores a different kind of sensor, something that has been used in planes and ships for decades: radar. But not just any radar. I work with a newer version called 4D radar, which can measure not only where things are, but also how fast they are moving. It works in all weather and lighting conditions. It also protects privacy, as it does not capture detailed images of people. Radar data is harder to understand than images. It often looks like scattered points or noisy patterns. I design algorithms that help machines detect moving objects, understand how they and the vehicle itself are moving, and form a clear picture of the environment around them. These algorithms use data gathered and processed from 4D radar.

This technology could help driverless cars navigate safely in rain and fog. It could allow search-and-rescue robots to move through smoke and locate people who need help. It could also support drones delivering supplies in poor weather. Radar is more affordable, more private, and more dependable than many of the sensors used today. But until recently, we lacked the tools to make full use of it in everyday robots and vehicles. My work helps unlock this potential. We live in an unpredictable world. To build technology that works in real conditions, we need machines that can see in all conditions. My research takes us closer to that goal.

Acknowledgements

As my four-year journey in Edinburgh draws to a close, I feel immensely fortunate to have made the right decision to pursue my PhD here. What I have gained goes far beyond a degree. I have found inspiring mentorship, strong collaborations, supportive friendships, and meaningful insights into both research and life. This journey has shaped me in many ways, both professionally and personally. Most of all, it is the experience itself that I will treasure. The invaluable time I spent living and studying in Edinburgh will forever remain a cherished and unforgettable part of my life. Throughout this PhD journey, I owe my sincere thanks to many people.

I feel truly thankful to my supervisor, Chris Xiaoxuan Lu, for his continuous support and guidance throughout my PhD. He guided me in doing research, encouraged me to aim higher, and supported me in many ways, from access to equipment and collaborations to professional connections. I have learned so much from him, and his advice and encouragement meant a great deal to me. I am also grateful to my co-supervisor, Barbara Webb, for her kind support with my thesis and career development. Thanks also to my thesis examiners, Changjian Li and Sen Wang, for taking the time to assess my work and for their valuable comments.

I am grateful to my collaborators who have supported me along the way. Special thanks to Peijun Zhao, who not only worked with me on research but also offered important guidance and recommendations for my career. Many thanks to Dariu M. Gavrila for his support in various aspects of my research journey. I also thank my co-authors from the lab, including Xiangyu Wen, Lawrence Zhu, Zhen Luo, Zhijun Pan, and Ivan Zhong, for their collaboration and support during our time working together. In addition, I appreciate my remote collaborators, including Andras Palffy, Gaowen Liu, and Yiming Li, for their helpful input and kind support throughout our joint projects.

Outside of research, I am especially grateful to my friends at the University: Wei Han, Liuyuan Na, Lei Zhong, Gen Li, and Hao Yu, whose company made this journey fun and memorable. Lastly, I want to deeply thank my family for their unwavering support throughout my studies, and for their constant understanding and encouragement.

Declaration

I declare that this thesis was composed by myself, and the work contained herein is my own except where explicitly stated otherwise in the text. This work has not been submitted for any other degree or professional qualification.

(*Fangqiang Ding)*

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

1.1 Research Background

Mobile autonomy refers to the ability of mobile systems, such as autonomous vehicles, field robots and drones, to perceive their surroundings and make decisions to navigate and act in complex, real-world environments without human intervention [1–3]. These systems are poised to transform a wide range of domains, including transportation, logistics, inspection, disaster response, and domestic services, by enhancing productivity, reducing human exposure to repetitive or hazardous tasks, and promoting sustainable development. Over the past decade, the field of mobile autonomous systems has made remarkable progress, driven by substantial R&D efforts and a rapidly growing industrial ecosystem [4–6]. However, achieving reliable and generalizable mobile autonomy remains a formidable challenge, particularly under diverse deployment scenarios, unstructured environments, and under adverse weather or illumination conditions [7, 8].

A key enabler for achieving reliable and generalizable mobile autonomy is robust spatial perception, which ensures accurate understanding of the ambient environment and precise localization of the ego-agent under diverse and complex scenarios. Such capability is indispensable for the safe planning and navigation of autonomous agents in the wild [9, 10]. In recent years, spatial perception has seen rapid development thanks to the advances of deep learning-based computer vision [11–15] and the availability of large-scale data [16–20]. Key tasks of spatial perception, such as odometry [21, 22], depth estimation [23, 24], 3D object detection [25, 26] and tracking [27, 28], 3D instance/semantics segmentation [29, 30], scene flow estimation [31, 32], 3D occupancy prediction [33, 34], have undergone notable performance improvements. Despite this progress, the vast majority of spatial perception approaches rely heavily on optical

sensors, including depth or RGB cameras, and LiDAR [35, 36]. While these sensors perform well under controlled environments, they often struggle to cope with the complexity and diversity of real-world scenarios. In particular, they lack robustness against adverse weather (e.g., fog, rain, snow) and bad lighting conditions (e.g., darkness, sun glare, dimness) [37, 38]. This vulnerability poses significant risks to the reliablity of mobile autonomous agents, limiting their widespread, long-term deployment and hindering the realization of reliable and generalizable mobile autonomy.

This raises a fundamental research question: *how can we empower mobile autonomous agents with robust spatial perception to tackle the environmental challenges in real-world scenarios?* Our research draws inspiration from biological sensing mechanisms in nature [39, 40]. Many animals have evolved to rely on modalities beyond vision to sense their surroundings. For instance, bats use echolocation by emitting ultrasonic pulses and analyzing the returning echoes to navigate and hunt in complete darkness [41], while sharks, equipped with electroreceptive organs, can detect electric fields to locate prey hidden beneath sand or mud [42]. While the biological sensing is not limited to the visible signals, our embodied intelligent systems should also embrace a broader range of sensing modalities to enhance robustness against environmental challenges. Indeed, beyond the visible or near-infrared signals used by the aforementioned optical sensors, the electromagnetic (EM) spectrum contains several other bands that can be leveraged for sensing. One prominent invisible EM band is the millimeter-wave (mmWave) spectrum, and sensors that leverage the mmWave signals for sensing are commonly known as mmWave radars [43].

Unlike optical sensors, mmWave radar senses the environment by actively transmitting and receiving waveforms in the mmWave frequency range [44]. Due to their much longer wavelength compared to optical signals, mmWave radar signals can penetrate airborne particles such as raindrops and snowflakes, showing resilience against adverse weathers where optical sensors often fail. Moreover, mmWave radar can measure the Doppler velocity of objects, i.e., the relative radial velocity (RRV) (c.f. Fig. 1.1), providing an extra dimension of motion awareness for perception. This thesis focuses on single-chip mmWave radar sensors, which feature a solid-state and compact design, integrating all necessary components onto a single chip. Nowadays, off-the-shelf commercial mmWave radars, used for automotive and industrial applications, predominantly fall into this category. Unless otherwise specified, the term "mmWave radar" in the remainder of this thesis refers to single-chip solid-state mmWave radar. These sensors offer significant advantages in terms of cost-effectiveness and lightweight design, mak-

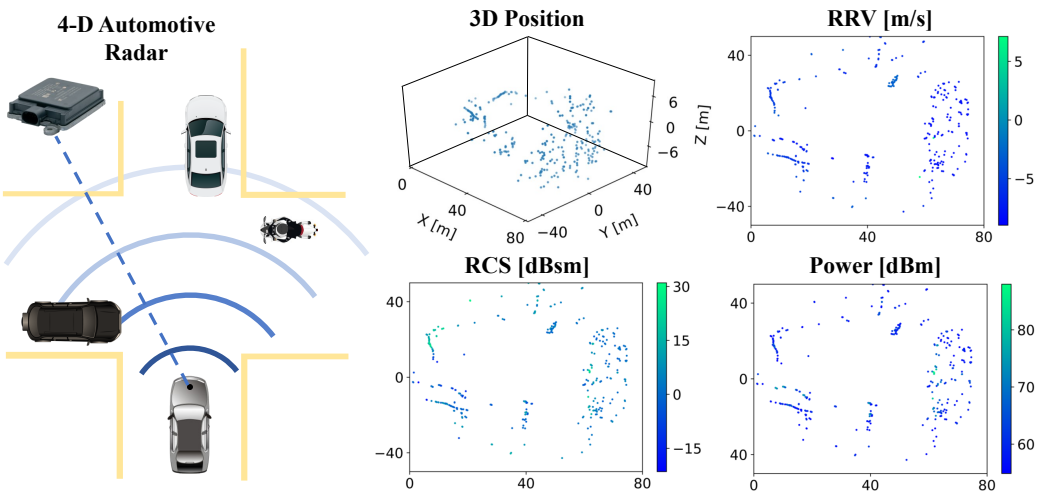


Figure 1.1: Visualization of the measurements of a 4D radar point cloud, including the 3D positional information, RRV, RCS and Power measurements. Note that the latter three features are also 3D yet we visualise them in bird's-eye view and color them by their values per point for readability.

ing them well-suited for deployment on mobile platforms with constrained payload capacity or budget. A key advance in mmWave radar technology is the evolution from traditional 3D mmWave radar to 4D mmWave radars [45,46]. Traditional 3D radars typically estimate the range, azimuth angle, and RRV of objects, but lack precise elevation information. In contrast, 4D radars introduce elevation angle estimation as the fourth dimension, significantly improving vertical resolution and enabling accurate object detection and localization in both horizontal and vertical planes. Such a LiDAR-imitating imaging capability, jointly with the typical advantages of mmWave radar, positions 4D radar as an essential component for mobile autonomy and a robust supplement, or even alternative, to LiDARs. Thus, we employ the emerging 4D radar as the primary sensor modality for robust spatial perception, formulating a sensor-driven solution to our research question.

1.2 4D Radar Signal Processing Pipeline

To help better understand the input radar modality used in this work, here we briefly introduce the signal processing pipeline of 4D FMCW radar as a preliminary.

ADC samples. To measure the surroundings, a sequence of FMCW waveforms, aka.

chirp signals, are emitted by the transmit (Tx) antennas within a short timeframe. These signals are reflected off objects and captured by the receive (Rx) antennas. The intermediate frequency (IF) signal is produced by mixing the signals from a Tx-Rx antenna pair. This mixed signal is then sampled by an Analog-to-Digital Converter (ADC) to generate discrete samples for each chirp [47]. By compiling ADC samples from all chirps and Tx-Rx antenna pairs, the FMCW radar system constructs a 3D complex data cube for each frame. This data cube is organized into three dimensions: *fast time*, *slow time*, and *channel*, which correspond to range, range rate, and angle, respectively [48].

Radar tensor. Utilizing ADC samples, Fast Fourier Transforms (FFTs) are applied across relevant dimensions to extract detailed information. The first FFT, known as range-FFT, is performed across the sample (fast time) axis to separate objects at different distances into distinct frequency responses within range bins defined by hardware specifications. Subsequently, a Doppler-FFT along the chirp (slow time) axis decodes phase variances—Doppler bins—to derive relative radial velocities, producing a range-Doppler heatmap. For configurations with multiple Rx-Tx antenna pairs, termed *virtual* antenna elements, additional FFTs (angle-FFT) are executed across the spatial dimensions of the virtual antenna array to determine Angles of Arrival (AoA) for azimuth and elevation angles. This series of transformations results in a comprehensive 4D radar tensor (4DRT), characterized by power measurements across range, Doppler velocity, azimuth, and elevation dimensions.

Radar point cloud. Beyond analyzing radar tensors, most FMCW radar sensors further refine their output to identify salient targets, which typically represent less than 1% of the data. Target detection algorithms such as CA-CFAR [49] and OS-CFAR [50] are commonly applied to the range-Doppler heatmap [48, 51] or directly on the 3D/4D radar tensors [52, 53] to isolate peak measurements. This process generates a sparse 4D radar point cloud, with each point characterized by 3D coordinates and attributes such as Doppler velocity (RRV), power intensity, or radar cross-section (RCS) (as seen in Fig. 1.1). While this step significantly reduces data volume and mitigates noise, it also eliminates a substantial amount of potentially valuable information. The RCS of a target is the hypothetical area that intercepts the radar's transmitted signals and reflects them back to the radar receiver, whose unit is usually decibels relative to a square meter (dBsm). The Power measurement equals the signal power level received from a target, whose unit is usually decibels relative to one milliwatt (dBm) [54].

1.3 Technical Challenges

To achieve reliable and generalizable mobile autonomy, spatial perception is expected to endow a mobile agent with the ability to interpret the static structure of the environment, perceive scene dynamics, understand the motion of surrounding objects, and localize itself within a global map. These capabilities are essential to ensure that the agent can make informed decisions, plan safe trajectories, and operate effectively across diverse and unstructured real-world scenarios. These high-level requirements give rise to a set of core perception tasks that serve as the foundation of autonomous behavior.

This thesis centers on three of these fundamental problems: (i) motion estimation, which involves understanding both the dynamics of the surrounding environment and the motion of the ego-agent; (ii) moving object tracking, the task of identifying and localizing moving entities across consecutive frames; and (iii) 3D occupancy prediction, which comprehensively reasons about occupied spaces and their semantics within a voxelized 3D representation. However, effectively harnessing 4D radar for these spatial perception problems is non-trivial, and research in this field remains limited, mainly because of the lack of a mature 4D radar research ecosystems and the unique characteristics of 4D radar sensors and their data. In the following, we detail the specific technical challenges associated with each individual problem.

1.3.1 Motion Estimation

As a critical spatial perception aspect for mobile autonomy, motion estimation involves accurately understanding the scene dynamics and the ego-agent's motion, enabling safe planning and navigation. A widely used representation of such motion is *scene flow*, defined as a set of point-wise displacement vectors describing the 3D motion between consecutive frames relative to the ego-agent [55]. Accurate scene flow provides holistic motion cues and serves as a cornerstone for motion perception subtasks such as motion segmentation and odometry. Recent advances in scene flow estimation have been driven by supervised [56–58] and self-supervised learning methods [59–61], predominantly on LiDAR data. However, these methods are not readily applicable to 4D radar.

Compared to LiDAR, 4D radar point clouds are significantly sparser, noisier, and lower in resolution. A single radar sweep typically contains only a few hundred points (c.f. Fig. 1.1), which undermines the robustness of local feature extraction. Moreover, radar returns are often corrupted by ghost points caused by multi-path effects and specular reflections, further complicating point association and motion estimation. The

limited angular and distance resolution of radar, constrained by hardware design, also leads to reduced spatial precision. In addition to these sensing-related challenges, there is a lack of public 4D radar datasets with point-level scene flow annotations. This limits the applicability of supervised learning approaches, which rely on high-quality labels to constrain model outputs. On the other hand, while self-supervised methods avoid manual annotation, they often fail to produce sufficiently reliable results for safety-critical scenarios. This creates a trade-off between annotation cost and model performance that need to be addressed for better 4D radar-based scene flow learning.

1.3.2 Moving Object Tracking

Robustly detecting and associating moving objects across frames in 3D space is essential for mobile autonomous systems, and serves as a prerequisite for higher-level autonomy tasks such as trajectory prediction [62–64], obstacle avoidance [65–67], and path planning [68–70]. Existing methods for 3D multi-object tracking (MOT) have made significant progress using LiDAR [71–73], RGB cameras [74–76], or their fusion [77–79]. However, the potential of 4D mmWave radar for this task remains under-explored.

Integrating 4D radars into moving object tracking presents non-trivial challenges. The prevalent approaches [71–73,78,80] often follow the *tracking-by-detection* paradigm. Such a paradigm involves first detecting objects in each frame independently and then linking these *detected object types and 3D bounding boxes* across consecutive frames to form continuous object trajectories. Key to the *tracking-by-detection* success depends on the detection accuracy. This paradigm struggles when adapted to 4D radar data, due to the inherent radar noise and point sparsity, undermining accurate type classification and bounding box regression. Specifically, the non-negligible multi-path noises in radar data complicate the correct identification of objects while the sparsity of radar point clouds makes the object property (e.g., extension and orientation) regression even more difficult. As exhibited in [81], the mAP performance [82] of the 4D radar detection method is only 38.0, $\sim$40% inferior to its LiDAR counterpart in the same scene. Such degraded detection quality compromises the effectiveness of radar-based tracking, highlighting the need for alternative approaches that are less dependent on detection outputs and more robust to radar-specific measurement artifacts.

1.3.3 3D Occupancy Prediction

As a unified voxel-based scene representation, 3D occupancy has gained increasing attention [83–87] due to its comprehensive scene depiction, capturing both geometric and semantic aspects. It offers a detailed open-set depiction of scene geometry, effectively handling out-of-vocabulary items (e.g., animals) and irregular shapes (e.g., cranes), addressing a broader range of corner cases. Previous research has predominantly utilized either LiDAR point clouds [84, 88–95], RGB images [86, 87, 96–106], or a combination of both [85] for 3D occupancy prediction. However, the potential of 4D imaging radar [45, 107] has been largely untapped in this area.

Unlike object-based perception, which focuses on the foreground entities (e.g., car, pedestrian, and trucks), 3D occupancy prediction requires to reason all occupied spaces, encompassing both foreground and background elements such as roads, barriers, and buildings. The traditional reliance on sparse radar point clouds, therefore, is not optimal for 3D occupancy prediction, as critical environmental signals are often lost during the point cloud generation process [49, 108]. For instance, the surface of highways, typically made of low-reflectivity materials such as asphalt, often yields weak signals back to the radar receiver. These limitations motivate the use of raw 4D radar tensors (4DRTs) that can preserve richer radar signal information for 3D occupancy prediction.

1.3.4 Research Goals

Given the aforementioned technical challenges associated with the three targeted spatial perception problems using 4D radar, this thesis sets out the following research goals:

- **Scene flow.** Develop learning frameworks that can learn to estimate scene flow from 4D radar data without relying on costly human annotations. These frameworks should be tailored to the sparse and noisy nature of radar point clouds, and provide reliable scene flow, supporting downstream tasks such as motion segmentation and ego-motion estimation.

- **Moving object tracking.** Design robust tracking methods that move beyond the conventional tracking-by-detection paradigm. These methods should be specifically designed for 4D radar and remain effective under point sparsity and noise, without relying on accurate 3D bounding boxes as intermediate output.

- **3D occupancy prediction.** Explore approaches for dense 3D occupancy prediction using raw 4D radar tensors (4DRTs). These approaches should align with

the characteristics of 4DRT data, address the associated challenges, and capture both foreground and background structures for reliable spatial understanding.

By addressing these goals, this thesis aims to unlock the potential of 4D radar as a robust sensing modality for spatial perception, and to promote its integration into the stack of next-generation mobile autonomous systems.

1.4 Problem Definition

This thesis addresses three spatial perception problems using 4D radar data: scene flow estimation, moving object tracking, and 3D occupancy prediction. To improve clarity and consistency, we provide a unified and standardized problem formulation for each task in this section. While we adopt consistent notation within each task, the nature of the inputs and outputs varies significantly across tasks. Therefore, we define task-specific symbols separately for each problem. The same notation is used consistently throughout the corresponding technical chapters and appendices.

Scene Flow Estimation. This task aims to solve a motion field that describes the non-rigid transformations induced both by the motion of the ego-agent and the dynamic objects in the scene. In this thesis, we formulate our task as a point cloud-based scene flow estimation problem. The inputs are two consecutive point clouds, the source one $\mathbf{P}^s = \{\mathbf{p}_i^s = \{\mathbf{c}_i^s, \mathbf{x}_i^s\}\}_{i=1}^N$ and the target one $\mathbf{P}^t = \{\mathbf{p}_i^t = \{\mathbf{c}_i^t, \mathbf{x}_i^t\}\}_{i=1}^M$, where $\mathbf{c}_i^s$, $\mathbf{c}_i^t \in \mathbb{R}^3$ are the 3D coordinates of each point, and $\mathbf{x}_i^s$, $\mathbf{x}_i^t \in \mathbb{R}^C$ are their associated raw point features. The outputs are point-wise 3D motion vectors $\mathbf{F} = \{\mathbf{f}_i \in \mathbb{R}^3\}_{i=1}^N$ that align each point in $\mathbf{P}^s$ to its corresponding position $\mathbf{c}_i' = \mathbf{c}_i^s + \mathbf{f}_i$ in the target frame. Note that $\mathbf{P}^s$ and $\mathbf{P}^t$ do not necessarily have the same number of points and there is no strict point-to-point correspondence between them under real conditions. Therefore, the corresponding location $\mathbf{c}_i'$ is not required to coincide with any points in the target point cloud $\mathbf{P}^t$, as seen in Fig. 3.2. The dimension and content of the point-wise feature vector $\mathbf{x}_i$ may vary depending on the specific 4D radar sensor and experimental settings. In Chapter 4, the raw point features $\mathbf{x}_i^s$, $\mathbf{x}_i^t \in \mathbb{R}^2$ include the relative radial velocity (RRV) and the radar cross-section (RCS) measurements [54]. RRV measurements, resolved by analyzing Doppler shift in radar frequency data, contain partial motion information of points. RCS can be seen as the proxy reflectivity of each point, which is mainly affected by the reflection property of the target and the incident angle of the beams. In Chapter 3, besides RRV and RCS measurements, the raw point features additionally includes the

Power measurements, which also represents the reflectivity information from objects.

Moving Object Tracking. In a standard 3D MOT setup, all objects of interest, such as cars and motorcycles, are tracked regardless of their motion status [71, 109, 110]. This thesis focuses solely on the moving objects. This focus stems from the premise that dynamic entities hold greater significance for tracking than static ones. Additionally, the inherent ability of radar sensors to measure velocity makes it a trivial task to distinguish between moving and stationary objects. Given this context, we consider the problem of online *moving object detection and tracking* with 4D automotive radar. The input is an ordered 4D radar point cloud sequence $\mathcal{P} = \{\mathbf{P}^t\}_{t=1}^{T}$ comprised of T frames captured by the same radar sensor mounted on a moving vehicle. A frame $\mathbf{P}^t = [\mathbf{p}_1^t; ...; \mathbf{p}_i^t; ...; \mathbf{p}_{N^t}^t]$ contains N^t radar points. Each radar point $\mathbf{p}_i^t$ is characterised by its 3D position $\mathbf{x}_i^t \in \mathbb{R}^3$ in the metric space and auxiliary velocity features $\mathbf{v}_i^t = [v_{r,i}^t, v_{c,i}^t]$, where $v_{r,i}^t$ and $v_{c,i}^t$ are the measured relative radial velocity (RRV) and its ego-motion (assumed known) compensated variant. Given each radar point cloud $\mathbf{P}^t$ from the stream, our objective is to detect multiple moving objects $\mathbf{D}^t = \{\mathbf{d}_k^t\}_{k=1}^{K^t}$ in a class-agnostic manner without the need to regress their 3D bounding boxes. These detected objects are then associated with objects tracked in the previous frame $\mathbf{O}^{t-1} = \{\mathbf{o}_m^{t-1}\}_{m=1}^{M^t}$. The result of this process is a set of updated objects $\mathbf{O}^t$ in track for the current frame.

3D Occupancy Prediction. In this thesis, we also considers the task of 3D occupancy prediction with single-frame 4DRT output from 4D imaging radar. Given a 4DRT captured in the current frame denoted as $\mathbf{V} \in \mathbb{R}^{R \times A \times E \times D}$, our task aims to predict a 3D volume $\mathbf{O} = \{o_i\}_{i=1}^{H \times W \times L}$, of which each voxel element $o_i \in \{c_0, c_1, \ldots, c_C\}$ is represented as either free (*i.e.,* c_0) or occupied with a certain semantics $c_i (i > 0)$ out of C classes. Here, R, A, E, and D denote the number of bins along the range, azimuth, elevation and Doppler axis, respectively, and each scalar of the 4DRT is the power measurement mapped to a location within the space defined by these four axes. H, W and L represent the volumetric size of the predefined region of interest (RoI) in the height, width and length dimensions.

1.5 Thesis Contributions

The contributions of this thesis correspond to four peer-reviewed publications in the domain of 4D radar-based spatial perception [119, 124–126]. Figure 1.2 provides a timeline overview of recent research developments in this area, highlighting how the four technical chapters of this thesis, relate to the broader evolution of the field.

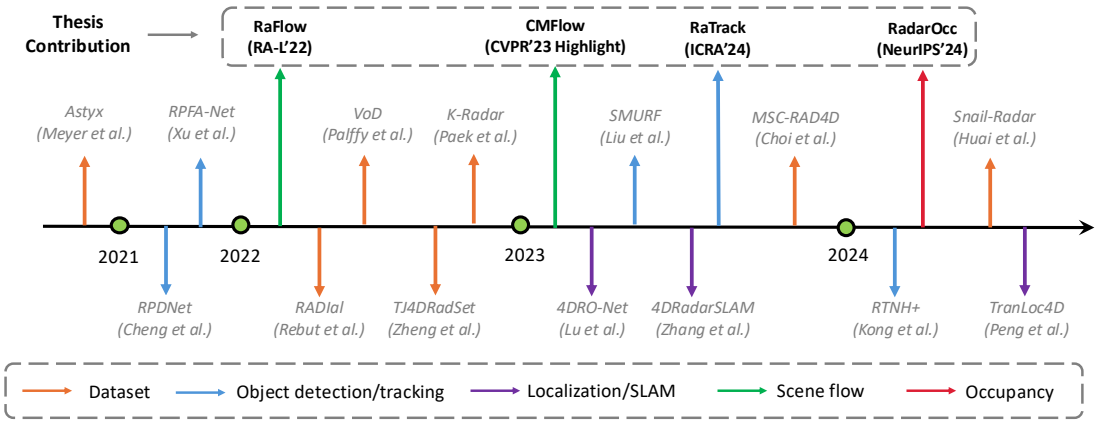


Figure 1.2: Timeline of recent 4D radar spatial perception research and the contributions of this thesis. The figure highlights key published works on datasets [52, 81, 111–115], object detection/tracking [116–120], localization and SLAM [121–123], scene flow [124, 125], and 3D occupancy prediction [126] in the past few years. The contributions of this thesis are based on four peer-reviewed publications, highlighted with boxed labels.

These contributions address three fundamental spatial perception tasks using 4D radar as the input modality: scene flow estimation, moving object tracking, and 3D occupancy prediction. A summary of each contribution is provided below.

- Self-supervised 4D Radar Scene Flow Estimation (c.f. Chapter 3): the first-of-its-kind work to investigate scene flow estimation from 4D radar data. A self-supervised learning method called `RaFlow` is introduced to estimate scene flow on 4D radar point clouds. A novel architecture and three loss functions are specifically designed to address the challenges induced by the characteristics of radar sensors, including sparsity, noise and low resolution. Without the need of annotated labels, we can collectively regularize the model to learn to estimate scene flow by exploiting the underlying supervision signals embedded in the radar measurements, such as relative radial velocity (RRV). `RaFlow` is evaluated on an in-house dataset, and achieves state-of-the-art performance on 4D radar scene flow estimation and effectively supports downstream motion segmentation.

- Cross-modal Supervision for 4D Radar Scene Flow Learning (c.f. Chapter 4): a novel cross-modal supervised approach, `CMFlow` for 4D radar scene flow learning. `CMFlow` overcomes the trade-off between annotation efforts and model performance by using complementary supervision signals retrieved from co-located heterogeneous sensors such as LiDAR, camera, and GPS/INS. To bootstrap the

cross-supervised learning, CMFlow applies a multi-task model architecture and subtly combines different types of supervision cues, formulating a multi-task learning problem. CMFlow outperforms all baseline methods on a public dataset, and can even surpass fully-supervised method when sufficient unannotated samples are used in our training. CMFlow can also improve two downstream tasks, i.e., motion segmentation and ego-motion estimation.

- Moving Object Detection and Tracking with 4D Radar (c.f. Chapter 5): a dedicated work for moving object tracking with 4D radar. Recognizing the challenges posed by radar noise and point sparsity in 4D radar data, we introduce RaTrack, an innovative solution tailored for radar-based tracking. RaTrack bypasses the typical reliance on specific object types and 3D bounding boxes, focusing on motion segmentation and clustering, enriched by a motion estimation module. Extensive experiments on the View-of-Delft dataset validate the superiority of RaTrack over existing techniques in moving object detection and tracking precision. Moreover, our results underscore the merits of the cluster-based detection method and scene flow estimation in both detection and data association phases.

- 3D Occupancy Prediction with 4D Imaging Radar (c.f. Chapter 6): introduction of the first method, RadarOcc, for 3D occupancy prediction with 4D imaging radar. RadarOcc, circumvents the limitations of sparse radar point clouds by directly processing the raw 4D radar tensor, thus preserving essential scene details that are typically lost during post-processing. RadarOcc innovatively addresses the challenges associated with the voluminous and noisy 4D radar data by employing Doppler bins descriptors, sidelobe-aware spatial sparsification, and range-wise self-attention mechanisms. To minimize the interpolation errors associated with direct coordinate transformations, we also devise a spherical-based feature encoding followed by spherical-to-Cartesian feature aggregation. The results demonstrate RadarOcc's state-of-the-art performance in radar-based 3D occupancy prediction, achieving comparable or superior accuracy to LiDAR or camera-based methods, while offering robustness under adverse conditions.

1.6 Outline

This thesis is organized into six chapters. Chapter 1 introduce the research background, technical challenges, problem definitions and thesis contributions. Chapter 2 provides a

focused review of literature relevant to the core research topics addressed in this thesis. The core technical content (Chapters 3 to 6) is based on four peer-reviewed publications, addressing three key spatial perception problems using 4D radar as the input modality: scene flow estimation, moving object tracking, and 3D occupancy prediction. The publications corresponding to each chapter are listed below:

- Chapter 3 is based on "Self-Supervised Scene Flow Estimation With 4-D Automotive Radar", F. Ding, Z. Pan, Y. Deng, J. Deng, C.X. Lu, *IEEE Robotics and Automation Letters* (2022).
- Chapter 4 is based on "Hidden Gems: 4D Radar Scene Flow Learning Using Cross-Modal Supervision", F. Ding, A. Palffy, D.M. Gavrila, C.X. Lu, *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023).
- Chapter 5 is based on "RaTrack: Moving Object Detection and Tracking with 4D Radar Point Cloud", Z. Pan, F. Ding, H. Zhong, C.X. Lu, *IEEE International Conference on Robotics and Automation* (2024).
- Chapter 6 is based on "RadarOcc: Robust 3D Occupancy Prediction with 4D Imaging Radar", F. Ding, X. Wen, Y. Zhu, Y. Li, C.X. Lu, *The Thirty-Eighth Annual Conference on Neural Information Processing Systems* (2024).

Chapter 3 proposes a self-supervised learning pipeline for scene flow estimation on 4D radar point clouds, enabling robust understanding of motion in dynamic environments. Chapter 4 extend the study of 4D radar scene flow learning and improve the performance via leveraging cross-modal supervision from co-located sensors, e.g., LiDAR, camera, GPS/INS. Chapter 5 focuses on moving object detection and tracking with 4D radar, and present a tailored solution that detects and tracks objects as class-agnostic clusters of points to address radar-based challenges. Chapter 6 introduces a novel method for dense 3D occupancy prediction from raw 4D radar tensors, circumvents the limitations of sparse radar point clouds by directly processing the raw 4D radar tensor. Finally, Chapter 7 summarizes the thesis, discusses limitations of the proposed methods, outlines potential future directions, and reflects on their broader impact.

Chapter 2

Literature Review

This chapter provides a focused review of literature relevant to the core research topics addressed in this thesis: scene flow estimation (Section 2.1), moving object tracking (Section 2.2), and 3D occupancy prediction (Section 2.3). For each task, we first discuss prior work within its broader context, emphasizing existing approaches and limitations. Although our work represents the first of its kind to tackle each of these problems using 4D radar at the time of publication, we also acknowledge the emergence of several cocurrent or subsequent studies. These recent advancements are briefly reviewed at the end of each section to provide a complete picture of the evolving research landscape. Beyond the three core tasks, 4D radar has also gained increasing attention for other spatial perception problems, such as 3D object detection [52, 53, 81, 111–113, 117–120, 127–138], odometry [114, 121, 122, 125, 139–142], and mapping [122, 141, 142]. These developments further highlight the growing potential of 4D radar in enabling robust and cost-effective spatial perception. For broader overviews of 4D radar perception, we refer readers to the dedicated survey articles [46, 143, 144].

2.1 Scene Flow Estimation

2.1.1 Image vs. Point-based Scene Flow

Scene flow was first defined in [145] as a 3D uplift of optical flow, describing the point-wise 3D displacement in a dynamic scene. Early approaches resolve pixel-wise scene flow from either RGB or RGB-D images based on prior knowledge assumptions [146–153] or by training deep networks in a supervised [154–158] or unsupervised [159–162] way. In contrast, some other methods directly infer point-wise scene flow from 3D sparse

point clouds. Among them, some methods rely on online optimization to solve scene flow [163–167]. Inspired by the success of point cloud feature learning [12, 168–170], deep learning-based methods [55–59, 61, 171–178] have been dominant for point cloud-based scene flow estimation, establishing state-of-the-art performance by leveraging large amount of data for training.

2.1.2 Supervised Point-based Scene Flow

Many point cloud-based scene flow estimation approach learn scene flow estimation in a fully-supervised manner with ground truth flow [55, 56, 171, 172, 174, 177, 179, 180]. Early efforts in this vein [55, 172] designed DNN architectures based on the point cloud processing layers [168, 170] for effective scene flow estimation. Concurrently, a multi-task pipeline [171] was proposed to predict scene flow and detect objects simultaneously with voxel-based networks. Subsequent works improved scene flow estimation accuracy by incorporating geometric constraints [177], optimal transport [174], cost volume [56], Con-HCRF [179] and adversarial learning [180]. Nevertheless, supervised DNN methods come with the demand of large-scale annotated scene flow data for training, which are costly to acquire in practice.

As a cost-effective alternative, synthetic [154] or converted [181] datasets are used for offline DNN training. However, these methods are not available for 4D radar as constructing an accurate radar synthesizer/generator itself is an unsolved topic, not to mention the common discrepancy between any simulator and real-world data. Some recent methods [58, 176] try to reduce annotation efforts by combining the ego-motion and manually annotated background segmentation labels. Although *pseudo* ground truth ego-motion can be accessed from onboard odometry sensors (GPS/INS), annotating background mask labels is still expensive and need human experts to identify foreground entities from complex scenarios.

2.1.3 Self-supervised Point-based Scene Flow

To avoid both human labour and the pitfalls of synthetic data, some methods design self-supervised learning frameworks that train a scene flow estimator without the need of labeled data [56, 57, 59–61, 164, 182]. These methods often exploit or mine a supervision signal from the input itself by either designing specific losses [56, 59, 61] or generating pseudo scene flow labels [60] to guide the model training. Yet, all these methods only consider dense LiDAR point clouds as input and their performance may degrade greatly

when applied to radars for the following reasons.

Temporal matching. Most self-supervised methods [56, 57, 59, 61, 164, 182] rely on temporal directed or bidirectional matching between two point clouds to exploit pseudo correspondence information for training. This temporal matching scheme works well for LiDAR point clouds since most points have their close neighbours in the next frame to be matched with. However, it cannot be transferred to radar point clouds directly because: a) the sparsity and noise of radar data make temporal matching between consecutive frames difficult; b) radar points have a much lower resolution which hampers accurate matching between correspondences. In Chapter 3, we propose a radar-friendly soft Chamfer loss to mitigate the negative effects of the above issues resulting from the inherent properties of the radar sensor.

Rigid ego-motion estimation. Scene flow can be disentangled into two parts induced by rigid ego-motion and moving objects. Separating these two components can not only allow us to obtain sensor ego-motion but also enable holistic refinement of flow vectors for static points. Some works [58, 171] jointly estimate ego-motion, scene flow and object rigid motion but need various labels for supervision. The method in [182] regresses ego-motion at first and then estimates the non-rigid object motion with the transformed point cloud. Another work [61] predicted point-level motion classification with the network and solved the ego-motion using the differentiable Kabsch algorithm [183]. However, these methods are either susceptible to the non-negligible amounts of outliers in radar point clouds [182], or highly rely on a large amount of artificial labels [61] that are hard to access for sparser and noiser radar data.

2.1.4 Recent Advancements in 4D Radar Scene Flow

Our contributions [125, 184] represent the initial attempts to formulate scene flow estimation directly from 4D radar data. Chapter 3 proposes a self-supervised learning pipeline for scene flow estimation on 4D radar point clouds, enabling robust understanding of motion in dynamic environments. Chapter 4 extend the study of 4D radar scene flow learning and improve the performance via leveraging cross-modal supervision from co-located sensors, e.g., LiDAR, camera, GPS/INS. Following our initial efforts, several subsequent works [185–187] have further improved 4D radar scene flow performance. DMRFlow [185] enhances matching and refinement by decoupling position and velocity features, as well as static and dynamic motions. TARS [186] integrates object detection into the pipeline and builds a traffic-level vector field to guide flow

estimation with global motion context. RadarSFEMOS [187] improves robustness through a self-supervised diffusion-based framework and Transformer-based feature encoding, jointly learning scene flow and motion segmentation.

2.2 Moving Object Tracking

Given the absence of prior work on 4D radar-based moving object tracking, we will touch on existing research in general 3D Multiple Object Tracking (MOT). As an uplift of 2D MOT [188–192] in the 3D space, 3D MOT has attracted increasing interests due to its significant application to autonomous systems. Most online 3D MOT solutions adopt a tracking-by-detection approach, focusing on 3D bounding box detection and data association. The core of data association lies in extracting comprehensive tracking cues and matching new detections with previous tracklets.

2.2.1 3D Bounding Box Detection

Tracking-by-detection approaches rely fundamentally on precise 3D object detection. With the rapid progress in this area, a variety of well-established 3D detectors [72, 193–197] have been widely integrated as the detection backbone in modern MOT pipelines. Depending on the sensor modality used, existing 3D multi-object tracking systems are typically categorized into LiDAR-based [71, 73, 109, 110, 198–204], image-based [74, 75, 205, 206], and LiDAR-image fusion-based [77–80] methods.

Unlike these pipelines that rely on explicit bounding box predictions, our approach in Chapter 5 operates solely on 4D radar point clouds and identifies object instances through point clustering, rather than bounding box estimation. This design eliminates the dependence on regression-based box detectors and is more suited to the sparsity and noise characteristics of radar data.

2.2.2 Tracking Cues

Different types of cues can be exploited from the detection results or input data for data association. To extract motion-related information for data association, one widely adopted approach is to model object dynamics using a Kalman filter with a constant velocity assumption, as introduced in AB3DMOT [71]. This approach, along with its extensions [73], has been adopted by many later studies [77, 78, 109, 201, 202] to provide motion cues for associating detections across frames. An alternative strategy involves

directly regressing object velocities through detection networks [72, 79, 203]. Additionally, some works [80, 200] extract latent temporal features using LSTM architectures to capture object dynamics over time. In addition to motion, appearance information is also widely used to enhance association robustness. These features are typically learned by neural networks from various modalities, such as RGB images [198], LiDAR point clouds [80, 203, 204], or multi-modal fusion [73, 77, 79, 200].

Unlike prior methods, our approach in Chapter 5 estimates per-point scene flow vectors, which serve as direct motion cues for object association. These flow vectors help match objects exhibiting similar motion and also contribute to the initial detection process. Beyond scene flow, we also extract point-wise geometric features from each cluster, enabling more robust and structure-aware data association.

2.2.3 Track-detection Matching

With motion or appearance cues extracted, most MOT frameworks construct an affinity matrix to evaluate the similarity between current detections and existing tracks. Traditional methods [71,77,198] compute such similarity using hand-crafted distance metrics, such as cosine or Euclidean (L_2) distance. In contrast, several approaches [80, 200] adopt neural networks to learn adaptive matching scores, enabling more flexible affinity modeling. Once similarity scores are obtained, assignment is typically performed via bipartite matching algorithms, most notably the Hungarian method [207], or through greedy heuristics. More recent strategies [110, 202] incorporate graph-based structures and Neural Message Passing [208] to capture inter-object dependencies and context for improved association accuracy.

In our method, we follow a learnable matching framework and employ an MLP to predict similarity between cluster pairs. Distinctively, we integrate the differentiable Sinkhorn algorithm [209] to solve the assignment problem in a fully end-to-end manner, facilitating gradient flow through the data association module and enabling tighter integration with downstream tasks such as motion forecasting and trajectory planning.

2.2.4 Recent Advancements in 4D Radar Tracking

Chapter 5 presents a tailored solution for 4D radar-based moving object tracking that detects and tracks objects as class-agnostic point clusters, addressing the challenges of radar sparsity and noise. Concurrently or subsequently, several studies [210–213] have explored alternative radar-specific designs for multi-object tracking. RadarTracker [211],

similar to ours, also avoids bounding boxes and focuses on instance-level motion segmentation. It employs temporal offset prediction and attention-based association, with an emphasis on appearance-guided tracking. Liu *et al.* [212] conduct a framework-level comparison of point object tracking (POT) and extended object tracking (EOT), and propose a hybrid TBD-EOT design for online 3D MOT using 4D radar.

In contrast to point cloud-based approaches, another two recent works [210, 213] operate directly on 4D radar tensors, utilizing its rich measurement information. CenterRadarNet [210] performs joint 3D detection and re-identification, enabling online tracking via embedding-based association. Bayes-4DRTrack [213] incorporates Bayesian uncertainty modeling and transformer-based motion prediction to improve robustness.

2.3 3D Occupancy Prediction

2.3.1 LiDAR- and Camera-based 3D Occupancy Prediction

Early research on 3D occupancy prediction, also referred to as semantic scene completion (SSC) [214], primarily focused on small-scale indoor environments [214–223]. The release of SemanticKITTI [224] extended SSC to large-scale outdoor driving scenarios, enabling LiDAR-based methods [88–92] to demonstrate promising results. In parallel, MonoScene [96] introduced SSC from a single monocular image, marking the beginning of vision-only solutions.

Following Tesla's public disclosure of their occupancy network for Full Self-Driving (FSD) [83], research interest in 3D occupancy prediction for autonomous vehicles has surged. While several recent approaches utilize LiDAR [84, 85, 93–95], the majority of methods adopt vision-based pipelines that lift 2D features into 3D representations [86, 87, 97–106]. Despite these advances, 4D radar remains significantly under-explored for 3D occupancy prediction, even though it offers unique advantages under adverse weather and low-visibility conditions.

2.3.2 Radar Tensors for Perception

Besides the post-processing radar point cloud, another data type of mmWave radar is the radar tensor (RT), which is the product of applying FFT along the corresponding dimensions to the raw ADC samples (*c.f.* Sec. 1.2). Unlike the sparse radar point cloud, dense RTs contain rich and complete measurements of the environment, refraining from information loss during point cloud generation (*e.g.*, CFAR [49, 108]). Consequently,

some works attempt to use 2D [127, 225–228], 3D [229–231] or 4D [52, 53, 120] RTs for object detection, yielding satisfactory performance. In Chapter 6, we develop a tailored approach to 4D radar-based 3D occupancy prediction based on 4DRTs.

2.3.3 Recent Advancements in 4D Radar Occupancy

Chapter 6 introduces a novel method for dense 3D occupancy prediction from raw 4D radar tensors. By directly operating on the full tensor, our approach overcomes the limitations of sparse point clouds and enables rich spatial understanding under adverse conditions. Following our work, 4D-ROLLS [232] proposes a weakly supervised framework that only uses LiDAR to generate pseudo-labels for training 4D radar point-based occupancy networks. It demonstrates strong generalization in adverse weather and supports real-time inference. In parallel, recent methods explore multi-modal fusion of multi-view 4D radar and cameras for improved occupancy prediction. MetaOcc [233] adopts a semi-supervised strategy with open-set segmentation for pseudo-label generation, and addresses radar sparsity via height-aware attention and adaptive modality fusion. DORACAMOM [234] jointly performs 3D detection and occupancy prediction by combining radar geometry and image semantics. It leverages voxel-based queries, temporal encoding, and cross-modal fusion to enhance omnidirectional perception.

Chapter 3

Self-supervised Scene Flow Estimation with 4D Automotive Radar

Chapter 1 introduced the research background of this thesis, highlighting the vulnerability of optical sensors under adverse conditions and introduced 4D radar as a robust alternative for robust spatial perception. Chapter 2 provides a focused review of literature relevant to the core research topics addressed in this thesis. In this chapter, we tackle the problem of scene flow estimation, aiming to explore the potential of 4D radar for this critical task in spatial perception. Scene flow enables autonomous vehicles to reason about the arbitrary motion of multiple independent objects, which is essential for long-term mobile autonomy. While LiDAR-based scene flow estimation has seen significant progress, it remains largely unexplored for 4D radar. Compared to LiDAR point clouds, radar data are sparser, noisier, and lower in resolution (c.f. Fig. 3.1). Additionally, the lack of annotated radar scene flow datasets makes supervised learning infeasible, further complicating the problem. This chapter addresses these challenges by introducing a self-supervised learning framework for scene flow estimation on 4D radar point clouds. We propose a robust model architecture along with three novel loss functions tailored to handle the sparsity, noise, and unique characteristics of radar data. The proposed self-supervised learning framework `RaFlow`, including its model design and loss functions, is detailed in Sec. 3.1. Experimental results are presented in Sec. 3.2, with a conclusion and discussion of limitations in Sec. 3.3.

Statement of Contribution. This chapter is based on the following publication: "Self-Supervised Scene Flow Estimation With 4-D Automotive Radar," Fangqiang Ding, Zhijun Pan, Yimin Deng, Jianning Deng, Chris Xianxuan Lu, *IEEE Robotics and Automation Letters*, 2022. As the first author, I designed the self-supervised learning

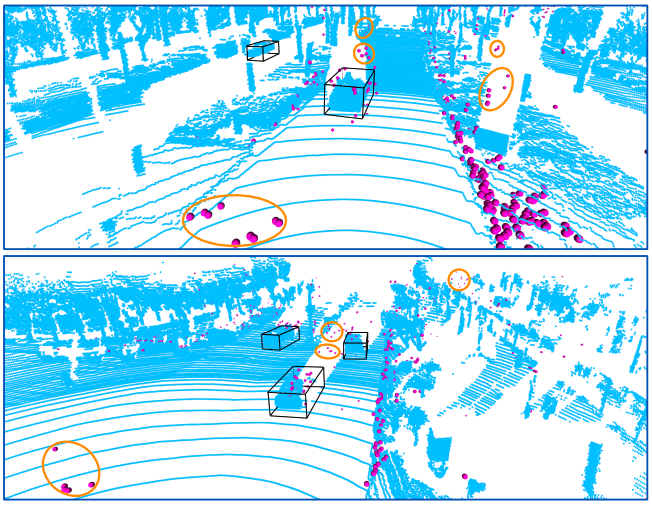


Figure 3.1: Visualized comparison between LiDAR (blue) and radar (magenta) point clouds. Black bounding boxes come from LiDAR object detection output (highlighted with bold lines). As seen in the figure, radar point clouds are much sparer than LiDAR point clouds with only a small fraction of overlapping points. Moreover, there are plenty of radar noise points, apparent clusters of which are circled in orange. Due to a much lower resolution, many points that should be inside the bounding boxes fall outside.

framework, implemented the proposed method and most baseline models, conducted the experiments, and wrote the initial draft of all sections of the paper. My co-author Zhijun Pan implemented the SLIM baseline [61]. Yimin Deng provided the in-house dataset, and I referred to the dataset preprocessing codes provided by Jianning Deng. Chris Xiaoxuan Lu offered extensive feedback on the writing and helped refine the manuscript to enhance readability and clarity, and contributed valuable insights into the method's development, experimental design, and presentation.

3.1 Proposed Method

3.1.1 Overview

The 4D radar scene flow estimation problem and some symbols is formally defined in Sec 1.4. Fig. 3.3 illustrates the overall architecture of `RaFlow`, which is composed of a *Radar-Oriented Flow Estimation* (ROFE) module and a *Static Flow Refinement* (SFR) module. The ROFE module takes point clouds $\mathbf{P}^s$ and $\mathbf{P}^t$ as the input pair and firstly estimates a coarse scene flow $\mathbf{F}_c \in \mathbb{R}^{N\times 3} = \{\mathbf{f}_{c,i} \in \mathbb{R}^3\}_{i=1}^{N}$, where each flow vector is unconstrained. Based on the coarse scene flow, the SFR module generates a static mask

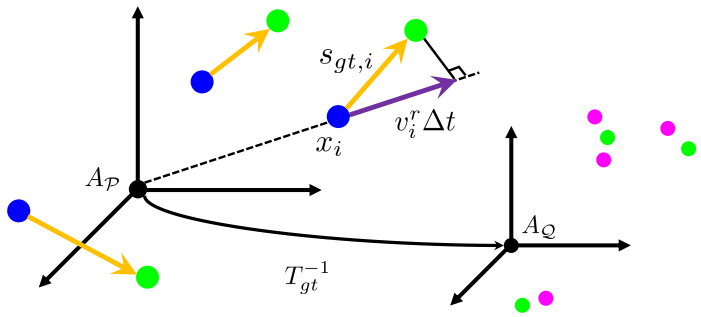


Figure 3.2: Illustration of radar scene flow and 4D radar RRV measurement. Points from point cloud $\mathcal{P}$ and $\mathcal{Q}$ are colored as blue and magenta, respectively. Green is used to denote points warped by the truth flow vectors. In above figure, we assume that all points keep constant velocity during Δt. $A_{\mathcal{P}}$ and $A_{\mathcal{Q}}$ indicate two local sensor coordinates where point cloud $\mathcal{P}$ and $\mathcal{Q}$ are captured respectively. T_{gt}^{-1} is the true ego-motion. Note that the point clouds in these consecutive frames are non-bijective, as the point gap shown in the $A_{\mathcal{Q}}$ coordinate. (Best viewed in color)

$\mathbf{M}_s$ using Algorithm 1 and then estimates a global rigid transformation $\mathbf{T}_r \in SE(3)$ with the differentiable Kabsch algorithm [183]. The rigid scene flow estimation $\mathbf{F}_r \in \mathbb{R}^{N_1 \times 3} = \{\mathbf{f}_{r,i} \in \mathbb{R}^3\}_{i=1}^{N_1}$ can be reparameterized through $\mathbf{f}_{r,i} = (\mathbf{T}_r - I_4)\tilde{\mathbf{c}}_i^s$, where $\tilde{\mathbf{c}}_i^s$ denotes $\mathbf{c}_i^s$ in the homogeneous coordinate. Finally, we refine the coarse flow vectors of all static points with the rigid scene flow $\mathbf{F}_r$ to obtain the final scene flow $\mathbf{F}$. The entire architecture is end-to-end trainable. With this architecture, we can not only predict a refined scene flow $\mathbf{F}$ but also address the motion segmentation and rigid ego-motion estimation problems. The static mask $\mathbf{M}_s$ can be used to segment moving points, while the rigid transformation $\mathbf{T}_r$ describes the ego-motion of radar.

3.1.2 Radar-oriented Flow Estimation (ROFE) Module

As shown in Fig. 3.3, the ROFE module consists of three components: multi-scale encoder, cost volume layer and flow estimator, which are explained below. The detailed layer parameters of the ROFE module can be seen in Table 3.1.

Multi-scale Encoder. Extracting robust local features from radar points clouds is hindered by two major factors: a) the severe sparse nature of radar, b) uneven point density. While enlarging the receptive field can mitigate the sparse issue, it is still hard to tackle uneven point density with single-scale feature extraction. Inspired by the discussion in [168], we adopt a multi-scale grouping scheme to encode features on radar point clouds. Given input point set $\mathbf{P}^s$, we use multiple *set conv* layers [55] to group

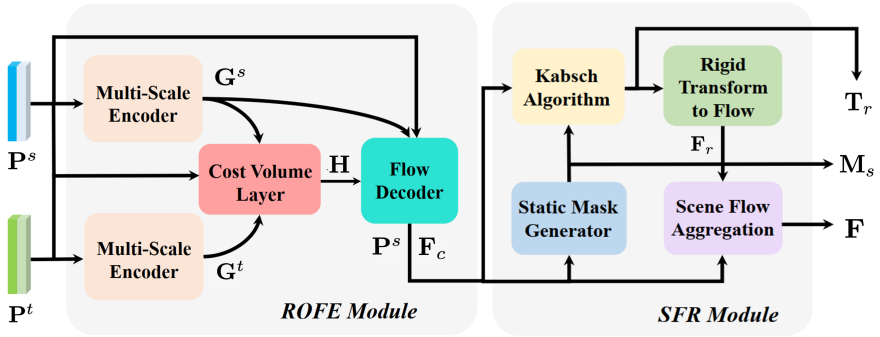


Figure 3.3: Overview of our radar scene flow estimation pipeline. Two point clouds $\mathbf{P}^s$ and $\mathbf{P}^t$ are used as the input to `RaFlow`. The output includes the aggregated final scene flow $\mathbf{F}$, the static mask $\mathbf{M}_s$ and the rigid transformation $\mathbf{T}_r$. Note that the two multi-scale encoders share the same weights. The entire architecture is end-to-end trainable.

multi-scale local features with different radius neighbourhoods specified by a radius set R. To include global knowledge, we also use a channel-wise max-pooling operation to aggregate global features following [12] and concatenate it to per-point local features. As a result, we can obtain local-global features $\mathbf{G}^s \in \mathbb{R}^{N \times (2\sum_{k=1}^{N_{sc}} C_k)}$ for point cloud $\mathbf{P}^s$ and $\mathbf{G}^t \in \mathbb{R}^{N_2 \times (2\sum_{k=1}^{M_{sc}} C_k)}$ for $\mathbf{P}^t$ similarly.

Cost Volume Layer. To robustly and stably correlate features across two frames, we adopt the *Cost Volume* layer proposed by [56]. This cost volume layer aggregates costs in a patch-to-patch manner, which is particularly useful to mitigate the low-resolution issue of radar point clouds and the resultant non-bijective mapping across frames. After passing the point clouds and local-global features through cost volume layer, we can attain correlated features $\mathbf{H} \in \mathbb{R}^{N \times C_{cor}}$.

Flow Decoder. To decode the flow estimation from features, we firstly form flow embedding by concatenating correlated, local-global and input features of point cloud $\mathbf{P}^s$. With flow embedding and point cloud $\mathbf{P}^s$, we use another multi-scale encoder to group embedding features to include spatial smoothness for the final output. We denote these features as $\mathbf{U} \in \mathbb{R}^{N \times (2\sum_{k=1}^{N_{sc}} C'_k)}$ from multiple *set conv* layers. Lastly, $\mathbf{U}$ is fed into the final four-layer MLP whose output is the coarse scene flow $\mathbf{F}_c$.

3.1.3 RRV Measurement and Scene Flow

One important property of 4D radars is that their RRV measurements from Doppler effects are intuitive self-supervision signals for scene flow estimation. RRV measurements describe the moving speed of ambient objects relative to the observer in the radial direc-

Component	Layer type	Radius	Samples	MLP width
multi-scale encoder	set conv	2.0	4	[32, 32, 64]
	set conv	4.0	8	[32, 32, 64]
	set conv	8.0	16	[32, 32, 64]
	set conv	16.0	32	[32, 32, 64]
cost volume layer	-	-	8	[512, 512, 512]
flow decoder	set conv	2.0	4	[512, 256, 64]
	set conv	4.0	8	[512, 256, 64]
	set conv	8.0	16	[512, 256, 64]
	set conv	16.0	32	[512, 256, 64]
	output layer	-	-	[256, 128, 64, 3]

Table 3.1: Architecture specifications of our ROFE module.

tion. We denote the RRV measurements of point cloud $\mathbf{P}^s$ as $\mathcal{V}^s \in \mathbb{R}^N = \{v_i^s \in \mathbb{R}\}_{i=1}^N$, where v_i^s is positive when point $\mathbf{c}_i^s$ is moving away from the observation point. Generally, $\mathcal{V}^s$ can be used as a component of the input features to include point-level motion cues. However, we argue that RRV measurement has the potential to play more roles in `RaFlow`. Assuming the velocity of point $\mathbf{c}_i^s$ keeps constant during time interval Δt between two scans, we can reach the following equation:

$$v_i^s \Delta t = \mathbf{f}_{gt,i}^\top \frac{\mathbf{c}_i^s}{||\mathbf{c}_i^s||} \tag{3.1}$$

where $\mathbf{f}_{gt,i}$ is the true flow vector of point $\mathbf{c}_i^s$. Eq. 3.1 means that the projection of flow vector on the radial direction equals the measured RRV times Δt, as explained in Fig. 3.2. The constant velocity assumption is rational because the time interval between two radar scans is usually very short (e.g., 100ms) so that the average velocity of points can be seen as an approximation of the instantaneous velocity in most cases. As we will see in the rest of this chapter, RRV is crucial for our self-supervised learning framework even without the availability of the truth scene flow. We propose a static mask generation module in Section 3.1.4 and formulate a radial replacement loss function in Section 3.1.5, which are both inspired by Eq. 3.1.

3.1.4 Static Flow Refinement (SFR) Module

Due to the intrinsic sparsity and non-negligible noise of radar data, the scene flow $\mathbf{F}_c$ estimated from the ROFE module is only coarse-grained and needs to be refined further. Since the motion of stationary scene points is caused by the radar ego-motion,

Algorithm 1 Static Mask Generation

Input: Point cloud $\mathbf{P}^s$, coarse scene flow $\mathbf{F}_c$, RRV measurements $\mathcal{V}^s = \{v_i^s\}_{i=1}^N$ of $\mathbf{P}^s$, time interval Δt between $\mathbf{P}^s$ and $\mathbf{P}^t$, preset threshold ζ

Output: Static mask $\mathbf{M}_s = \{m_{s,i} \in \{0,1\}\}_{i=1}^N$

Get the warped point cloud $\mathbf{P'} = \{\mathbf{c}_i' = \mathbf{c}_i^s + \mathbf{f}_{c,i}\}_{i=1}^N$ using $\mathbf{F}_c$ and $\mathbf{P}^s$
Form correspondences $\{\mathbf{c}_i^s, \mathbf{c}_i'\}_{i=1}^N$ between $\mathbf{P}^s$ and $\mathbf{P'}$
Feed $\{\mathbf{c}_i^s, \mathbf{c}_i'\}_{i=1}^N$ into Kabsch algorithm and get $\mathbf{T}_{cr}$
Compute $\mathbf{F}_{cr} \in \mathbb{R}^{N\times 3} = \{\mathbf{f}_{cr,i} \in \mathbb{R}^3\}_{i=1}^N$ with $\mathbf{T}_{cr}$ by $\mathbf{f}_{cr,i} \leftarrow (\mathbf{T}_{cr} - I_4)\tilde{\mathbf{c}}_i^s$
for $i = 1$ to N **do**
    Obtain $\mathbf{f}_{cr,i}$'s radial component $\mathbf{f}_{cr,i}^r \leftarrow \mathbf{f}_{cr,i}^\top \frac{\mathbf{c}_i^s}{\|\mathbf{c}_i^s\|}$
    Compute the radial residual $r_i \leftarrow \mathbf{f}_{cr,i}^r - v_i^s \Delta t$
    Compute the relative residual $e_i \leftarrow \left|\frac{r_i}{v_i^s \Delta t}\right|$
    if $e_i \leq \zeta$ **then**
        $m_{s,i} \leftarrow 1$
    else
        $m_{s,i} \leftarrow 0$
    end if
end for

one plausible refinement is to regularize those flow vectors of all static (background) points via a rigid transformation matrix respective to radar ego-motion. Unfortunately, *accurate* ego-motion is often in absence in the real world or costly to acquire in practice. To circumvent this ego-motion absence, we first propose a static mask generator based on the RRV measurements available in a radar scan and then leverage the Kabsch algorithm [183] to obtain the rigid transformation of the identified static points. Particularly, similar to other scene flow works in this vein (e.g., [58,61]), we adopt a specialized form of the Kabsch algorithm, where the centered point coordinates of two sets of paired points are firstly computed by subtracting the centroid coordinates. Then, the optimal rotation matrix is solved through a singular value decomposition. The translation vector is finally restored by comparing two centroid coordinates after using the rotation matrix for compensation.

Concrete details of our proposed static mask generation algorithm can be seen in Algorithm 1. Based on our observation that most points in the scene are stationary, we intuitively assume that all points are stationary and then utilize the Kabsch algorithm [183] to get an intermediate transformation $\mathbf{T}_{cr} \in SE(3)$. Obviously, $\mathbf{T}_{cr}$ is

only a coarse transformation as not all points in the scene are stationary. With $\mathbf{T}_{cr}$, an intermediate rigid scene flow $\mathbf{F}_{cr}$ can be computed accordingly. With Eq. 3.1 in place, we judge if points are static or moving by comparing their relative residual e_i with a preset threshold ζ. Here, e_i is defined as the difference between the radial displacement induced by the intermediate rigid flow vector $\mathbf{f}_{cr,i}$ and RRV measurement v_i^s.

After applying the static mask $\mathbf{M}_s$ on point cloud $\mathbf{P}^s$ and the warped one $\mathbf{P}'$, we feed only static correspondences into the Kabsch algorithm again. As exhibited in Fig. 3.3, a more reliable rigid-motion transformation $\mathbf{T}_r$ can be thus obtained. Then, we derive the rigid scene flow $\mathbf{F}_r$ from $\mathbf{T}_r$, which can be shown as the *Rigid Transform to Flow* block in Fig. 3.3. Lastly, we aggregate the final scene flow $\mathbf{F}$ by:

$$\mathbf{f}_i = \begin{cases} \mathbf{f}_{r,i} & \text{if } m_{s,i} = 1 \\ \mathbf{f}_{c,i} & \text{if } m_{s,i} = 0 \end{cases} \tag{3.2}$$

3.1.5 Loss Function

We leverage three types of self-supervision signals for model training: radial displacement loss $\mathcal{L}_{rd}$, soft Chamfer loss $\mathcal{L}_{sc}$ and spatial smoothness loss $\mathcal{L}_{ss}$. The overall loss function can be written as:

$$\mathcal{L} = \mathcal{L}_{rd} + \mathcal{L}_{sc} + \mathcal{L}_{ss} \tag{3.3}$$

These losses jointly regulate the network learning in terms of RRV, temporal, and spatial coherence respectively.

Radial Displacement Loss. As discussed in Section 3.1.3, the product of RRV measurement and time interval Δt can be seen as an approximation of the radial projection of the truth flow vector. This insight is crucial to our self-supervised learning framework because we can use the existing radar inputs to constrain the radial component of flow vectors. Formally, we formulate a radial displacement loss based on Eq. 3.1:

$$\mathcal{L}_{rd} = \sum_{\mathbf{c}_i^s \in \mathbf{P}^s} |\mathbf{f}_i^\top \frac{\mathbf{c}_i^s}{||\mathbf{c}_i^s||} - v_i^s \Delta t| \tag{3.4}$$

Despite some inevitable measurement errors, we empirically found that RRV renders strong supervision signals for training scene flow estimators, as shown in Section 3.2.4.

Soft Chamfer Loss. Introduced by [56], Chamfer loss is an effective constraint for self-supervised learning of scene flow estimation. By employing the mutual nearest neighbours as pseudo correspondences, it enforces the scene flow to pull the warped source point cloud $\mathbf{P}' = \{\{\mathbf{c}_i', \mathbf{x}_i'\} \in \mathbb{R}^C\}_{i=1}^N$ obtained by: $\mathbf{c}_i' = \mathbf{c}_i^s + \mathbf{f}_i$ as close as possible to the target point cloud $\mathbf{P}^t$.

The challenge in our context, however, is that radar point clouds are much sparser and noisier, resulting in quite some points having no real correspondences or even close neighbors in the opposite point cloud. As a result, mapping all points to their nearest neighbours by the vanilla Chamfer loss will incur erroneous scene flow estimation. We therefore propose to formulate the Chamfer matching constraints in a probabilistic manner by taking into account the Euclidean distance between a point and its neighbor points in the opposite point cloud. Motivated by [169], we further use a kernel density estimation (KDE) method to approximate per-point Gaussian density factor $\nu(\mathbf{c}'_i)$ as follows:

$$\nu(\mathbf{c}'_i) = \frac{1}{M} \sum_{\mathbf{c}^t_j \in \mathbf{P}^t} \mathcal{N}(\mathbf{c}^t_j; \mathbf{c}'_i, 1) \tag{3.5}$$

Before loss computation, we precompute the density factors for $\mathbf{P}'$ and $\mathbf{P}^t$ as prior knowledge about the chance of attaining a successful matching of each point. To mitigate the problem caused by unsatisfying point matching, we select points whose density factor is lower than threshold δ and discard these points as outliers when computing losses. Our soft Chamfer loss can be formulated as:

$$\begin{aligned} \mathcal{L}_{sc} = & \sum_{\mathbf{c}'_i \in \mathbf{P}'} \mathbb{I}(\nu(\mathbf{c}'_i) > \delta)[\min_{\mathbf{c}^t_j \in \mathbf{P}^t} ||\mathbf{c}'_i - \mathbf{c}^t_j||_2^2 - \epsilon]_+ \\ & + \sum_{\mathbf{c}^t_i \in \mathbf{P}^t} \mathbb{I}(\nu(\mathbf{c}^t_i) > \delta)[\min_{\mathbf{c}'_j \in \mathbf{P}'} ||\mathbf{c}^t_i - \mathbf{c}'_j||_2^2 - \epsilon]_+ \end{aligned} \tag{3.6}$$

where $\mathbb{I}(\cdot)$ is the indicator function that returns one when the condition is satisfied, and zero otherwise. Due to the low resolution of radar sensor, a perfect match between two consecutive clouds is nearly impossible. It is reasonable to let the Chamfer matching tolerate small matching discrepancy. Such toleration is implemented via the $[\cdot]_+ = \max(0, \cdot)$ operator so that matching errors lower than ϵ are ignored.

Spatial Smoothness Loss. It has been found that estimating unconstrained point-wise flow vectors can easily lead to ill-posed results [58]. For example, two points from the same vehicle might move in different directions or magnitudes. To avoid unreasonable results, it is non-trivial to impose spatial coherence on the estimated motion fields. In [56, 57], a regularization term is added to constrain the predictions when training networks. They formulate a loss function by enforcing the scene flow of $\mathbf{c}^s_i$ close to that of points in its neighbour set $\mathcal{O}(\mathbf{c}^s_i)$. However, this loss is not suitable to be directly applied to radar point clouds as it implicitly assumes that all points in the neighbour set are close enough to the target point and thus can be considered as measurements from the same rigid body. This assumption does not necessarily hold when it comes to the

Sequence	*Seq0*	*Seq1*	*Seq2*	*Seq3*	*Seq4*	*Seq5*	*Seq6*
Period (min)	12.7	10.0	10.0	10.0	10.0	10.0	10.0
Distance (km)	9.5	6.3	5.6	5.7	5.4	5.4	5.2
Avg. Speed (km/h)	44.9	37.8	33.6	34.2	32.4	32.4	31.2
Usage	Test	Train	Train	Train	Train	Train	Val

Table 3.2: Details of collected sequences for our experiments.

radar point clouds due to the point sparsity and low resolution as shown in Fig. 4.1.

To address this challenge, we propose to weigh each neighbour point according to its Euclidean distance to the target point. Intuitively, farther neighbour points are less likely to correlate with the target point and less reliable when enforcing smoothness constraints. To smoothly measure the correlation between point $\mathbf{c}_i^s$ and its neighbour $\mathbf{c}_j^s \in \mathcal{O}(\mathbf{c}_i^s)$, we use the same RBF kernel as in [164] to formulate the weight:

$$k(\mathbf{c}_i^s, \mathbf{c}_j^s) = \exp(\frac{-||\mathbf{c}_i^s - \mathbf{c}_j^s||_2^2}{\alpha}) \tag{3.7}$$

where α controls the impact of the distance on the weight. All weight values are normalized together using softmax function. With the assigned weight for each pair of point and neighbour, our spatial correlation loss is formulated as:

$$\mathcal{L}_{ss} = \sum_{\mathbf{c}_i^s \in \mathbf{P}^s} \sum_{\mathbf{c}_j^s \in \mathcal{O}(\mathbf{c}_i^s)} k(\mathbf{c}_i^s, \mathbf{c}_j^s) ||\mathbf{f}_i - \mathbf{f}_j||_2^2 \tag{3.8}$$

By placing local constraints weighted by the metric distance in Eq. 3.8, the spatial coherence of predicted scene flow can be enforced more appropriately for sparser radar point clouds.

3.2 Evaluation

3.2.1 Experiment Settings

Data Collection. An in-house multi-modal dataset was collected to facilitate our evaluation. The collection vehicle was equipped with a set of sensors including a Full-HD RGB camera, an Asensing INS570D navigation system, an RS-Ruby RoboSense (128-beam) LiDAR system and a HASCO LRR30 4D mmWave automotive radar. In particular, the 4D automotive radar has a maximum range of 75m with an azimuth/elevation FOV angle of 120°/20°. The range resolution of this 4D radar is $0.2\mathrm{m}$ while the angular resolution is $\{1.6°, 1°\}$ for azimuth/elevation measurement, respectively. We manually drove the vehicle for over 70 minutes on multiple roads and 7 sequences of data were

Parameter	Meaning	Value
N_{sc}	The number of feature extraction scales	4
C_k, C'_k	The number of local feature channels	64
C_{cor}	The number of correlated feature channels	512
R	The radius set for encoding	$\{2, 2^2, 2^3, 2^4\}\mathrm{m}$
ζ	The threshold to classify points	0.15
δ	The threshold to discard outliers	0.005
ϵ	The threshold for error toleration	$0.1\mathrm{m}$
α	The factor to control smoothness weights	0.5
N_{nb}	The number of neighbours for smoothness	8

Table 3.3: Values of all hyperparameters for reproduction.

collected with a total distance of 43.6km. We synchronize radar point clouds to 10Hz LiDAR point clouds and use pose data for motion compensation. For experiments, we select one sequence for test, another one for validation and the rest five for training. Details about these sequences are shown in Table 3.2.

Labels. We use unlabelled samples from the training set for self-supervised learning of DNNs and from the validation set for model selection. As the test set is in a moderate amount of data, we manually label the test radar frames by referencing a 3D object detection module from the co-located RoboSense LiDAR and the accurate ego-vehicle pose information provided by the RTK-GPS and IMU sensors. Specifically, the scene flow labels of these samples can be annotated using converted vehicle pose for static points and by tracking bounding boxes (after manual inspection and correction) of dynamic objects for moving points.

Baselines. As there is no prior work studying the sparse radar scene flow, our competing approaches are drawn from the general point-based scene flow estimation methods, including *five* state-of-the-art self-supervised learning methods: *Just go with the Flow* (JGWTF) [59], PointPWC-Net [56], GraphL-L (the learning version) [164], FlowStep3D [57], SLIM [61], and *two* non-learning-based methods: Iterative Closest Point (ICP) [235] and GraphL-N (the non-learning version) [164].

Evaluation Metrics. Prior point-based scene flow estimation works commonly use end point error (EPE) $\mathcal{E}_i = ||\mathbf{f}_i - \mathbf{f}_{gt,i}||_2$ as the major metric for evaluation. Many of them [55, 56, 59, 164] achieve a mean EPE lower than $0.2\mathrm{m}$ with dense point clouds (e.g., LiDARs) as input, as shown in their experiments. However, directly applying EPE to radar point clouds lacks sensor-specific consideration and may incur short-

sighted conclusion because most 4D radars currently have a much lower-resolution than LiDARs. For example, our used 4D radar (i.e., HASCO LRR30 4D mmWave automotive radar) has a range resolution of $0.2\mathrm{m}$, making a sub-resolution EPE lower than $0.2\mathrm{m}$ nearly impossible. For this reason, we also introduce a metric called Resolution-normalized EPE (RNE) in this chapter to make our evaluation on par with the ones in LiDAR scene flow estimation. To compensate the resolution gap, we divide the computed EPE by $\Delta x_i^r/\Delta x_i^l$, which is the ratio between our radar resolution Δx_i^r and a common Velodyne LiDAR resolution Δx_i^l used by previous LiDAR scene flow works [56, 57, 59, 61, 164]. As both LiDAR [82] and radar point clouds are generated in the spherical coordinate, each point should have a unique resolution Δx_i in the Cartesian coordinate. To implement fine-grained RNE, given the fixed sensor spherical resolution values $\{\Delta r, \Delta\theta, \Delta\phi\}$, we first approximate per-point Cartesian resolution values $\{\Delta X_i, \Delta Y_i, \Delta Z_i\}$ by

$$\Delta c_i = \sum_{h\in\{r,\theta,\phi\}} \left| \frac{\partial c(r,\theta,\phi)}{\partial h}|_{(r,\theta,\phi)=(r_i,\theta_i,\phi_i)} \right| \Delta h \tag{3.9}$$

where $c \in \{X, Y, Z\}$ denotes the Cartesian coordinate values. We then obtain the point-level resolution Δx_i as:

$$\Delta x_i = \sqrt{(\Delta X_i)^2 + (\Delta Y_i)^2 + (\Delta Z_i)^2} \tag{3.10}$$

We first compute per-point ratio between Δx_i^r and Δx_i^l and then obtain the RNE value by dividing EPE by the ratio.

Following the accuracy score metrics in [55,57,61], we also define two extra metrics based on RNE by calculating the percentage of points that satisfy certain requirements: a) strict accuracy score (SAS): the ratio of points with RNE $\leq 0.1\mathrm{m}$ or relative error $\leq 10\%$; b) relaxed accuracy score (RAS): the ratio of points with RNE $\leq 0.2\mathrm{m}$ or the related error $\leq 20\%$. Following [61], we also report results for static and moving points separately and calculate their average to avoid the imbalance of these two classes. For notation, we use 50-50 RNE as the average of between Stat. RNE and Mov. RNE, while Avg. RNE denotes the mean RNE for all test points.

Training Details. We trained our model for 50 epochs using the Adam optimizer [236]. The initial learning rate is set as 0.001 and decays by 0.9 after each epoch. Data augmentation is implemented by randomly rotating and translating each point cloud from the training set. During training, we downsample all point clouds to $N_1 = N_2 = 256$ for fast batch processing but do not downsample test samples in order to estimate

Type	Method	Avg. EPE [m]↓	Avg. RNE [m]↓	50-50 RNE [m]↓	Mov. RNE [m]↓	Stat. RNE [m]↓	SAS↑	RAS↑
Non-learning-based	ICP [235]	1.152	0.485	0.419	0.355	0.483	0.308	0.479
	GraphL-N [164]	1.229	0.518	0.339	0.150	0.529	0.177	0.206
Self-supervised learning	JGWTF [59]	1.345	0.566	0.379	0.181	0.578	0.095	0.185
	PointPWC-Net [56]	1.242	0.523	0.380	0.227	0.533	0.108	0.390
	GraphL-L [164]	1.275	0.537	0.352	0.154	0.549	0.153	0.216
	FlowStep3D [57]	2.111	0.889	0.670	0.439	0.900	0.081	0.242
	SLIM [61]	1.224	0.515	0.338	0.149	0.527	0.178	0.207
	RaFlow (w.o. SFR)	0.608	0.256	0.185	**0.110**	0.261	0.290	0.640
	RaFlow	**0.570**	**0.240**	**0.180**	0.117	**0.244**	**0.316**	**0.675**

Table 3.4: Performance of `RaFlow`, `RaFlow` (without SFR module), and baselines on the test set. ↑ means bigger values are better while ↓ means smaller values are better. Best results are shown in **bold**.

the flow vector of each point. The values of hyperparameters can be seen in Table 3.3, which are fixed for all experiments.

3.2.2 Quantitative Results

Our experiments begin with the comparison among `RaFlow` and baselines. For a fair comparison, we train networks of self-supervised baselines [56,57,59,61,164] under the same setting as ours. For the evaluation of ICP [235], we iteratively optimize its rigid ego-motion output and apply this transformation to all points to generate scene flow. We also iteratively optimize the objective function of GraphL-N [164] to obtain the final scene flow prediction. As seen in Table 3.4, `RaFlow` outperforms other self-supervised learning and non-learning-based methods on all metrics. This confirms the effectiveness of our radar-oriented architecture design and specific loss functions to cope with sparse, noisy and low-resolution radar point clouds.

It is also worth noting the importance of the SFR module that refines all static points through a predicted rigid transformation. As we can see, while `RaFlow` without SFR can still achieve comparable results, the Stat. RNE increases by 6.5% and the SAS decreases by 9.0% due to the absence of static flow regularization in the process. We also observe that the Mov. RNE decreases by $0.7\mathrm{cm}$ without the SFR module. This drop is reasonable since the SFR module cannot be perfect and it inevitably mis-classifies a few moving points to static, whose flow vectors are further replaced by the rigid flow derived from the estimated rigid ego-motion. In our data collection, moving points (e.g., vehicles and motorcycles etc.) are relatively fewer than the static points (e.g., trees and road railings etc.), which can be seen in Fig. 3.4. It is therefore a trade-off to achieve the

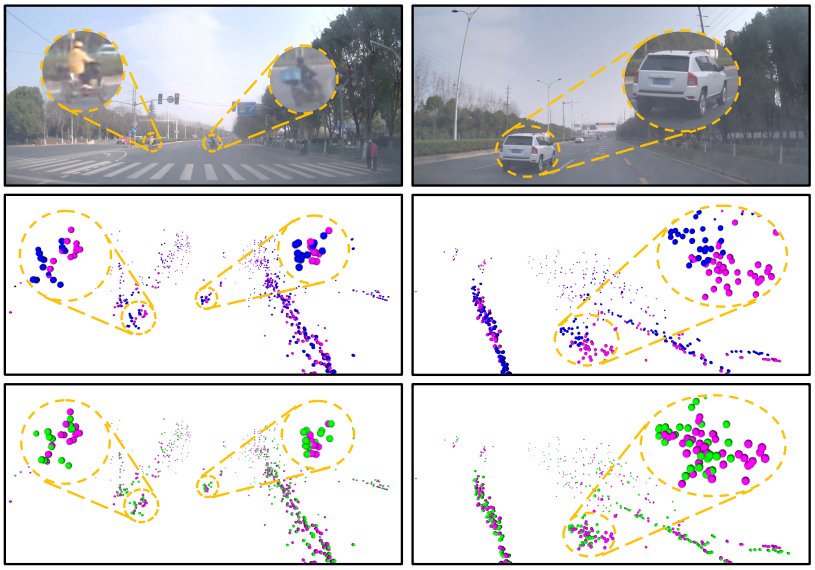


Figure 3.4: Scene flow estimation visualization. Figures on the top are the corresponding images captured by camera. The middle row shows points from $\mathbf{P}^s$ (blue) and $\mathbf{P}^t$ (magenta) respectively, while the bottom figures are the warped point cloud $\mathbf{P}'$ (green) and point cloud $\mathbf{P}^t$. Yellow circles denote zooming-in operation. It can be seen that, after applying the predicted scene flow, the warped points (green) are clearly closer to the (magenta) points in the next frame, either for the moving vehicles, motorcycles or the static road railing. This point alignment across frames demonstrates the accurate radar scene flow estimation.

optimal RNE on average by sacrificing a little RNE performance (-0.007m) of dynamic points for a large improvement margin (+0.017m) of static points. If the performance on dynamic points is favored by the user, the hyperparameter ζ (the threshold to classify points as moving) can tuned to be smaller during inference.

Despite the satisfying performance achieved on dense point clouds (e.g., LiDAR), state-of-the-art methods all struggle in delivering the same efficacy on radar data. Interestingly, the traditional ICP [235] achieves better results than other baselines on Avg. RNE because it can solve accurate static flow vectors in many simple scenarios (e.g., ego-vehicle is stationary). Nevertheless, ICP falls behind on 50-50 RNE due to its incompatibility to tackle moving points.

3.2.3 Qualitative Results

To intuitively show the performance of `RaFlow`, we follow [55] and visualize our scene flow estimation results by warping point cloud $\mathbf{P}'$ with the scene flow in Figure 3.4.

RRV	RCS	Power	Avg. RNE [m]↓	SAS↑	RAS↑
✓	✓		0.247	0.305	0.657
✓		✓	0.246	0.302	0.658
	✓	✓	0.362	0.204	0.419
✓	✓	✓	**0.240**	**0.316**	**0.675**

Table 3.5: Ablation results of input features.

Losses	Avg. RNE [m]↓	SAS↑	RAS↑
$\mathcal{L}_c + \mathcal{L}_{ss} + \mathcal{L}_{rd}$	0.450	0.165	0.421
$\mathcal{L}_{sc} + \mathcal{L}_s + \mathcal{L}_{rd}$	0.242	0.308	0.675
$\mathcal{L}_{sc} + \mathcal{L}_{rd}$	0.255	0.296	0.652
$\mathcal{L}_{ss} + \mathcal{L}_{rd}$	0.257	0.281	0.635
$\mathcal{L}_{sc} + \mathcal{L}_{ss}$	0.405	0.236	0.375
$\mathcal{L}_{sc} + \mathcal{L}_{ss} + \mathcal{L}_{rd}$	**0.240**	**0.316**	**0.675**

Table 3.6: Ablation results of loss functions.

In the scene described by the left column, our ego-vehicle is stationary while two highlighted motorcycles are moving forward. In the right scene, one car is driving forward at a slower speed than ours. It is clear that our method can correctly estimate the flow vectors of both static and moving points whether the ego-vehicle is moving or keeps still.

3.2.4 Ablation Study

Impact of Features. Besides the 3D positional information, we use the RRV, RCS and Power measurements as auxiliary input features which are unique strengths of radar. To understand their impacts, we ablate each component and report the performance change. As shown in Table 3.5, removing the RRV feature has the largest impact on our method. We attribute this to the fact, that under the supervision of our radial replacement loss, our model can implicitly learn to exploit the motion cues provided by the RRV feature (c.f. Sec. 3.1.5). The RCS and Power features can also make non-negligible contributions to our method because of the extra semantic information provided by them.

Impact of Losses. To validate the individual effectiveness of our three specific loss functions, we also conduct ablation studies with different combinations of them. Moreover, we also include the previous Chamfer loss $\mathcal{L}_c$ and smoothness term $\mathcal{L}_s$ used

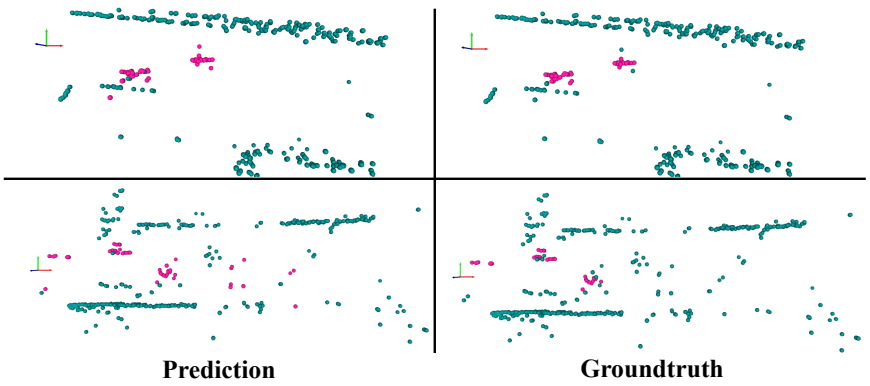


Figure 3.5: Visualization of motion segmentation results. Left column shows our prediction while right column is the ground truth. Moving and stationary points are rendered as pink and teal, respectively. Note that this is non-trivial as the ego-vehicle is also *moving* in both two scenes. (Best viewed in color)

by [56, 57] as reference for our loss functions. The results can be seen in Table 3.6. The bottom combination is our `RaFlow` with full loss functions. It is clear that each loss term improves the overall performance. Specifically, the radial displacement loss $\mathcal{L}_{rd}$ counts for more than another two losses due to its strong supervision in the radial direction. When replacing the previous *hard* Chamfer loss $\mathcal{L}_c$ with our soft Chamfer loss $\mathcal{L}_{sc}$, the results are upgraded by a large margin. This confirms the effectiveness of our proposed a) outlier discarding scheme that loosens the strong assumption imposed by absolute bijective mapping and b) small error toleration design where only errors originated from wrong correspondence matching are considered for network update. Last but not least, our spatial smoothness loss $\mathcal{L}_{ss}$ also shows advantages over the prior smoothness term.

3.2.5 Downstream Motion Segmentation Task

As one of the outputs of our scene flow estimation pipeline, the generated static mask (c.f. Sec. 3.1.4) can be used to segment scene points into stationary and moving points. Following the metrics used by [61], we evaluate our model on the test set for the motion segmentation task. As a result, an accuracy score of 81.9% can be achieved with a mIoU score of 47.9% and a sensitivity of 82.7%. Qualitative results for two scenes can be seen in Figure 5.1.3.

3.3 Conclusion and Limitations

In this chapter, we present a self-supervised learning method called `RaFlow` to estimate scene flow on 4D radar point clouds. A novel architecture and three loss functions are specifically designed to address the challenges induced by the characteristics of radar sensors. We collect a multi-modal dataset by driving a vehicle in the wild and compare `RaFlow` with state-of-the-art point-based scene flow methods. We validate the effectiveness of our designs in different aspects and show that the result of `RaFlow` can enable downstream motion segmentation tasks.

As the first attempt of radar scene flow estimation, our method has room to improve. First, our performance is still somewhat limited due to the lack of real supervision signals. Better accuracy might be needed in practice to unlock a wider range of downstream tasks, e.g., multi-object tracking and point cloud stitching. To this end, cross-modal supervision signals from other co-located sensors (e.g., RGB camera or IMU) will be exploited in conjunction with our self-supervised learning in Chapter 4 to further improve the flow estimation accuracy. Second, as we discussed in Sec. 3.2.2, due to the introduction of the SFR module, the performance on a few moving points is sacrificed to trade for a larger performance gain on average. Considering the importance of the moving points for autonomous driving, more investigation is needed to allow adaptive thresholding of ζ in response to different road situations on-the-fly. Third, the constant velocity assumption introduced in Sec. 3.1.3 might not hold in some cases where the vehicle platform has a large acceleration between two frames. Higher radar sampling rates or short time intervals between frames are needed to mitigate such extreme situations.

Chapter 4

4D Radar Scene Flow Learning using Cross-modal Supervision

In Chapter 3, we introduced a self-supervised learning pipeline for 4D radar-based scene flow estimation. However, as discussed in Sec. 3.3, self-supervised learning is inherently limited by the absence of real supervision signals, which constrains its performance. While supervised learning methods can achieve higher accuracy, they require costly manual annotations, making large-scale deployment impractical. In this chapter, we aim to improve scene flow estimation performance without requiring manual labels, thereby breaking the trade-off between annotation efforts and performance. This chapter introduces a novel approach to 4D radar-based scene flow estimation via cross-modal learning (c.f. Fig. 4.1), inspired by the co-located sensing redundancy in modern autonomous vehicles. This redundancy implicitly provides supervision cues for radar-based scene flow learning, which we exploit to develop an effective training paradigm. Specifically, we introduce a multitask model architecture for the identified cross-modal learning problem and propose loss functions to opportunistically engage scene flow estimation using multiple cross-modal constraints for effective model training. The chapter is organized as follows. The proposed multi-task model architecture and cross-modal loss functions are detailed in Sec. 4.1. Sec. 4.2 presents extensive evaluations of our method on a public dataset, followed by the conclusion in Sec. 4.3. Finally, Sec.4.4 discusses the limitations and potential directions for future work on scene flow estimation based on 4D radar.

Statement of Contribution. This chapter is based on the following publication: "Hidden Gems: 4D Radar Scene Flow Learning Using Cross-Modal Supervision", Fangqiang Ding, Andras Palffy, Dariu M. Gavrila, Chris Xiaoxuan Lu, *IEEE/CVF Conference on*

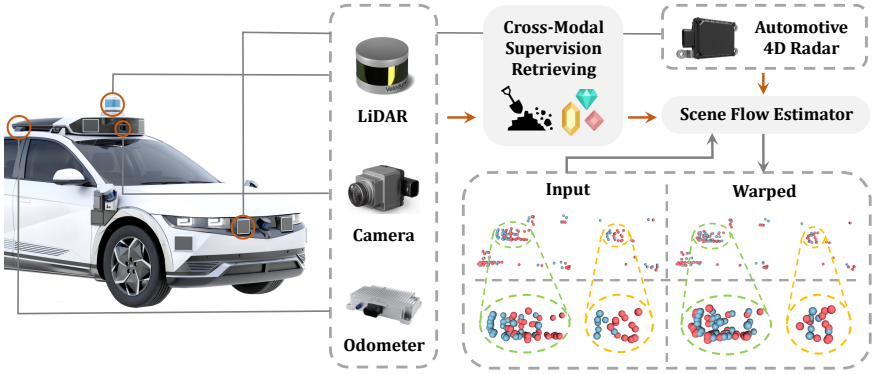


Figure 4.1: Cross-modal supervision cues are retrieved from co-located odometer, LiDAR and camera sensors to benefit 4D radar scene flow learning. The source point cloud (red) is warped with our estimated scene flow and gets closer to the target one (blue).

Computer Vision and Pattern Recognition (2023). As the first author, I designed the cross-supervised learning framework, implemented the method, conducted all experiments, and wrote the initial draft of all sections of the paper. My co-authors Andras Palffy and Dariu M. Gavrila provided additional unannotated part of VoD [81] to support part of my experiments and gave feedback on revisions to my manuscript. Chris Xiaoxuan Lu contributed valuable insights into the method's development and experimental design and helped refine the paper to improve readability and clarity.

4.1 Method

4.1.1 Overview

The scene flow estimation problem for our 4D radar context is defined in Sec. 1.4. The overall pipeline of our proposed method is depicted in Fig. 4.2. To bootstrap cross-modal supervised learning, we apply a multi-task model that predicts radar scene flow in two stages. The first stage starts by extracting base features with two input point clouds, which are then forwarded to two independent heads to infer per-point moving probabilities and initial point-wise scene flow vectors. On top of them, the second stage first infers a rigid transformation respective to radar ego-motion and outputs a binary motion segmentation mask. Then, the final scene flow is obtained by refining the flow vectors of identified static points with the derived rigid transformation. In summary, our multi-task model's outputs include a rigid transformation (i.e., the ego-motion), a motion segmentation mask (i.e., which targets are static or dynamic), and the final

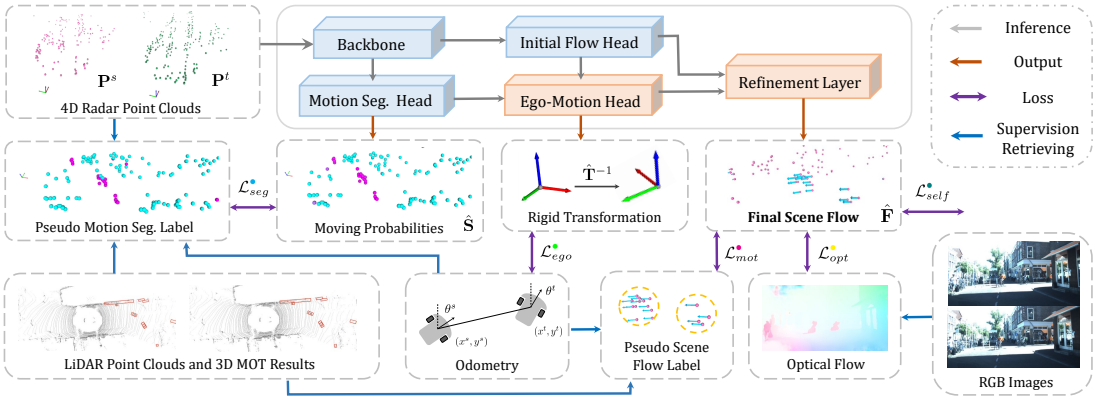


Figure 4.2: Cross-modal supervised learning pipeline for 4D radar scene flow estimation. The model architecture (c.f. Sec. 4.1.2) is composed of two stages (blue/orange block colours for stage 1/2) and outputs the final scene flow together with the motion segmentation and a rigid ego-motion transformation. Cross-modal supervision signals are utilized to constrain outputs with various loss functions (c.f. Sec. 4.1.3).

refined scene flow.

To supervise these predictions, we extract corresponding supervision signals from co-located modalities and train the entire model end-to-end by minimizing a loss $\mathcal{L}$ composed of three terms:

$$\mathcal{L} = \mathcal{L}_{ego}^{\bullet} + \mathcal{L}_{seg}^{\bullet} + \mathcal{L}_{flow}^{\bullet} \,. \tag{4.1}$$

Here, $\mathcal{L}_{ego}^{\bullet}$ is the ego-motion error that supervises the rigid transformation estimation using the odometry information. Our motion segmentation error $\mathcal{L}_{seg}^{\bullet}$ constrains the predicted moving probabilities with a point-wise pseudo motion segmentation label, which is obtained by fusing information from the odometer and LiDAR. We further supervise the final scene flow with signals given by LiDAR and RGB camera in $\mathcal{L}_{flow}^{\bullet}$. In the following section (Sec. 4.1.2), we first briefly introduce each module of our model. We then explain how we extract signals from co-located modalities to supervise our outputs in the training phase (Sec. 4.1.3). More details on our method can be found in the supplementary materials.

4.1.2 Model Architecture

Similar to [58, 61, 124, 176, 182], our model is designed in a two-stage fashion, with a rough initial flow derived in the first stage and refined in the second stage to obtain the final estimate of scene flow, as shown in Fig. 4.2. A description of all components of

our model can be found below.

Backbone. Following [55, 56, 124, 176], our backbone network directly operates on unordered point clouds $\mathbf{P}^s$ and $\mathbf{P}^t$ to encode point-wise latent features. In particular, we first apply the *set conv* layers [55] to robustly extract multi-scale local features for individual point clouds and propagate features from $\mathbf{P}^t$ to $\mathbf{P}^s$ using the *cost volume* layer [56] for feature correlation. We then concatenate multi-stage features of $\mathbf{P}^s$ (including the correlated features) and forward them into another multi-scale *set conv* layers to generate the base backbone features $\mathbf{E} \in \mathbb{R}^{N \times C_e}$. Specifically, the max-pooling operation is used along the channel axis to restore the global scene features after each *set conv* layer, which are then concatenated to per-point local features.

Initial flow and motion segmentation head. Given the base backbone features $\mathbf{E}$ that represents the inter-frame motion and intra-frame local-global information for each point $\mathbf{p}_i^s \in \mathbf{P}^s$, we apply two task-specific heads for decoding. The first head is to produce an initial scene flow $\hat{\mathbf{F}}^{init} = \{\hat{\mathbf{f}}_i^{init} \in \mathbb{R}^3\}_{i=1}^N$, while another is for generating a probability map $\hat{\mathbf{S}} = \{\hat{s}_i \in [0,1]\}_{i=1}^N$ that denotes the moving probabilities of points in $\mathbf{P}^s$ (*w.r.t.* the world frame). We implement both heads with a multi-layer perceptron and map the output to probabilities with $Sigmoid(\cdot)$ in the motion segmentation head.

Ego-motion head. With the natural correspondences $\{\mathbf{c}_i^s, \mathbf{c}_i^s + \hat{\mathbf{f}}_i^{init}\}_{i=1}^N$ formed by the initial scene flow $\hat{\mathbf{F}}^{init}$ between two frames and the probability map $\hat{\mathbf{S}}$[1], we restore a rigid transformation $\hat{\mathbf{T}} \in \mathbb{R}^{4\times4}$ that describes the radar ego-motion using the differentiable weighted Kabsch algorithm [183]. To mitigate the impact of the flow vectors from moving points, we compute $1 - \hat{\mathbf{S}}$ as our weights and normalize the sum of them to 1. Besides, we generate a binary motion segmentation mask by thresholding $\hat{\mathbf{S}}$ with a fixed value η_b to indicate the moving status of each point. We use this binary mask as our motion segmentation output and to identify stationary points for flow refinement below.

Refinement layer. As the flow vectors of static points are only caused by the radar's ego-motion, we can regularize their initial predictions with the more reliable rigid transformation $\hat{\mathbf{T}}$. The refinement operation is simply replacing the initial flow vector $\hat{\mathbf{f}}_i^{init}$ of identified stationary points with the flow vector induced by radar's ego-motion, which is derived as $[\hat{\mathbf{f}}_i \ 1]^\top = (\hat{\mathbf{T}} - \mathbf{I}_4)[\mathbf{c}_i^s \ 1]^\top$. The final scene flow is attained as $\hat{\mathbf{F}} = \{\hat{\mathbf{f}}_i \in \mathbb{R}^3\}_{i=1}^N$.

Temporal update module. Apart from the aforementioned basic modules, we also

[1]Specifically, we use the pseudo label $\bar{\mathbf{S}}$ (c.f. Sec. 4.1.3) instead of $\hat{\mathbf{S}}$ in the ego-motion head during training for stable scene flow learning.

propose an optional module that can be embedded into the backbone to propagate previous latent information to the current frame. More specifically, we apply a GRU network [237] that treats the global features in the backbone as the hidden state and update it temporally across frames. During training, we first split long training sequences into mini-clips with a fixed length T and train with mini-batches of them. The hidden state is initialized as a zero vector for the first frame of each clip. When evaluating on test sequences, we emulate the training conditions by re-initializing the hidden state after T frames.

In general, our model can deliver solutions to three different tasks, i.e., scene flow estimation, ego-motion estimation and motion segmentation, with 4D radar point clouds. Specifically, these outputs are compactly correlated to each other. For example, accurate motion segmentation results will benefit ego-motion estimation, which further largely determines the quality of the final scene flow.

4.1.3 Cross-Modal Supervision Retrieving

A key step in our proposed architecture is to retrieve the cross-modal supervision signals from three co-located sensors on autonomous vehicles, i.e., odometer, LiDAR and camera, to support model training without human annotation. This essentially leads to a multi-task learning problem. Of course, the supervision signals from individual modality-specific tasks (e.g., optical flow) are inevitably noisier than human annotation. However, we argue that if these noisy supervision signals are well combined, then the overall noise in supervision can be suppressed and give rise to effective training anyway. In the following, we detail how we extract cross-modal supervision signals and subtly combine them to formulate a multi-task learning problem.

Ego-motion loss. To supervise the rigid transformation $\hat{\mathbf{T}}$ derived in our ego-motion head, it is intuitive to leverage the odometry information from the odometer (GPS/INS). As a key sensor for mobile autonomy, the odometer can output high-frequency ego-vehicle poses, which can be used to compute the *pseudo* ground truth radar ego-motion transformation $\mathbf{O} \in \mathbb{R}^{4\times4}$ between two frames. In our case, the odometry information is directly provided by the *View-of-Delft* dataset [81] as a filtered combination of RTK GPS, IMU, and wheel odometry ($\sim$100 Hz), without any additional post-processing, and is used to compute frame-to-frame transformations. The ground truth rigid transformation $\mathbf{T} = \mathbf{O}^{-1}$ can then be derived to summarize the rigid flow component $\mathbf{F}^r = \{\mathbf{f}_i^r\}_{i=1}^N$ induced by the ground truth radar ego-motion, where $[\mathbf{f}_i^r\ 1]^\top = (\mathbf{T} - \mathbf{I}_4)[\mathbf{c}_i^s\ 1]^\top$. Our

ego-motion loss is formulated as:

$$\mathcal{L}_{ego} = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\mathbf{T}}[\mathbf{c}_i^s\ 1]^\top - \mathbf{T}[\mathbf{c}_i^s\ 1]^\top\right\|_2, \tag{4.2}$$

where we supervise the estimated $\hat{\mathbf{T}}$ by encouraging its associated rigid flow components to be close to the ground truth ones. By supervising $\hat{\mathbf{T}}$, we can implicitly constrain the initial scene flow $\hat{\mathbf{f}}_i^{init}$ for static points. More importantly, the static flow vectors in the final flow $\hat{\mathbf{F}}$ can also be supervised as the refinement is based on $\hat{\mathbf{T}}$.

Motion segmentation loss. Unlike ego-motion estimation, supervising motion segmentation with cross-modal data is not straightforward as no sensors provide such information. To utilize the odometry information, we generate a pseudo motion segmentation label with the rigid flow component $\mathbf{F}^r$ given by the odometer and the radar RRV measurements $\{v_i\}_{i=1}^N$. More specifically, we first approximate the RRV component ascribed to the radar ego-motion by $v_i^r \approx \mathbf{u}_i^\top \mathbf{f}_i^r/\Delta t$. Here, $\mathbf{u}_i$ is the unit vector with its direction pointing from the sensor to the point $\mathbf{c}_i^s$ and Δt is time duration between two frames. Then, we compensate the radar ego-motion and get the object absolute radial velocity $\Delta v_i = \mathrm{abs}(v_i - v_i^r)$. With per-point Δv_i, the pseudo motion segmentation label $\mathbf{S}^v = \{s_i^v \in \{0,1\}\}_{i=1}^N$ can be derived by thresholding, where 1 denotes moving points. More details on our thresholding strategy can be found in the Appendix A.4. Note that one shortcoming is that tangentially moving targets are not distinguished.

Besides the odometry information, we also leverage the LiDAR data to generate a pseudo foreground (movable objects) segmentation label $\mathbf{S}^{fg}$. To this end, we first feed LiDAR point clouds into an off-the-shelf 3D multi-object tracking (MOT) pretrained model [71]. Then we divide radar points from $\mathbf{P}^s$ into the foreground and background using the bounding boxes (e.g. pedestrian, car, cyclist) produced by 3D MOT to create $\mathbf{S}^{fg}$. Besides, we can also assign pseudo scene flow vector label to foreground points by: a) retrieving the ID for each bounding box from the first frame, b) computing the inter-frame transformation for them, c) deriving the translation vector for each inbox point based on the assumption that all points belonging to the same object share a *universal* rigid transformation. The resulting pseudo scene flow label is denoted as $\mathbf{F}^{fg} = \{\mathbf{f}_i^{fg}\}_{i=1}^N$, where we leave the label empty for all identified background points.

Given the two segmentation labels $\mathbf{S}^v$ and $\mathbf{S}^{fg}$, directly fusing them is impeded by their domain discrepancy, i.e., not all foreground points are moving. Therefore, we propose to distill the moving points from $\mathbf{S}^{fg}$ by discarding foreground points that either keep still or move very slowly. We implement this by removing the rigid flow component $\mathbf{F}^r$ from $\mathbf{F}^{fg}$ and get the non-rigid flow component of all foreground points.

Then we obtain a new pseudo motion segmentation label $\mathbf{S}^l = \{s_i^l \in \{0,1\}\}_{i=1}^N$ by classifying points with apparent non-rigid flow as dynamic. A more reliable pseudo motion segmentation $\mathbf{S} = \{s_i\}_{i=1}^N$ can be consequently obtained by fusing $\mathbf{S}^l$ and $\mathbf{S}^v$. For points classified as moving in $\mathbf{S}^l$, we have high confidence about their status and thus label these points as moving in $\mathbf{S}$. For the rest of the points, we label their s_i according to $\mathbf{S}^v$. Finally, as seen in Fig. 4.2, our motion segmentation loss can be formulated by encouraging the estimated moving probabilities $\hat{\mathbf{S}}$ to be close to the pseudo motion segmentation label $\mathbf{S}$:

$$\mathcal{L}_{seg}^{\bullet} = \frac{1}{2}\left(\frac{\sum_{i=1}^N (1-s_i)\log(1-\hat{s}_i)}{\sum_{i=1}^N (1-s_i)} + \frac{\sum_{i=1}^N s_i \log(\hat{s}_i)}{\sum_{i=1}^N s_i}\right) . \tag{4.3}$$

Here, we use the average loss of static and moving losses to address the class imbalance issue.

Scene flow loss. In the final scene flow output $\hat{\mathbf{F}}$, the flow vectors of static points have been implicitly supervised by the ego-motion loss (c.f. Eq. (4.2)). In order to further constrain the flow vectors of moving points, we formulate two new loss functions. The first one is $\mathcal{L}_{mot}^{\bullet}$, which is based on the pseudo scene flow label $\mathbf{F}^{fg}$ derived from 3D MOT results. In this loss, we only constrain the flow vectors of those moving points identified in $\mathbf{S}^l$ through:

$$\mathcal{L}_{mot}^{\bullet} = \frac{1}{\sum_{i=1}^N s_i^l} \sum_{i=1}^N \left\| s_i^l (\hat{\mathbf{f}}_i - \mathbf{f}_i^{fg}) \right\|_2 . \tag{4.4}$$

In addition to utilizing the odometer and LiDAR for cross-modal supervision, we also propose to extract supervision signals from the RGB camera. Specifically, we formulate a loss $\mathcal{L}_{opt}^{\bullet}$ using pseudo optical flow labels $\mathbf{W} = \{\mathbf{w}_i \in \mathbb{R}^2\}_{i=1}^N$. To get this pseudo label, we first feed synchronized RGB images into a pretrained optical flow estimation model [238] to produce an optical flow image. Then, we project the coordinate $\mathbf{c}_i^s$ of each point onto the image plane and draw the optical flow vector at the corresponding pixel $\mathbf{m}_i$ of each point. Given $\mathbf{W}$, it is intuitive to construct the supervision for our scene flow prediction by projecting it on the image plane. However, minimizing the flow divergence in pixel scale has less impact for far radar points due to the depth-unawareness during perspective projection. Instead, we directly take the point-to-ray distance as the training objective, which is more insensitive to points at different ranges. The loss function can be written as:

$$\mathcal{L}_{opt}^{\bullet} = \frac{1}{\sum_{i=1}^N s_i} \sum_{i=1}^N s_i \mathcal{D}(\mathbf{c}_i^s + \hat{\mathbf{f}}_i, \mathbf{m}_i + \mathbf{w}_i; \theta) \tag{4.5}$$

Method	Sup.	EPE [m]↓	AccS↑	AccR↑	RNE [m]↓	MRNE [m]↓	SRNE [m]↓
ICP [239]	None	0.344	0.019	0.106	0.138	0.148	0.137
Graph Prior* [164]	None	0.445	0.070	0.104	0.179	0.186	0.176
JGWTF* [59]	Self	0.375	0.022	0.103	0.150	0.139	0.151
PointPWC [56]	Self	0.422	0.026	0.113	0.169	0.154	0.170
FlowStep3D [57]	Self	0.292	0.034	0.161	0.117	0.130	0.115
SLIM* [61]	Self	0.323	0.050	0.170	0.130	0.151	0.126
RaFlow [124]	Self	0.226	0.190	0.390	0.090	0.114	0.087
CMFlow	Cross	0.141	**0.233**	0.499	0.057	0.073	0.054
CMFlow (T)	Cross	**0.130**	0.228	**0.539**	**0.052**	**0.072**	**0.049**

Table 4.1: Scene flow evaluation results on the *Test* set. The mean metric values across all test frames are reported. * indicates that we reproduce these methods referring to original papers since their source codes are not public. The best results on each metric are shown in **bold**. ↑ means bigger values are better, and vice versa.

where $\mathcal{D}$ denotes the operation that computes the distance between the warped point $\mathbf{c}_i^s + \hat{\mathbf{f}}_i$ and the ray traced from the warped pixel $\mathbf{m}_i + \mathbf{w}_i$. θ denotes sensor calibration parameters. Note that we only consider the scene flow of moving points identified in $\mathbf{S}$ here. Apart from the above two loss functions used to constrain the final scene flow, we also employ the self-supervised loss $\mathcal{L}_{self}^{\bullet}$ from the last chapter (c.f. Sec. 3.1.5) to complement our cross-modal supervision. See the Appendix A for more details. The overall scene flow loss is formulated as:

$$\mathcal{L}_{flow}^{\bullet} = \mathcal{L}_{mot}^{\bullet} + \lambda_{opt}\mathcal{L}_{opt}^{\bullet} + \mathcal{L}_{self}^{\bullet} \ , \tag{4.6}$$

where we set the weight $\lambda_{opt} = 0.1$ in all our experiments.

4.2 Experiments

4.2.1 Experimental Setup

Dataset. For our experiments, we use the *View-of-Delft* (VoD) dataset [81], which provides synchronized and calibrated data captured by co-located sensors, including a 64-beam LiDAR, an RGB camera, RTK-GPS/IMU based odometer and a 4D radar sensor. As is often the case with datasets focused on object recognition, the test set annotations of the official VoD dataset are withheld for benchmarking. However, our task requires custom scene flow metrics for which we need annotations to generate ground truth labels. Therefore, we divide new splits ourselves from the official sets

	O	L	C	EPE [m]↓	AccS↑	AccR↑	RNE [m]↓
(a)				0.228	0.184	0.392	0.091
(b)	✓			0.161	0.203	0.442	0.065
(c)	✓	✓		0.145	0.228	0.482	0.058
(d)	✓		✓	0.159	0.216	0.458	0.064
(e)	✓	✓	✓	**0.141**	**0.233**	**0.499**	**0.057**

Table 4.2: Ablation experiments on combing supervision signals from diverse modalities. Abbreviations: odometer (O), LiDAR (L), camera (C). Note that we disable the temporal update scheme in this study to highlight the impact of modalities for training.

(i.e., *Test*, *Val*, *Train*) to support our evaluation. Given sequences of data frames, we form pairs of consecutive radar point clouds as our scene flow samples for training and inference. We only generate ground truth scene flow and motion segmentation labels for samples from our *Val*, *Test* sets and leave the *Train* set unlabelled as our method has no need for ground truth scene flow annotations for training. Please see the Appendix A on our dataset separation and labelling process.

Metrics. We use three standard metrics [55, 56, 59] to evaluate different methods on scene flow estimation, including a) *EPE* [m]: average end-point-error (L_2 distance) between ground truth scene flow vectors and predictions, b) *AccS/AccR*: the ratio of points that meet a strict/relaxed condition, i.e. *EPE* $<$ 0.05/0.1 m or the relative error $<$ 5%/10%. We also use the c) *RNE* [m] metric [124] that computes resolution-normalized *EPE* by dividing *EPE* by the ratio of 4D radar and LiDAR resolution. This can induce sensor-specific consideration and maintain a fair comparison between different sensors. Besides, we compute the *RNE* [m] for moving points and static points respectively, and denote them as *MRNE* [m] and *SRNE* [m].

Baselines. For overall comparison, we apply seven state-of-the-art methods as our baselines, including five self-supervised learning-based methods [56, 57, 59, 61, 124] and two non-learning-based methods [164, 239], as seen in Tab. 4.1. To keep a fair comparison, we use their default hyperparameter settings. We retrain scene flow models offline on our VoD *Train* set for learning-based methods and directly optimize scene flow results online for non-learning-based ones.

Implementation details. We use the Adam optimizer [236] to train all models in our experiments. The learning rate is initially set as 0.001 and exponentially decays by 0.9 per epoch. The *Val* set is used for both model selection during training and for determining the values of our hyperparameters. We set our classification threshold

Figure 4.3: Illustration of the causes of noisy supervision signals from camera and LiDAR. The top row shows an example of the noisy optical flow estimation on RGB images. The bottom row exhibits unreliable object recognition on LiDAR point clouds. We enlarge regions of interest and mark them with amber circles.

$\eta_b = 0.5$ (c.f. Appendix A.3) for all experiments. When activating our temporal update module, the length of mini-clips T is set as 5. Please refer to Appendix A for our hyperparameter searching process.

4.2.2 Scene Flow Evaluation

Overall results. We quantitatively compare the performance of our methods to baselines on the *Test* set, as shown in Tab. 4.1. Compared with both non-learning-based and self-supervised counterparts, CMFlow shows remarkable improvement on all metrics by leveraging cross-modal supervision signals for training. Our method outperforms the second-best approach [124] by 37.6% on EPE, implying drastically more reliability. Our performance is further improved when adding the temporal update scheme (c.f. Sec. 4.1.2) in the backbone, i.e., CMFlow (T). We also observe that the performance slightly degrades on the AccS metric when activating the temporal update. This suggests us that introducing temporal information will somewhat disturb the original features and thus reduce the fraction of already fine-grained (e.g. EPE$<$0.05 m) predictions.

Impact of modalities. A key mission of this chapter is to investigate how supervision signals from different modalities help our radar scene flow estimation. To analyze their impact, we conduct ablation studies and show the breakdown results in Tab. 4.2. The results show that the odometer leads to the biggest performance gain for our method (c.f. row (b)). This can be credited to the fact, that most measured points are static (e.g. 90.5% for the *Test* set) and their flow predictions can be supervised well by the ego-motion supervision (Eq. (4.2)). Moreover, the odometer guides the generation

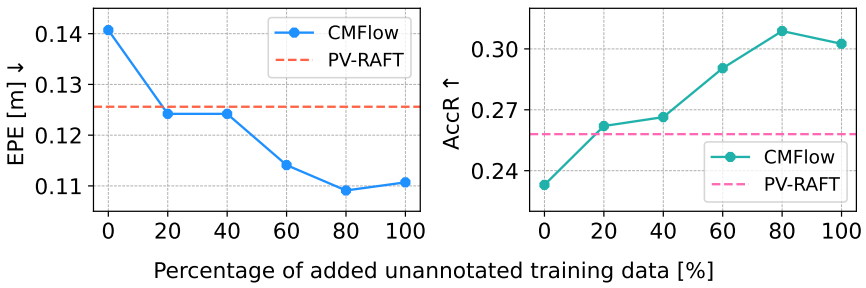


Figure 4.4: Analysis of the performance when adding more unannotated training data. For the training of CMFlow, we retain the samples from the *Train* set and add different percentages of data from the extra unannotated VoD part, which provides ~ 28.5k more training samples.

of pseudo motion segmentation labels (Sec. 4.1.3), which is indispensable for our two-stage model architecture.

Based on row (b), both LiDAR and camera contribute to a further improvement on all metrics, as seen in row (c)-(e). These results validate that our method can effectively exploit useful supervision signals from each modality to facilitate better scene flow estimation. However, compared to that from the odometer, the gains brought by these two sensors are smaller, especially for the camera, which only increases the AccS by 1.3%. In our opinion, the reason for this is two-fold. First, in Eq. (4.4) and Eq. (4.5), the pseudo scene/optical flow labels only constrain the flow vectors of identified moving points, which are significantly fewer than static ones. Thus, they have limited influence on the overall performance. Second, the supervision signals extracted from these sensor modalities could be noisy, as exhibited in Fig. 4.3. The dashboard reflections on the car windscreen severely disturb the optical flow estimation on images, which further results in erroneous supervision in Eq. (4.5). As for LiDAR point clouds, both inaccurate object localization and false negative detections occur in the 3D MOT results, which not only affect the generation of pseudo motion segmentation labels (Eq. (4.3)) but bring incorrect constraints to scene flow in Eq. (4.4).

Impact of the amount of unannotated data. Being able to use more unannotated data for training is an inherent advantage of our cross-modal supervised method compared to fully-supervised ones as no additional annotation efforts are required. Here, we are interested in if the performance of CMFlow could surpass that of fully-supervised methods when more unannotated data is available for training. To this end, we further

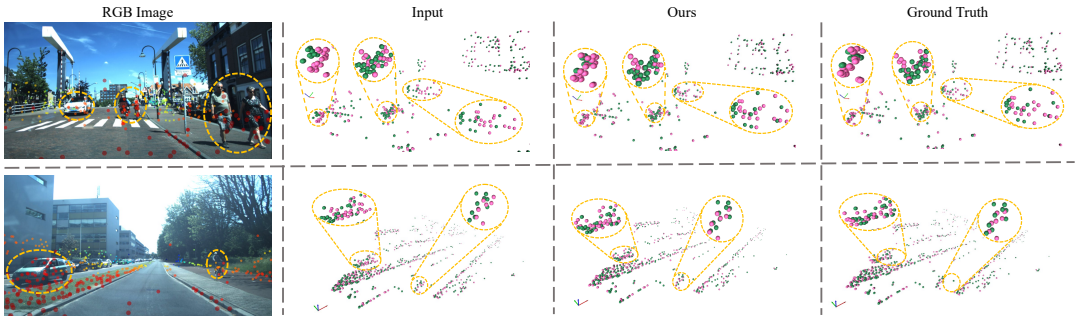


Figure 4.5: Qualitative scene flow results in two scenes. From left to right: 1) radar points from the source frame projected to the corresponding RGB image (points are coloured by distance from the sensor), 2) two input radar point clouds, the source one (pink) and the target one (green), 3) the source point cloud warped by our predicted scene flow and the target radar point cloud, 4) the source point cloud warped by ground truth scene flow and the target one. We mark dynamic objects in amber and apply the zooming-in operation for them.

mix in an extra amount of unlabeled data provided by the VoD dataset [81][2] in our cross-modal training sets. As for fully-supervised methods, we select the state-of-the-art method, PV-RAFT [175], for comparison. As PV-RAFT needs ground truth scene flow annotations for training, we utilize all available annotated samples from the *Train* set for it. The analysis of the correlation between the percentage of added unannotated training data and the performance is shown in Fig. 4.4. As we can see, the performance of CMFlow improves by a large margin on both two metrics by using extra training data. In particular, after adding only 20% of the extra unannotated training samples ($\sim$140% more than the number of training samples used for PV-RAFT), CMFlow can already outperform PV-RAFT [175] trained with less annotated samples. This implies the promise of our method for utilizing a large amount of unannotated data in the wild.

Qualitative results. In Fig. 4.5, we showcase example scene flow results of CMFlow (trained with the extra training samples) compared to the ground truth scene flow. By applying the estimated scene flow (i.e. moving each point in the source frame by its estimated 3D motion vector), both the static background and multiple objects with different motion patterns are aligned well between two frames. It can also be observed that our method can give accurate scene flow predictions, close to the ground truth one.

Runtime efficiency. We evaluate the computational efficiency of CMFlow on a PC

[2]This part is currently in a beta testing phase, only available for selected research groups. Except for having no object annotations, it provides the same modalities of input data as the official one and has $\sim$ 28.5k frames.

	Label $\mathbf{S}^v$	Label $\mathbf{S}^l$	A.D.	mIoU (%)	Gain (%)
(a)				46.9	-
(b)	✓			52.8	+5.9
(c)	✓	✓		54.1	+1.3
(d)	✓	✓	✓	57.1	+3.0

Table 4.3: Motion segmentation evaluation. For row (a), we report the results of the self-supervised baseline [124]. In row (b), we replace $\mathbf{S}$ by $\mathbf{S}^v$ provided by odometer. A.D. denotes adding extra unannotated data (c.f. Fig. 4.4) for training.

with a single RTX 3090 GPU. To emulate real-world processing, one sample is fed into our model per time. Our method contains 4.23 million model parameters and performs one inference step in 0.069 seconds on average ($\sim$14 Hz). The maximum allocated GPU memory is 162 MB during inference. It can be seen that our method can achieve satisfactory real-time performance while having relatively low GPU memory consumption.

4.2.3 Subtask Evaluation

Motion segmentation evaluation. Apart from scene flow estimation, our multi-task model can additionally predict a motion segmentation mask that represents the real moving status of each radar point (c.f. Sec. 4.1.2). Here, we evaluate this prediction of CMFlow and analyze the impact of its performance in Tab. 4.3. Since this is a binary classification task for each point, the mean intersection over union (mIoU) is computed by taking the IoU of moving and static classes and averaging them. As the two ingredients to form the final pseudo motion segmentation label $\mathbf{S}$ used to supervise our output in Eq. (4.3), both $\mathbf{S}^v$ and $\mathbf{S}^l$ contributes to our performance improvement on the motion segmentation task (row (a)-(c)). Moreover, our mIoU further increases with extra training data to learn motion segmentation on 4D radar point clouds. We also visualize some qualitative motion segmentation results of row (d) in Fig. 4.6, where our method can segment moving points belonging to multiple dynamic objects accurately in complicated scenarios.

Discussion on ground truth quality. The ground truth labels for motion segmentation are generated by assigning radar points within annotated 3D bounding boxes of dynamic objects as moving. However, due to radar's low spatial resolution and multi-path effects, these box-based labels may miss a few genuinely moving points outside the boxes,

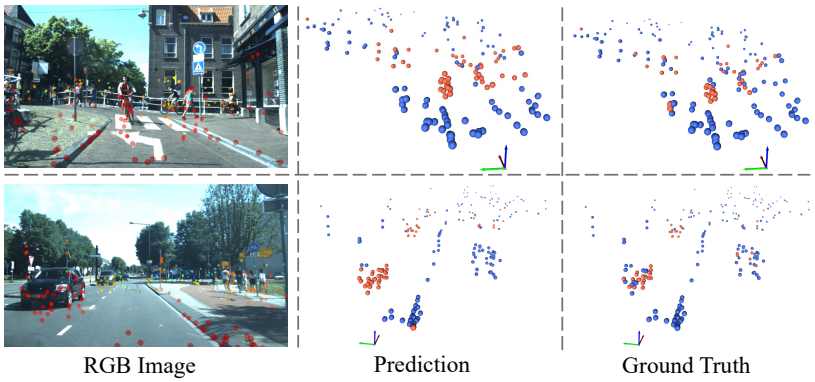


Figure 4.6: Visualization of motion segmentation. The left column shows the corresponding image with radar points (coloured by range) projected onto it. In the middle and right columns, moving points are shown in orange while static points are shown in blue.

introducing false negatives (c.f. Fig. 4.6). However, such inaccuracies represent only a small portion of the overall motion segmentation labels, and most labels remain consistent and informative. The substantial performance gain, e.g., a +5.9% mIoU improvement from row (a) to (b) (c.f. Tab. 4.3), demonstrates the effectiveness of incorporating cross-modal pseudo labels (e.g., from odometry and LiDAR) despite minor label noise. To further reduce this issue, one possible remedy is to slightly enlarge bounding boxes to capture potentially overlooked dynamic points.

Ego-motion estimation evaluation. One important feature of our method is that we can estimate a rigid transformation that represents relative radar ego-motion transform between two consecutive frames in dynamic scenes (c.f. Sec. 4.1.2). To demonstrate this feature, we evaluate this output on the *Test* set and show the ablation study results in Tab. 4.4 with two metrics: the relative translation error (RTE) and the relative angular error (RAE). With the cross-modal supervision from the odometry in row (b), we can directly constrain our ego-motion estimation in Eq. (4.2) and thus improve the performance by a large margin. Using LiDAR and camera supervision (c.f. row (c)) can also help as they lead to better motion segmentation and scene flow outputs, which further benefit the compactly associated ego-motion estimation. We also activate the temporal update module in the backbone, which also increases the overall performance. With our high-level accuracy on ego-motion estimation between consecutive frames, we are also interested in whether our results can be used for the more challenging long-term odometry task. We accumulate the inter-frame transformation estimations and plot two ego-vehicle trajectories in Fig. 4.7. Without any global optimization, our method can provide accurate long-term trajectory estimation in dynamic scenes by only estimating

	O	L + C	A.D.	T	RTE [m]	RAE [°]
(a)					0.090	0.336
(b)	✓				0.086	0.183
(c)	✓	✓			0.085	0.145
(d)	✓	✓	✓		0.071	**0.089**
(e)	✓	✓	✓	✓	**0.066**	0.090

Table 4.4: Evaluation of ego-motion estimation between two frames. T denotes we activate the temporal update module.

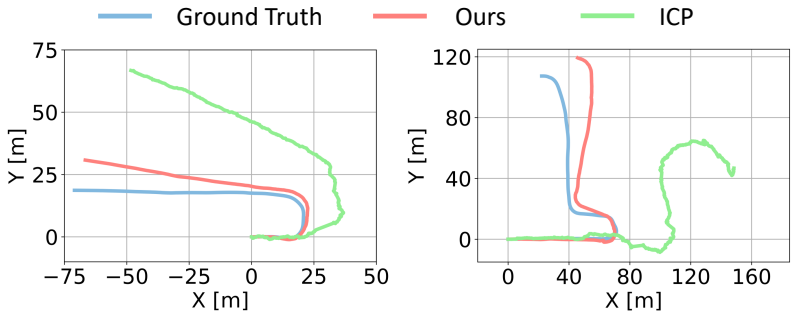


Figure 4.7: Odometry results as a byproduct of our scene flow estimation. The ground truth is generated using the RTK-GPS/IMU measurements. We plot the results on two challenging test sequences.

inter-frame transformations and remarkably outperform the ICP baseline [239].

Discussion on the ego-motion evaluation limitation. Given that most of the data collection scenarios in the VoD dataset [81] involve roads with minimal elevation changes and zero inclines, the dataset assumes a fixed vehicle height and constant roll and pitch angles. As a result, the ground truth contains no variation along the z-axis and in roll and pitch. Consequently, our ego-motion evaluation is inherently limited to three degrees of freedom: 2D translation (x, y) and yaw rotation. The long-term trajectory visualization in Fig. 4.7 is therefore restricted to the horizontal (XY) plane. Although our predicted motion may contain drift along the z-axis, this cannot be assessed due to the absence of corresponding ground truth. We acknowledge this as a limitation of our current evaluation setup. While our method estimates full 6-DoF motion, including vertical translation and full 3D rotation, the lack of ground truth prevents us from validating these components. Future work leveraging datasets with complete 6-DoF pose variations would enable a more thorough evaluation across all motion dimensions.

4.3 Conclusion

In this chapter, we presented a novel cross-modal supervised approach, CMFlow, for estimating 4D radar scene flows. CMFlow is unique in that it does not require manual annotation for training. Instead, it uses complementary supervision signals retrieved from co-located heterogeneous sensors, such as odometer, LiDAR and camera, to constrain the outputs from our multi-task model. Our experiments show that CMFlow outperforms our baseline methods in all metrics and can surpass the fully-supervised method when sufficient unannotated samples are used in our training. CMFlow can also improve two downstream tasks, i.e., motion segmentation and ego-motion estimation. We hope our work will inspire further investigation of cross-modal supervision for scene flow estimation and its application to more downstream tasks, such as multi-object tracking and point cloud accumulation.

4.4 Limitations and Future Works

Our proposed cross-supervised learning frameworks, enable effective scene flow learning on 4D radar point clouds without the need for manual annotations, providing robust reasoning for point-level scene dynamics (including scene flow and motion segmentation) and frame-level ego-agent motion estimation (odometry). Despite this progress, the real-world application of 4D radar-based scene flow still faces the following challenges.
Incompatibility with dense scene representation. The sparse, irregular point-level motion representation provided by scene flow vectors is incompatible with the dense, grid-based scene geometry/semantic representations (e.g., BEV maps [240], 3D occupancy grids [33]) required by downstream planning and decision-making modules [9]. This limits the seamless integration of 4D radar scene flow outputs into high-level autonomous driving systems. To address this problem, one possible solution is to build a dense scene flow map from sparse motion vectors output using discrete interval-based sampling combined with learning-based interpolation. Another direction is to densify the input 4D radar point cloud, via multi-frame accumulation and transform it into a dense representation via voxelization. However, the implementation of either solution remains non-trivial, and future efforts are needed to cope with associated challenges.
Lack of future scene flow prediction. An essential component of autonomous driving systems is the prediction module, which forecasts the future behavior of dynamic agents to anticipate potential risk and plan optimal trajectories. Scene flow prediction offers

fine-grained and point-level motion forecasts, complementing traditional trajectory prediction [241] by capturing subtle dynamics and including untracked entities to enhance the navigation safety in complex environments. While the work presented here estimates scene flow between the past and current 4D radar point clouds, the prediction of scene flow for the future frames has not been explored. Future research will focus on developing spatio-temporal models to predict future scene flow for 4D radar. Combined with the dense scene flow representations discussed earlier, this approach can further support downstream tasks like BEV map prediction and 3D occupancy forecasting.

Constrained scene dynamics representation. The depiction of scene dynamics is constrained by the sparse nature of 4D radar point clouds, which results from non-negligible signal loss during point cloud generation [48]. This incomplete scene representation can cause critical dynamic information to be overlooked, even when scene flow estimation is accurate. Inspired by the success of using raw radar data for 3D occupancy prediction (c.f. Chapter 6), future research will explore predicting scene flow directly from raw radar data, such as radar tensors, by estimating the displacement of each tensor element. While this approach can provide a more comprehensive representation of scene dynamics, it also introduces significant challenges, including the difficulty of extracting meaningful motion information from noisy radar tensor elements and the high computational complexity associated with processing large-scale tensor data.

Chapter 5

Moving Object Detection and Tracking with 4D Radar Point Cloud

In Chapter 3 and Chapter 4, we introduced self-supervised and cross-modal supervised learning approaches for 4D radar scene flow estimation, yielding robust reasoning of point-level scene dynamics and frame-level ego-vehicle motion. Beyond these, another critical aspect of dynamic environment perception is the ability to track moving objects at the object level. Robustly tracking moving objects in 3D space plays a pivotal role in subsequent autonomy tasks. While most existing methods rely on LiDARs or cameras for Multiple Object Tracking (MOT), the capabilities of 4D imaging radars remain largely unexplored. Traditional tracking-by-detection paradigm struggles when adapted to 4D radar data, due to the inherent radar noise and point sparsity, undermining accurate type classification and bounding box regression. In this chapter, we unveiled the untapped potential of 4D mmWave radars for multiple moving object tracking and tackle the inherent challenges posed by radar noise and point sparsity. We introduces an innovative solution tailored for radar-based tracking, called `RaTrack`, which bypasses the typical reliance on specific object types and 3D bounding boxes, focusing instead on motion segmentation and clustering. It also leverages the experience on scene flow from the past experience, to benefit the object detection and association. In this chapter, Sec. 5.1 presents the proposed `RaTrack` method, illustrating its components in sequence. Sec. 5.2 provides experimental results and analysis, followed by Sec. 5.3 that concludes the chapter and discussion the limitations and potential future works.

Statement of Contribution. This chapter is based on the following publication: "RaTrack: Moving Object Detection and Tracking with 4D Radar Point Cloud", Zhijun Pan, Fangqiang Ding, Haotao Zhong, Chris Xiaoxuan Lu, *IEEE International Conference on*

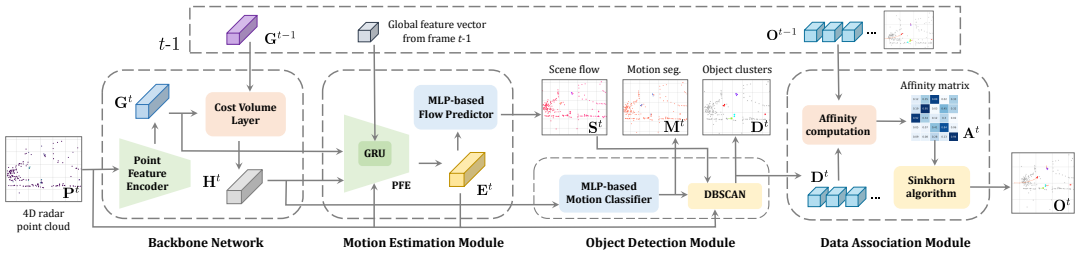


Figure 5.1: Overall network pipeline of RaTrack. Note that radar 3D points are shown in the bird's eye view for visualization.

Robotics and Automation (2024), where Zhijun Pan, Fangqiang Ding and Haotao Zhong are marked as equal contributions. As one of the leading authors, I proposed the key ideas of this work, developed the overall framework, implemented part of the model, designed the experimental process, and wrote the published paper. Another leading author, Zhijun Pan, significantly contributed to the implementation of our method and baselines, conducted the experiments, and evaluated the results. Haotao Zhong made non-trivial contributions by implementing some baseline methods, running experiments, and analyzing results. Chris Xiaoxuan Lu provided key insights into the development of our method, guided us during experiment evaluation and paper writing, and edited the manuscript to improve clarity, presentation, and technical accuracy.

5.1 Proposed Method

5.1.1 Overview

In this chapter, we conside the problem of online *moving object detection and tracking* with 4D automotive radar and the problem has been definied in Sec. 1.4. We introduce RaTrack, a generic learning-based framework bespoken for 4D radar-based moving object detection and tracking. As seen in Fig. 5.1, in our network pipeline, we first apply a backbone (c.f. Sec. 5.1.2) to encode intra- and inter-frame radar point cloud features. With the extracted features, our point-wise motion estimation module (c.f. Sec. 5.1.3) infers point-level scene flow as an explicit complement to augment the latent features of radar point clouds. Our advocated idea of class-agnostic detection without bounding boxes is introduced in the object detection module (c.f. Sec. 5.1.4), in which moving points are first identified and then used to detect moving objects via clustering. Finally, our data association module (c.f. Sec. 5.1.5) computes the affinity matrix with a learnable distance metric and then optimises the bipartite matching problem. The entire

network is end-to-end trainable with a multi-task loss that incorporates three supervised subtasks: motion segmentation, scene flow estimation, and affinity matrix computation.

5.1.2 Backbone Network

On receiving a new 4D radar point cloud $\mathbf{P^t}$, our backbone neural network is used to extract representative latent features for each radar point to facilitate subsequent tasks, e.g., motion segmentation and scene flow estimation. To this end, we first extract point-level local-global features $\mathbf{G}^t$ using a point feature encoder (PFE), which comprises a) three set abstraction layers [242] to extract local features at different scales in parallel, b) three MLP-based feature propagation layer to map local features into high-level representations, and c) the max-pooling operation to aggregate the global feature vector that is attached to per-point features. To further encode inter-frame point motion for the current frame, we recall the features $\mathbf{G}^{t-1}$ from the last frame and correlate features across two frames by the cost volume layer [56], as seen in Fig. 5.1. The output is the cost volume $\mathbf{H}^t$ that represents the motion information for each point in $\mathbf{P}^t$.

5.1.3 Motion Estimation Module

Radar sensors' inherent sparsity and noise result in latent features from radar point clouds that are deficient in informative geometric cues, complicating object detection and tracking. To address this, we introduce a point-wise motion estimation module that explicitly determines per-point motion vectors. Contrary to the conventional scene flow approach, which estimates per-point motion vectors in a forward direction (from frame t to $t+1$), we opt for a backward estimation (from frame t to $t-1$). This not only negates tracking latency but also ensures that the estimated flow vectors correspond to the points in the current frame t.

Before decoding scene flow, we first aggregate mixed features by integrating diverse backbone outputs: input point features $\mathbf{P}^t$, local-global features $\mathbf{G}^t$ and cost volume $\mathbf{H}^t$. Subsequently, another PFE is employed, aiming to facilitate information exchange among these diverse features and enhance their spatial coherence, producing the flow embedding $\mathbf{E}^t$ as shown in Fig. 5.1. Notably, within this PFE, a GRU network [237] is utilized to introduce temporal information into the global feature vector prior to its association with per-point features. The scene flow $\mathbf{S}^t = [\mathbf{s}_1^t; ...; \mathbf{s}_i^t; ...; \mathbf{s}_{N^t}^t] \in \mathbb{R}^{N^t \times 3}$ is finally decoded with an MLP-based flow predictor. This module plays a crucial role in our framework and is leveraged to augment per-point features before clustering (c.f.

Sec. 5.1.4) and to lead up data association as an extra motion clue (c.f. Sec. 5.1.5).

5.1.4 Object Detection Module

In the classical tracking-by-detection paradigm, objects are first detected as class-specific 3D bounding boxes based on which tracklets are built up across frames. However, the performance of such approaches inevitably relies on accurate type classification and bounding box regression, which are hard to accomplish from the sparse and noisy 4D radar data.

Rather than relying on the error-prone 3D bounding box detection tailored for specific object types, we emphasize the fundamental necessity to simply group scattered points into clusters for effective tracking. Consequently, we champion a class-agnostic object detection approach that eschews bounding boxes in our solution. By adopting this methodology, the fallible bounding box detector is replaced by a more dependable combination of motion segmentation and clustering, which proves to be particularly suited for radar point clouds. In other words, we detect objects in a bottom-up fashion, where points are first classified into moving and static (i.e., motion segmentation) and those close in the latent feature space are aggregated into object clusters. For motion segmentation, we leverage the cost volume $\mathbf{H}^t$ provided by the backbone and compute the moving possibility score c_i^t for each point $\mathbf{p}_i^t$ through an MLP-based motion classifier. The classification results are reliable enough as both the crucial RRV measurements and the inter-frame motion information are encoded by our backbone, yielding a robust motion representation. A fixed threshold ζ_{mov} is further used to separate the moving targets from the static background, resulting in a motion segmentation mask $\mathbf{M}^t = \{m_i^t \in \{0,1\}\}_{i=1}^{N_t}$ as exhibited in Fig. 5.1. To delineate the boundaries of moving objects from the pinpointed moving points, we employ the classic clustering algorithm, DBSCAN [243]. This groups analogous points into object clusters, represented as $\mathbf{D}^t = \{\mathbf{d}_k^t\}_{k=1}^{K^t}$ and we term $\mathbf{D}^t$ as the detected (moving) objects hereafter. For robust clustering, we utilize the point cloud $\mathbf{P}_t$, their estimated scene flow $\mathbf{S}^t$ and the flow embedding $\mathbf{E}^t$ as the salient features and identify neighbour points. In this way, each detected object $\mathbf{d}_k^t$ is represented as a cluster containing a subset of points $\{\mathbf{p}_{k,j}^t\}_j$ with their corresponding scene flow and flow embedding vectors $\{[\mathbf{s}_{k,j}^t, \mathbf{e}_{k,j}^t]\}_j$. It is worth noting that we forego estimating explicit object categories (e.g., car, pedestrian). For the purposes of object tracking, such categorization is *not* imperative. Instead, we identify them simply as *class-agnostic* entities.

5.1.5 Data Association Module

Given the objects $\mathbf{D}^t$ detected in the current frame, our data association module endeavors to align them with the previously tracked objects $\mathbf{O}^{t-1}$ from frame $t-1$. For end-to-end learning and inspired by [73, 80, 200], we opt for the MLP over traditional hand-crafted metrics to derive the affinity matrix $\mathbf{A}^t \in \mathbb{R}^{K^t \times M^t}$ for bipartite matching. Such a learnable distance metric can automatically adjust the weights of different features when calculating the similarity scores. For each pair of clusters $\{\mathbf{d}_k^t, \mathbf{o}_m^{t-1}\}$, its corresponding similarity score $a_{k,m}^t$ can be computed as follows:

$$a_{k,m}^t = MLP(\mathbf{l}_k^t - \mathbf{l}_m^{t-1}) \tag{5.1}$$

where $\mathbf{l}_k^t$, $\mathbf{l}_m^{t-1}$ are the aggregated features of two clusters respectively. Taking the object $\mathbf{d}_k^t$ as the example, to generate its aggregated features $\mathbf{l}_k^t$, we concatenate a) the average and variance of its associated point subset $\{\mathbf{p}_{k,j}^t\}_j$, b) the max-pooling of the point-level scene flow and embedding vectors $\{[\mathbf{s}_{k,j}^t, \mathbf{e}_{k,j}^t]\}_j$. This process can effectively aggregate the information for each cluster and ensure the dimension consistency given clusters with various numbers of points.

Once the affinity matrix $\mathbf{A}^t$ is computed, we identify the optimal matching pairs based on their similarity scores. To address this optimization challenge, we employ the Sinkhorn algorithm [209], as shown in Fig. 5.1. This method involves iterative normalization of $\exp(\mathbf{A}^t)$ across both rows and columns, ensuring the entire data association process remains differentiable. Post optimization, we reassign object IDs to the successfully matched pairs, allocate new IDs for newly detected objects, and remove IDs associated with previously tracked objects absent in the current frame. Notably, we adopt the finalized scores of matched pairs as confidence scores for currently detected objects, given the inability of our detection module to provide such scores. Such confidence scores are used for certain metrics, such as AMOTA and AMOTP, which integrate results over all recall values.

5.1.6 End-to-End Training

We leverage labelled samples to end-to-end train our network with a multi-task loss of scene flow estimation, motion segmentation and affinity matrix computation:

$$\mathcal{L} = \alpha_1 \mathcal{L}_{flow} + \alpha_2 \mathcal{L}_{seg} + \alpha_3 \mathcal{L}_{aff} \tag{5.2}$$

where $\alpha_1, \alpha_2, \alpha_3$ are hyperparameters to weigh different loss functions. For scene flow loss, we compute the L_2 distance between the estimated scene flow $\mathbf{S}^t$ and the ground

truth one $\tilde{\mathbf{S}}^t = \{\tilde{\mathbf{s}}_i^t\}_{i=1}^{N^t}$:

$$\mathcal{L}_{flow} = \frac{1}{N^t}\sum_i ||\mathbf{s}_i^t - \tilde{\mathbf{s}}_i^t||_2^2 \tag{5.3}$$

Given the ground truth motion segmentation mask $\tilde{\mathbf{M}}^t = \{\tilde{m}_i^t\}_{i=1}^{N^t}$, we separately supervise the classification scores of real moving and static points using the cross-entropy to address the low ratio ($< 10\%$) of moving points in point clouds. The motion segmentation loss can be written as:

$$\mathcal{L}_{seg} = \beta\frac{\sum_i(1-\tilde{m}_i^t)\log(1-c_i^t)}{\sum_i 1-\tilde{m}_i^t} + (1-\beta)\frac{\sum_i \tilde{m}_i^t\log(c_i^t)}{\sum_i \tilde{m}_i^t} \tag{5.4}$$

where β is used to balance the influence of moving and static points. To supervise the computation of the affinity matrix $\mathbf{A}^t$, we formulate the prediction of the similarity score as a binary classification (matched or unmatched) problem and compute the binary cross-entropy loss as:

$$\mathcal{L}_{aff} = \frac{1}{K^t M^t}\sum_k\sum_m \tilde{a}_{k,m}^t\log(a_{k,m}^t) + (1-\tilde{a}_{k,m}^t)\log(1-a_{k,m}^t) \tag{5.5}$$

where $\tilde{a}_{k,m}^t \in \{0,1\}$ is the ground truth affinity score for object pair $\{\mathbf{d}_k^t, \mathbf{o}_m^{t-1}\}$. To see our ground truth label generation process, please refer to Sec. 5.2.2.

Note that the above tasks are inherently intertwined and simultaneously optimized via end-to-end training. Supervising both scene flow estimation and motion segmentation directly aids the computation of the affinity matrix, which utilizes scene flow and clusters of moving points as input. Conversely, the gradients originating from the affinity matrix loss instruct the backbone to encode potent features from point clouds. This indirect guidance subsequently enhances both motion segmentation and scene flow estimation.

5.2 Experiments

5.2.1 Evaluation Settings

Dataset. In our experiments, we demonstrate the effectiveness of `RaTrack` using the View-of-Delft (VoD) dataset [81], which includes essential components (i.e., 4D radar point clouds, odometry information, object bounding boxes and tracking IDs annotations) for our problem. As an official benchmark specific for 3D object detection, the annotations of its test split are not publicly available, thereby we evaluate our trained models with its validation split, which is unseen during our training process.

Evaluation metrics. To quantify our performance, we use the classical MOTA, MODA, MT, ML metrics [244, 245] and the popular sAMOTA, AMOTA, AMOTP metrics [71] for evaluation. To make these metrics adapt to our cluster-based object detections, we compute the IoU by counting the number of intersected and united radar points between the ground truth object and the predicted one. The threshold for our point-based IoU is set as 0.25 across all experiments.

Baselines. As there are no prior works for 4D radar-based moving object tracking, we select two state-of-the-art LiDAR-oriented 3D MOT methods, i.e., AB3DMOT [71] and CenterPoint [72] as our baselines. To ensure the comparison is fair, we keep their original settings and also train their models on the VoD training split. Note that baseline methods, though designed for LiDAR, also take 4D radar point clouds as input in this chapter. Specifically, we develop two augmented baselines AB3DMOT-PP and CenterPoint-PP by replacing the detector with PointPillars [246] for AB3DMOT and the backbone with that of PointPillars for CenterPoint.

5.2.2 Implementation Details

Label generation. We follow [61, 125, 184, 247] to generate *pseudo* scene flow labels using the ego-motion and object annotations. The ground truth motion segmentation mask can then be obtained by thresholding after compensating the ego-motion from the scene flow. To get the ground truth affinity matrix, we first match ground truth objects with our detected objects. A detected object that has a point-based IoU higher than 0.25 with any ground truth object will be assigned the same ID as the ground truth. Then, the ground truth affinity matrix can be constructed by assigning $\tilde{a}^t_{k,m} = 1$ if the object pair $\{\mathbf{d}^t_k, \mathbf{o}^{t-1}_m\}$ has the same ID, and verse vice.

Hyperparameters. Our hyperparameters are determined both empirically and according to the testing results. The threshold ζ_{mov} (c.f. Sec. 5.1.4) is simply set as 0.5. For the DBSCAN algorithm, the neighbourhood radius is 1.5m while the minimum number of points in a cluster is set as 2. For loss weights, we set $\alpha_1, \alpha_2, \alpha_3$ in the overall loss (c.f. Eq. 5.2) to be 0.5, 0.5 and 1.0 respectively, and set β in the motion segmentation loss (c.f. Eq. 5.4) to be 0.4.

Network training. Training our network is non-trivial due to the dependencies of intermediate outputs across modules and frames. For example, the efficacy of our data association module depends on whether moving objects are correctly detected in the object detection module. To more effectively train our network, we separate the training

Method	sAMOTA [%] ↑	AMOTA [%] ↑	AMOTP [%] ↑	MOTA [%] ↑	MODA [%] ↑	MT [%] ↑	ML [%] ↓
CenterPoint [72] (A)	37.72	8.77	38.92	33.34	35.17	11.73	57.41
CenterPoint-PP [72, 246] (A)	42.64	10.87	43.64	36.20	37.29	15.43	50.62
AB3DMOT [71] (A)	33.56	6.66	33.34	31.00	31.20	14.81	64.20
AB3DMOT-PP [71, 246] (A)	35.82	7.67	36.70	38.44	41.96	19.12	38.24
CenterPoint [72]	43.21	14.40	54.55	38.44	41.96	19.12	38.24
CenterPoint-PP [72, 246]	44.54	16.33	58.80	43.96	44.91	19.12	54.41
AB3DMOT [71]	51.23	15.00	53.21	46.72	47.38	20.59	39.71
AB3DMOT-PP [71, 246]	60.71	21.51	**62.75**	49.38	49.86	26.47	33.82
RaTrack	**74.16**	**31.50**	60.17	**67.27**	**77.83**	**42.65**	**14.71**

Table 5.1: Performance of RaTrack and baselines on VoD. Baselines with (A) represent methods trained and evaluated on all objects, while others are trained and evaluated only on moving objects, which serve as the main comparison for RaTrack.

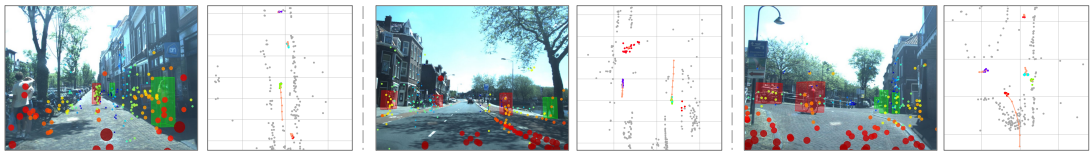


Figure 5.2: Qualitative results of `RaTrack`. We show on RGB images the ground truth 3D bounding boxes of moving objects (colors indicate different object classes) and projected radar point clouds (colors indicate the distances of points). On corresponding bird's eyes view figures, we show the detected moving objects and their predicted trajectories, where different colors are used to distinguish multiple object clusters.

into two stages: 1) We first train the backbone and the class predictor used for motion segmentation with Eq. 5.4 and keep other components frozen. The training keeps for 16 epochs with an initial learning rate of 0.001. This allows for fast learning of accurate moving object detection before data association. 2) We then train the whole network end-to-end for an additional 8 epochs with an initial learning rate of 0.0008. The Adam optimizer [248] is used for network parameter updates and the learning rate decays by 0.97 per epoch in both stages.

5.2.3 Overall Performance

We evaluate `RaTrack` and our baseline methods and compare their results on different metrics, as shown in Table 5.1. `RaTrack` exhibits a remarkable gain over baselines, increasing the sAMOTA, AMOTA and MOTA by 13.4%, 10.0%, 17.9% compared to the second-best scores. Such results demonstrate its superiority in 4D radar-based

Method	sAMOTA [%] ↑	AMOTA [%] ↑	AMOTP [%] ↑	MOTA [%] ↑	MODA [%] ↑	MT [%] ↑	ML [%] ↓
Replace ODM with PP [246]	53.12	17.47	53.65	41.66	42.32	24.21	38.95
Remove MEM	68.45	26.08	51.23	62.32	71.27	38.24	16.18
Remove the velocity	31.50	5.63	16.83	24.17	32.45	11.76	30.88
RaTrack	**74.16**	**31.50**	**60.17**	**67.27**	**77.83**	**42.65**	**14.71**

Table 5.2: Ablation study results for RaTrack. For the first row, the bounding box detection results from PointPillars [246] are employed to generate object clusters as input to data association.

moving object detection and tracking. Notably, `RaTrack` achieves an improvement of 28.0% on the MODA metric that is used to measure the object detection accuracy. This supports that our class-agnostic object detection without bounding boxes is a better option than current 3D bounding box detectors [72, 195, 246] for recognizing and localizing moving objects in 4D radar point clouds. With more reliable object detection and data association module, `RaTrack` also surpasses all baselines on the MT and ML metric, which demonstrates its ability to maintain long-term tracking of moving objects. It can be also observed that `RaTrack` has a slightly lower AMOTP than the AB3DMOT-PP baseline [71, 246]. As a precision metric, AMOTP only calculates the IoU between successfully matched object pairs, thus it becomes less important when the MOTA and MODA, which assess the tracking and detection accuracy, are considerably lower. We also show some qualitative results of `RaTrack` in Fig. 5.2.

As seen in Table 5.1, we also train and evaluate baseline models with all annotated objects rather than only the moving ones to compare their performance in two settings. A consistent performance gap can be observed between the two settings of our baselines, which demonstrates that detecting and tracking moving objects is easier than that of general objects. Moreover, we find that our two augmented baselines (i.e., with -PP) outperform the original ones. We credit this to PointPillars [246] which is the state-of-the-art for 4D radar-based 3D object detection as exhibited in [81].

Runtime efficiency. To assess the practical deployment potential of our `RaTrack` framework, we evaluate its runtime performance on a single NVIDIA RTX 3090 GPU. The full inference pipeline runs at $\sim$11 Hz (91ms per frame) while maintaining a low GPU memory consumption. This demonstrates that `RaTrack` achieves real-time performance, making it suitable for deployment in mobile autonomous systems that typically operate at low sensor frame rates (e.g., 10 Hz for 4D automotive radars).

5.2.4 Ablation Study

To validate the effectiveness of the object detection module (ODM) (c.f. Sec. 5.1.4) and motion estimation module (MEM) (c.f. Sec. 5.1.3), here we conduct an ablation study to see their impact. We also analyze the impact of the auxiliary velocity features. The results are shown in Table 5.2.

Object detection module. By replacing our ODM with PointPillars detector [246], the performance on sAMOTA, MOTA and MODA degrades by 21.0%, 25.6% and 35.5% respectively. This supports again that our cluster-based object detection is more effective than previous detectors for moving object detection and can thus yield more accurate tracking results. However, the outcome of this ablated version is less optimal than anticipated. We credit this to the incompatibility of the bounding box detector with our framework. Specifically, the object clusters derived from bounding boxes might have more noisy points, which further interferes with our data association that utilizes point-level features.

Motion estimation module. The removal of our MEM yields the decrease of 5.7%, 5.4%, 6.6% on sAMOTA, AMOTA and MODA. Such a change in performance demonstrates that our estimated scene flow from MEM can facilitate robust object detection and benefit object temporal matching. Specifically, without MEM, our AMOTP drops by 8.9% which is a non-trivial change in the detection precision. This highlights the importance of our scene flow estimation as an additional motion cue in clustering. Points with similar scene flow vectors are prone to be grouped together.

Auxiliary velocity feature. By incorporating the velocity information into the input, a substantial gain is witnessed in all metrics, which demonstrates that the input velocity features are indispensable for `RaTrack`. Indeed, the velocity features are the key enabler to scene flow estimation and motion segmentation tasks in our network by providing point-level motion information to be encoded in the backbone features. On the other hand, `RaTrack` can sufficiently utilize such features with its bespoken network designed for 4D radar.

5.2.5 Sensitivity Analysis

Impact of valid object threshold. During the evaluation, we ignore invalid objects (< 5 points) from both predictions and ground truth as we are more interested in the objects that are *sufficiently* measured. Here we investigate the impact of this valid object threshold (i.e., the minimum number of points to identify an object as valid) on our

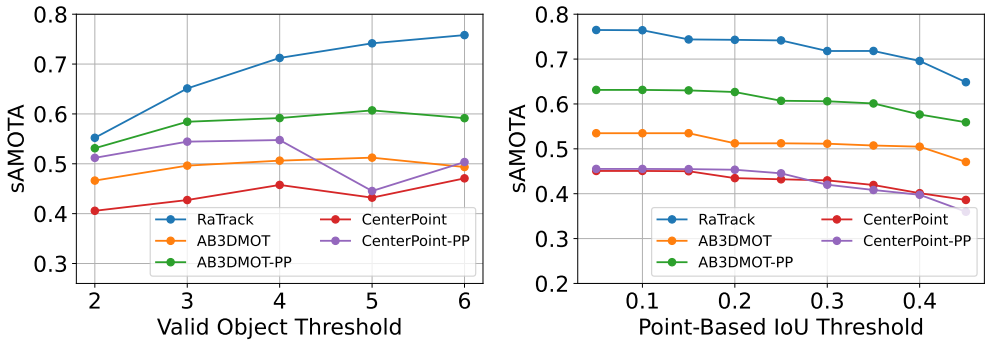


Figure 5.3: Result comparison between `RaTrack` and baselines on varying valid object thresholds (both predicted and ground truth objects are filtered) and point-based IoU thresholds.

evaluation results. As exhibited in Fig. 5.3 (left), `RaTrack` consistently outperforms all baseline methods, regardless of the valid object threshold enlarges, and continuously increase its scores as the threshold. We credited this to our cluster-based object detection that can group any number of neighbour points into objects for tracking, where objects with more points are more easily being recognized and tracked. In contrast, it is hard for our baseline methods to produce comparable results with 3D bounding box-based detection strategy.

Impact of point-based IoU threshold. We also analyze the impact of the point-based IoU threshold (i.e., the minimum IoU to be identified as a true positive sample) for our evaluation results. As seen in Fig. 5.3 (right), `RaTrack` achieves the best results on all IoU thresholds with a consistent improvement of $\sim 10\%$ over baselines, which further confirms the superiority of our pipeline.

5.3 Conclusion

In this chapter, we unveiled the untapped potential of 4D mmWave radars for multiple moving object tracking. Addressing the challenges of noise and point sparsity in radar data, our approach, `RaFlow`, offers a fresh perspective on the tracking of moving objects, emphasizing the utility of motion segmentation and clustering over the conventional dependence on specific object types and bounding boxes. This restructured approach not only simplifies the process but also boosts the accuracy of multi-object tracking in complex dynamic scenarios. Extensive evaluations on a public dataset highlight the method's competitive performance.

5.4 Limitation and Future Works

RaTrack enables reliable 4D radar-based detection and tracking of moving agents in dynamic environments, offering a fresh perspective on this problem. However, some limitations remain in the current method, and future work is expected to address them.

Limited Training Data. First, its performance is limited by the small scale of available training data provided by [81]. To improve performance, future work could incorporate larger annotated 4D radar data from newly released sources [136, 249] or leverage self-supervised learning [250] to utilize unannotated 4D radar data.

Sensitivity to Sparse Objects. Besides, RaTrack primarily focuses on objects with sufficient radar reflections, which are represented by a larger number of points. Consequently, the method exhibits performance degradation when applied to small or distant objects with fewer radar reflections. This focus also guides our evaluation protocol, where we consider objects with at least five points to align with the method's strengths. Future research could address this limitation by adopting learning-based clustering methods for adaptive point aggregation, replacing traditional methods like DBSCAN, which rely on fixed hyperparameters. Unlike DBSCAN, which struggles to adapt to varying point densities, learning-based approaches can dynamically adjust clustering behavior based on the spatial and semantic features of the data. This enables more effective grouping of points, even for sparse, isolated objects with weak radar reflections.

Lack of Compatible Output for Planning. Moreover, RaTrack outputs class-agnostic objects represented as clusters of points, which, while sufficient for moving object tracking, are incompatible with the downstream planning modules [9] that require bounding box representations as input. Although planning modules can be made more flexible in their input representations, an alternative approach from the perception side is to transform point clusters into bounding boxes. One potential method involves aggregating points from clusters corresponding to the same object across multiple frames to capture more complete spatial information. This aggregated data can then be used to learn probabilistic bounding box regression models, which predict bounding box properties with greater accuracy and robustness.

Birth and Death Memory. Lastly, RaTrack does not explicitly handle cases where objects are temporarily occluded or disappear from the scene, nor does it filter out transient false positives from newly detected objects. In contrast, methods like AB3DMOT [71] buffer unmatched detections and tracks for several frames to reduce false positives and avoid premature deletion. Incorporating a similar memory-based birth/death strategy

could improve robustness under occlusion and enhance tracking stability in future work.

Chapter 6

Robust 3D Occupancy Prediction with 4D Imaging Radar

In Chapter 5, we introduced a 4D radar-based tracking method for object-level perception, enabling reliable detection and tracking of moving agents. While object tracking captures motion dynamics, 3D occupancy perception has gained traction due to its comprehensive scene representation. Unlike object detection, which focuses on foreground entities, 3D occupancy maps the entire environment, including irregular and open-set objects, making it particularly effective for handling corner cases. Despite these advantages, 4D radar has not been explored for 3D occupancy prediction. To address this gap, this chapter presents `RadarOcc`, a novel framework that directly processes 4D radar tensors (4DRT) instead of relying on sparse radar point clouds. By preserving the full radar signal, `RadarOcc` retains crucial environmental details often lost during point cloud generation. To tackle the challenges of data sparsity, noise, and large-scale processing, `RadarOcc` incorporates Doppler bin descriptors, sidelobe-aware spatial sparsification, and range-wise self-attention mechanisms. Additionally, we introduce a spherical-based feature encoding and a spherical-to-Cartesian feature aggregation strategy to minimize interpolation errors during coordinate transformations.

The remainder of this chapter is organized as follows. Sec.6.1 explains the rationale of using 4DRT for 3D occupancy prediction, details the proposed `RadarOcc` framework and its core components. Sec.6.2 presents experimental evaluations, followed by conclusions in Sec.6.3. Sec. 6.4 discusses the limitations and potential future works.

Statement of Contribution. This chapter is based on the following publication: "RadarOcc: Robust 3D Occupancy Prediction with 4D Imaging Radar," Fangqiang Ding, Xiangyu Wen, Yunzhou Zhu, Yiming Li, Chris Xiaoxuan Lu, *The Thirty-Eighth*

Annual Conference on Neural Information Processing Systems (NeurIPS), 2024. As one of the leading authors, I proposed the core idea of this work, designed the framework, helped implement the labeling and visualization, conducted part of the experiments, and wrote the paper. I was also responsible for obtaining the dataset from the host organization. Another leading author, Xiangyu Wen, significantly contributed to idea development and framework design. He also implemented our proposed method and baselines and conducted most of the experiments. Yunzhou Zhu contributed to the evaluation of some baseline methods, performed visualization, and created video demos. Chris Xiaoxuan Lu helped polish the manuscript and provided valuable feedback in revising the paper.

6.1 Method

6.1.1 4DRT for 3D Occupancy Prediction

As a reminder, we refer readers to Section 1.2 for an overview of the 4D radar signal processing, which explains how to obtain different 4D radar data representations.

Rationale of using 4DRT. 4D radar tensors (4DRTs) serve as raw sensor data that amalgamate the strengths of LiDAR/radar point clouds and RGB images, providing direct 3D measurements in a continuous data format. These tensors comprehensively capture information from raw radar measurements, effectively addressing the shortcomings associated with the sparseness of radar point clouds caused by the signal post-processing. For instance, low-reflectivity surfaces like asphalt, common on highways, typically do not reflect enough radar signals for detection. By using 4DRTs, these minimal signal returns can be detected, significantly bolstering occupancy prediction capabilities. Furthermore, the volumetric structure of 4DRTs aligns well with 3D occupancy grids, making them ideally suited for advancing 3D occupancy prediction techniques.

Challenges. Despite their significant advantages, using 4D radar tensors (4DRTs) for 3D occupancy prediction presents substantial challenges. First, the large data size of 4DRTs (e.g., 500MB per frame in the K-Radar dataset [52]) hinders computational efficiency, necessitating data volume reduction before processing. Second, the inherent noise in radar data, exacerbated by the multi-path effect of mmWave, requires careful filtering to preserve essential signals while eliminating noise. Third, the discrepancy between the spherical coordinates of 4DRT data and the Cartesian coordinates required for 3D occupancy outputs calls for a tailored network design. This design must effectively

translate spatial interactions from spherical to Cartesian dimensions to ensure accurate occupancy predictions.

6.1.2 Overview

In this chapter, we consider the task of 3D occupancy prediction with single-frame 4DRT output from 4D imaging radar. The problem definition can be seen in Sec. 1.4. `RadarOcc` consists of four components in tandem (*c.f.* Fig. 6.1). Before loading heavy 4DRTs to the neural network, we reduce their data volume as the preprocessing steps via encoding the Doppler bins descriptor and performing sidelobe-aware spatial sparsifying to improve the efficiency without losing the key information (*c.f.* Sec. 6.1.3). To refrain from the interpolation error, we encode spatial features directly on the spherical RTs without transforming them into Cartesian volumes (*c.f.* Sec. 6.1.4) and aggregate the spherical features with 3D volume queries defined in the Cartesian coordinates (*c.f.* Sec. 6.1.5). Specifically, range-wise self-attention is used to alleviate the sidelobes, and sparse convolution and deformable attention are leveraged for fast feature encoding and aggregation. The occupancy probabilities are predicted in the 3D occupancy decoding step, which is supervised via our training loss (*c.f.* Sec. 6.1.6).

6.1.3 Data Volume Reduction

Direct processing of raw 4DRTs with neural networks is impractical due to its substantial data size (*e.g.*, 500MB per frame) which leads to heavy computation cost and memory usage. Moreover, the slow data transfer between the sensor, storage device and processing unit (CPU/GPU) of large-volume raw 4DRTs not only hinders the onboard runtime efficiency but also increases the training duration which demands repetitive data loading. For efficiency, we propose to reduce the data volume of 4DRTs through encoding the Doppler bins descriptor and sidelobe-aware spatial sparsifying as the preprocessing steps (see Fig. 6.1). Post reduction, the loading of 4DRTs into the processing unit for runtime inference can be more feasible and the network training can be more efficient.

Doppler bins descriptor. Unlike the three spatial axes, which are intuitively critical for spatial perception, the Doppler axis in 4DRTs has often been considered redundant in 3D object detection. Previous studies [52, 53, 120] have employed average-pooling to minimize this axis, aiming to reduce computational overhead. However, we argue that this ostensibly 'redundant' axis contains vital cues for geometric and semantic

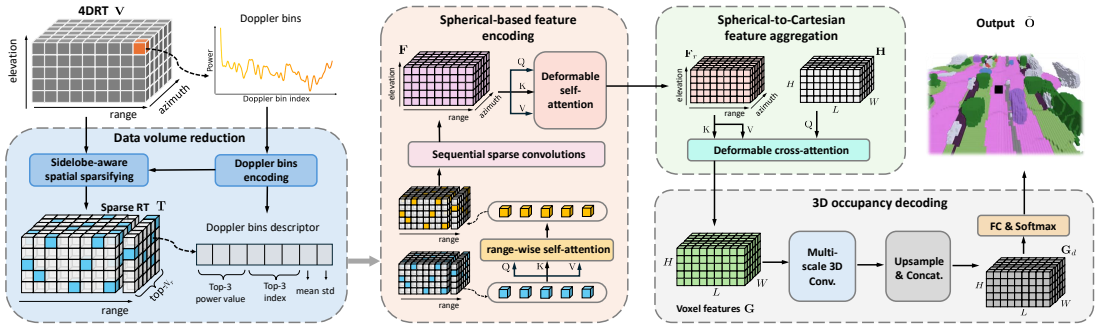


Figure 6.1: Overall pipeline of `RadarOcc`. The data volume reduction pre-processes the 4DRT into a lightweight sparse RT via Doppler bins encoding and sidelobe-aware spatial sparifying. We apply spherical-based feature encoding on the sparse RT and aggregate the spherical features using Cartesian voxel queries. The 3D occupancy volume is finally output via 3D occupancy decoding.

analysis in 3D occupancy prediction. Specifically, the Doppler axis provides essential information on object speed via peak locations, aiding in differentiating dynamic objects from static backgrounds. Moreover, the power distribution within the Doppler bins offers insights into the confidence levels of true targets—essentially, indicating their likelihood of occupancy. To preserve and utilize this crucial information, we have developed a method to encode the Doppler bins into a descriptor that captures specific statistics for each spatial location within the 4DRTs. This descriptor incorporates the top-three power values along with their indices, the mean power value, and the standard deviation, as depicted in Fig 6.1. Note that the number of preserved top values is determined empirically. Consequently, this approach enables us to reduce the data volume of raw 4DRTs by a factor of $\frac{D}{8}$, while retaining key information from the Doppler axis.

Sidelobe-aware spatial sparsifying. By encoding the Doppler bins into light-weight descriptors, we transform the raw 4DRT into 3D spatial data volume with the original Doppler axis as the 8-channel feature dimension. Nevertheless, it remains costly for neural networks to encode features from 3D dense data volume with operations like 3D convolution [251, 252]. To accelerate the computation, prior arts [52, 53] transfer the dense RT into a sparse format by retraining only the top-percentile elements based on power measurements. However, this approach tends to be biased towards specific ranges that exhibit exceptionally high measurements. It can be observed in Fig. 6.2 that after percentile-based sparsifying, a significant number of the reserved elements are concentrated within the same ranges spread across the azimuth and elevation axes.

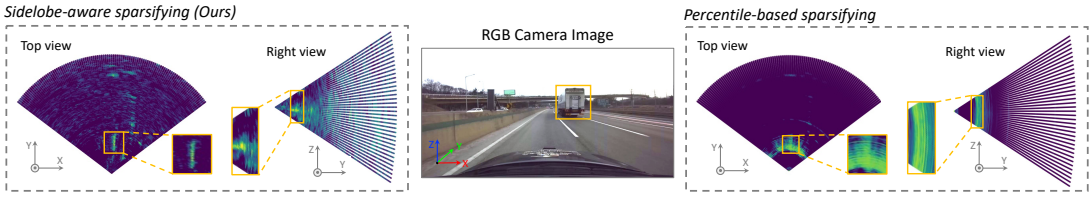


Figure 6.2: Comparison between the sparse RTs resulted by our sidelobe-aware and percentile-based sparsifying [52, 53]. We transform the spherical RT elements to the Cartesian coordinates and show them in two views. The arches on the heatmap indicate the same ranges. Percentile-based method retains many elements caused by sidelobe noise, which are concentrated at certain ranges. In contrast, our method can reduce the sidelobe level and reserve critical measurement from different ranges.

These elements manifest as artifacts of sidelobes, which can an be viewed as the diffraction pattern of the antenna [253, 254]. Consequently, this results in the loss of important measurements from other ranges and introduces lots of noise into the sparse tensor. To mitigate this issue, we propose to select the top-N_r elements for each individual range instead of on the whole dense RT for spatial sparsifying (see Fig. 6.1). In this way, the dominance of certain ranges can be avoided while the sidelobe level is reduced, as exhibited in Fig. 6.2. Note that our spatial element selection is based on the mean power value across the Doppler axis. The final sparse tensor is denoted as $\mathbf{T} = \{t_i \in \mathbb{R}^{N_r \times (8+2)}\}_{i=1}^{R}$ with the extra two feature channels storing the azimuth and elevation indices of reserved N_r elements for each range.

6.1.4 Spherical-based Feature Encoding

Given the sparse RT, we aim to encode representative features for accurate 3D occupancy prediction. As the sparse RTs are inherently in the spherical coordinates, previous works [52, 53] transfer them into the Cartesian coordinates before feature encoding. However, such a transfer would undermine their uniform density distribution and often incur interpolation errors. Inspired by the polar representation of point clouds [255–257], we propose to take the elements in RT as voxels rasterized in the spherical coordinates and apply the spherical-based feature encoding directly. The spherical voxel representation naturally matches the spherical-uniform distribution of RTs and can refrain from inducing interpolation errors. In practice, the 3D convolutions can be used to extract grid-based representations by only replacing the X-Y-Z axis with the range-azimuth-elevation axis. In what follows, we illustrate our spherical-based feature

encoding process.

Range-wise self-attention. In Section 6.1.3, we address the issue of sidelobes by selecting elements based on range-wise percentiles during the preprocessing phase. To further mitigate sidelobe interference, we introduce a range-wise self-attention mechanism [258] (see Fig. 6.1) as the initial step in our feature encoding process. Specifically, within each range component $t_i \in \mathbf{T}$, which includes N_r RT tokens, we utilize the Doppler bin descriptors as token features. Additionally, two index channels are employed for positional embeddings to enhance the specificity of our spatial encoding.

Sequential sparse convolution. For efficiency, we apply a series of 3D sparse convolutions [259] onto the sparse RT for spatial feature encoding in the spherical voxel space. This produces a 3D dense feature volume $\mathbf{F} \in \mathbb{R}^{\frac{R}{S} \times \frac{A}{S} \times \frac{E}{S} \times C_f} (N_f = \frac{R}{S} \times \frac{A}{S} \times \frac{E}{S})$ with a reduce spatial dimension characterized by the stride S, where C_f denotes the feature dimension. Note that $\mathbf{F}$ inherently aligns with the spherical space with each feature element's indices corresponding to a spherical coordinate.

Deformable self-attention. Following the consecutive 3D sparse convolution, we use the 3D deformable attention [260] to further refine and augment our feature volume $\mathbf{F}$ by enforcing spatial interaction. As a definition, for a query feature z corresponding to a reference point p in the input feature $\mathbf{X}$, its feature can be updated by deformable attention in the following equation:

$$\mathrm{DeformAttn}(z, p, \mathbf{X}) = \sum_{m=1}^{M} \mathbf{W}_m \left[\sum_{k=1}^{K} \mathbf{A}_{mk} \cdot \mathbf{W}'_m \mathbf{X}(p + \Delta p_{mk}) \right] \tag{6.1}$$

where $\mathbf{W}_m$ and $\mathbf{W}'_m$ are the learnable weights for the m-th attention head, while $\mathbf{A}_{mk}$ and Δp_{mk} is the attention weight and sampling offset calculated with z for its k-th sampling point and the m-th head. $\mathbf{X}(p + \Delta p_{mk})$ is the key features at the sample location $(p + \Delta p_{mk})$. By applying self-attention to $\mathbf{F} = \{f^q\}_{q=1}^{N_f}$, the refined feature volume $\mathbf{F}_r = \{f_r^q\}_{q=1}^{N_f}$ can be derived by:

$$f_r^q = \mathrm{DeformAttn}(f^q, p^q, \mathbf{F}) \tag{6.2}$$

6.1.5 Spherical-to-Cartesian Feature Aggregation

Decoding 3D Cartesian occupancy from a spherical feature volume is inherently challenging due to misalignments in spatial axes and discrepancies in the regions they represent. An intuitive approach would be to transform the spherical feature volume

into a Cartesian one and then decode the 3D Cartesian occupancy. However, this method can introduce feature-level interpolation errors, which we aim to avoid as discussed in Section 6.1.4.

To avoid conducting interpolation, we propose to aggregate the spherical features in a *learnable* way, with 3D volume queries defined in the Cartesian coordinates attending to the feature samples in $\mathbf{F}_r$, as shown in Fig. 6.1. First, we build learnable grid-based voxel queries $\mathbf{H} = \{h^q \in \mathbb{R}^{C_f}\}_{q=1}^{H \times W \times L}$ which has the same volumetric size as our desired output $\mathbf{O}$ and the same feature dimension as the spherical feature volume $\mathbf{F}_r$. Each voxel query h^q corresponds to a 3D point p^q in the Cartesian coordinate. Second, the 3D point p^q of each query is transformed from the Cartesian to the spherical coordinate, which is then mapped to a index position in $\mathbf{F}_r$ denoted as $\Phi(p^q)$. We take $\Phi(p^q)$ as a 3D reference point in the spherical space and sample key elements in its vicinity from the feature volume $\mathbf{F}_r$. Lastly, we leverage deformable cross-attention [260] to aggregate the key samples for each reference point and the output $\mathbf{G} = \{g^q \in \mathbb{R}^{C_f}\}_{q=1}^{H \times W \times L}$ can be calculated by:

$$g^q = \mathrm{DeformAttn}(h^q, \Phi(p^q), \mathbf{F}_r) \tag{6.3}$$

6.1.6 3D Occupancy Decoding and Supervision

With the aggregated voxel features $\mathbf{G}$, we leverage consecutive 3D convolutions [251, 252] with skip connection [14] to decode hierarchical feature volumes at N_s scales with a scaling step of 2. Multi-scale feature volumes are then merged in a top-down way [261] via upsampling features by a factor 2 and concatenated along the feature dimension, resulting in $\mathbf{G}_d \in \mathbb{R}^{H \times W \times L \times N_s C_f}$. Finally, the occupancy head equipped with the *softmax* function is employed to output the normalized occupancy probabilities $\tilde{\mathbf{O}} \in \{0, 1\}^{H \times W \times L \times (C+1)}$ for all voxels on C semantic classes and one free class.

Our network is trained in a supervised way with the ground truth occupancy. Following [85], we use the cross-entropy loss as the primary loss to optimize the training and incorporate the lovasz-softmax loss [262] to handle the class imbalances. Moreover, we utilize the scene- and class-wise affinity loss proposed in [96] to enhance the optimization of geometry and semantic IoU metrics.

6.2 Experiment

6.2.1 Experimental Setup

Dataset preparation. Our experiments are conducted on the K-Radar dataset [52], which is, to the best of our knowledge, the only autonomous driving dataset providing available 4DRT data. Besides, K-Radar also contains multi-modal data from LiDAR, camera, GPS-RTK and annotated 3D bounding boxes and tracking IDs, enabling us to compare between different modalities and generate 3D occupancy labels. Following [85, 98, 263], we generate occupancy ground truth by superimposing consecutive LiDAR sweeps and construct the dense 3D occupancy grids via voxelization. To handle scene dynamics, we register objects with the same tracking IDs across the sequence. Please refer to the Appendix B.1 for more details. As K-Radar does not annotate fine-grained point-level semantics, we segment the scene into the foreground (*e.g.,* sedan, truck, pedestrian) and background using bounding boxes and label the voxel grids into three classes, including foreground, background and free. Many sequences in K-Radar were collected under adverse weather (*i.e.,* sleet, rain, and snow), which results in non-negligible noise to the generated occupancy labels based on LiDAR sweeps. Therefore, we reserve this adverse-weather test split for qualitative comparison and only generate the occupancy labels for the well-condition sequences, which are separated into the training, validation and test splits.

Evaluation protocol. As the pioneering study of 3D occupancy prediction using the K-Radar dataset, we have tailored the evaluation protocol to align with our experimental needs. We define the Region of Interest (RoI) with specific dimensions: a front range of [0, 51.2m], a side range of [-25.6m, 25.6m], and a height range of [-2.6m, 3m]. The voxel resolution is set at 0.4m, resulting in a target occupancy volume of $128 \times 128 \times 14$ voxels. Consistent with established methods in the field [85, 224, 263], we employ the Intersection over Union (IoU) metric to evaluate the geometric accuracy of our occupancy predictions, focusing solely on the occupied or free status without integrating semantics. Additionally, to gauge the effectiveness of our foreground-background segmentation, we calculate the mean IoU (mIoU) across these two classes. In line with previous studies [87, 263], we present our findings across multiple ranges, specifically at 51.2m, 25.6m, and 12.8m.

Competing methods. We benchmark `RadarOcc` against state-of-the-art methods employing different modalities. Given that recent studies do not use radar data for 3D occupancy prediction, we adapt the OpenOccupancy LiDAR-based baseline and CONet [85]

		IoU (%)			mIoU (%)			BG IoU (%)			FG IoU (%)		
Method	Input	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m
L-baseline [85]	RPC	42.8	34.9	27.9	23.5	18.6	14.6	43.5	34.6	27.3	3.5	2.6	1.9
L-CONet [85]	RPC	46.1	36.0	25.0	24.6	20.3	14.4	43.3	35.4	25.6	5.8	5.2	3.1
L-baseline [85]	4DRT-XYZ	47.4	38.1	28.5	29.9	24.3	17.5	46.4	37.5	27.9	13.4	11.1	7.2
RadarOcc (Ours)	4DRT	**48.8**	**39.1**	**30.4**	**34.3**	**28.5**	**22.6**	**47.9**	**38.2**	**29.4**	**20.7**	**18.7**	**15.8**

Table 6.1: Quantitative comparison between `RadarOcc` and state-of-the-art radar-based baseline methods. Results are reported on K-Radar well-condition test split. Best result is shown in **bold**.

		IoU (%)			mIoU (%)			BG IoU (%)			FG IoU (%)		
	Method	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m
(a)	Ours	**48.8**	39.1	**30.4**	**34.3**	28.5	**22.6**	**47.9**	38.2	**29.4**	**20.7**	18.7	15.8
(b)	Ours w/o DBD	48.1	**39.4**	30.0	33.6	**28.9**	22.6	47.2	**38.7**	29.2	20.0	**19.1**	**16.0**
(c)	Ours w/o SSS	44.2	36.8	28.7	24.1	20.2	15.6	42.3	35.6	27.6	5.9	4.7	3.5
(d)	Ours w/o SFE	46.2	38.4	29.4	30.4	26.5	21.1	45.5	37.5	28.5	15.4	15.5	13.9

Table 6.2: Ablation studies on key designs of `RadarOcc`. DBD, SSS, SFE refer to the Doppler bins descriptor, sidelobe-aware spatial sparfiying, and spherical-based feature encoding, respectively.

to accommodate radar point cloud (RPC) inputs for our comparative analysis. Furthermore, we convert 4DRTs to Cartesian coordinates [52] with a voxel size of 0.4m, referred to as 4DRT-XYZ, and integrate them into the LiDAR-based OpenOccupancy framework [85]. Following best practices from [52, 53], we process 4DRT-XYZ into a sparser format. For a comprehensive inter-modality evaluation, we also replicate the OpenOccupancy LiDAR-based baseline [85] and both monocular and stereo camera-based SurroundOcc [98] configurations to fit our experimental setup. Notably, we enrich our comparisons by generating 16-beam and 32-beam LiDAR point clouds from the standard 64-beam configurations through elevation-wise downsampling. The evaluation focuses on the overlap area between the horizontal field of view (FoV) of all sensors and our defined RoI to minimize potential data discrepancies beyond the FoV. For implementation, we train all evaluated models on our K-Radar well-condition training set.

6.2.2 Comparison against Radar-based Methods

We first compare RadarOcc with state-of-the-art baseline methods using radar data for 3D occupancy prediction in Tab. 6.1. As can be seen, RadarOcc outperforms other approaches in every metric, demonstrating its state-of-the-art performance in radar-based 3D occupancy prediction. Specifically, our 4DRT-based RadarOcc largely improves the performance over RPC-based methods: the mIoU of L-CONet [85] is relatively improved by 39.4%, 40.4% and 56.9% for different volumes (12.8m, 25.6m, 51.2m). Such a significant improvement mainly stems from the dense data format of 4DRT, which retains critical information from low-reflectivity objects, enabling effective occupancy prediction for the whole scene. 4DRT-XYZ based L-baseline [85] also outperforms RPC-based methods but inferior to RadarOcc, especially in long-range FG IoU. We credit this to the interpolation errors led to small and far foreground objects when we converting 4DRT to Cartesian coordinates.

6.2.3 Ablation Study

To validate the effectiveness of our key designs, we ablate them alone from our 4DRT-based pipeline RadarOcc and show the evaluation results on K-Radar well-condition test split in Tab. 6.2.

Doppler bins descriptor. By replacing the Doppler bins descriptor with the average-pooling result, the performance of RadarOcc is degraded in most metrics (row (a) vs. (b) in Tab. 6.2), demonstrating the usefulness of preserving the information encoded by the Doppler axis (*c.f.* Sec. 6.1.3). However, the improvement is somehow marginal due to the limited Doppler measurement range of the radar used in K-Radar [52], which wraps around the overflow values, causing ambiguity in Doppler velocity.

Sidelobe-aware spatial sparsifying. We conduct this experiment (row (c) in Tab. 6.2) by changing our sidelobe-aware spatial sparsifying (*c.f.* Sec. 6.1.3) to the percentile-based spatial sparsifying used in [52, 53]. Our sidelobe-aware approach leads to a remarkable advancement in performance, especially in mIoU metrics. This is attributed to its ability to preserve more valid elements from diverse ranges and suppress sidelobes for sparse RTs, allowing for more accurate prediciton.

Spherical-based feature encoding. For row (d) in Tab. 6.2, we transform sparse RT to Cartesian coordinates before feature encoding (*c.f.* Sec. 6.1.4) and omit the spherical-to-Cartesian feature aggregation (*c.f.* Sec. 6.1.5). We can see that our spherical-based feature encoding gains the performance for each metric as our strategy preserves the

Method	range-wise attn.	seq. sparse conv.	deform. self-attn.	deform. cross-attn.	occ. decoding	total runtime	fps
RadarOcc	2.5	47.5	88.8	72.0	92.1	302.9	3.30
RadarOcc (w. optim.)	2.5	20.7 (-56.4%)	32.8 (-63.1%)	29.7 (-58.7%)	48.3 (-47.6%)	133.9 (-55.8%)	7.46 (+126.1%)

Table 6.3: Comparison between `RadarOcc` and its lightweight version after computation optimization in terms of each component's and total runtime (ms) and fps. Relative change is shown in (·).

	IoU (%)			mIoU (%)			BG IoU (%)			FG IoU (%)		
Method	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m
RadarOcc	**48.8**	**39.1**	**30.4**	34.3	**28.5**	**22.6**	**47.9**	**38.2**	**29.4**	20.7	**18.7**	**15.8**
RadarOcc (w. optim.)	46.5	38.0	29.3	**35.5**	27.6	20.9	46.0	37.6	28.8	**25.0**	17.5	13.1

Table 6.4: Comparison between `RadarOcc` and its lightweight version after computation optimization in terms of performance across metrics at different ranges. Better result is shown in **bold**.

original data distribution, avoiding incurring interpolation errors. This also validates the effectiveness of our learnable spherical-to-Cartesian feature aggregation.

6.2.4 Model Efficiency

To assess the runtime efficiency of `RadarOcc`, we conducted our model inference on a single Nvidia GTX 3090 GPU. The results shows an average inference speed of approximately 3.3fps. Although there is still a gap between the real-time application (*i.e.,* 10fps), our inference speed has surpassed that of many camera-based methods as reported in [98]. Further improvements in inference speed can be achieved by reducing network complexity and applying precision reduction techniques, such as converting model precision from *Float32* (FP32) to *Float16* (FP16).

To validate this, we simplified the feature encoding (*c.f.* Sec. 6.1.4) and aggregation (*c.f.* Sec. 6.1.5) modules by reducing some redundancy layer (*e.g.*, number of layers in deformable attention) for efficiency, and converted the computationally intensive 3D occupancy decoding module (*c.f.* Sec. 6.1.6) from FP32 to FP16 via the quantization in PyTorch. These optimizations resulted in a 126% increase in inference speed, reaching approximately 7.46 fps, with only a minimal impact on performance. Please refer to Tab. 6.3 and Tab. 6.4 for detailed changes in runtime for each module and performance. Given the increasing computational power of modern embedded GPUs, such as the Nvidia Jetson Orin, which can almost rival desktop GPUs like the Nvidia

		IoU (%)			mIoU (%)			BG IoU (%)			FG IoU (%)		
Method	Input	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m
L-baseline [85]	L (16)	**49.1**	**43.3**	**35.2**	**39.0**	**34.3**	**28.2**	**48.2**	**42.5**	**34.4**	**29.8**	**26.1**	**22.1**
	L (32)	**51.1**	**44.0**	**34.9**	**42.1**	**35.0**	**28.9**	**50.8**	**43.6**	**34.2**	**33.5**	**26.3**	**23.6**
	L (64)	**56.9**	**52.5**	**43.8**	**53.7**	**45.2**	**36.6**	**56.1**	**51.8**	**43.3**	**51.2**	**36.5**	**29.9**
SurroundOcc [98]	C	44.3	33.1	24.1	36.1	23.9	14.7	44.1	32.9	23.7	28.2	15.0	5.7
	C (S)	46.2	34.4	25.4	**40.8**	25.4	16.2	45.5	34.1	25.1	**36.1**	16.7	7.3
RadarOcc (Ours)	4DRT	**48.8**	**39.1**	**30.4**	34.3	**28.5**	**22.6**	**47.9**	**38.2**	**29.4**	20.7	**18.7**	**15.8**

Table 6.5: Quantitative comparison between `RadarOcc` and state-of-the-art methods based on LiDAR and camera. Results are reported on K-Radar well-condition test split. $(\cdot)$ is the number of LiDAR beams and (S) denotes stereo. The top four methods are colored as **red**, **green**, **blue**, and **orange**.

GTX 2090, we believe this enhanced inference speed demonstrates the potential for real-time application of our method in future vehicle systems, especially if further model quantization is applied.

6.2.5 Comparison between Different Modalities

To enrich our benchmark results and provide insights into the performance comparison between different modalities, we also evaluate state-of-the-art baseline methods [85, 98] on LiDAR and camera input. Quantitative results on K-Radar well-condition test split are reported in Tab. 6.5, while examples of qualitative results on K-Radar adverse-weather testing splits are exhibited in Fig. 6.3.

Quantitative results under normal weathers. As seen in Tab. 6.5, not surprisingly, LiDAR-based L-baselines [85] rank the top three in most metrics thanks to LiDAR's low-noise and high-resolution measurements (*vs.* radar) and direct depth measurement (*vs.* camera). Due to the inherently lower resolution and considerable noise of radar data, radar-based methods exhibit inferior to LiDAR-based methods in normal weather. However, `RadarOcc` still shows comparable performance to 16-beam LiDAR, and surpasses monocular and stereo camera-based method in most metrics. Notably, `RadarOcc` outperforms state-of-the-art SurroundOcc [98] relatively by 39.5%/19.7% and 53.7%/26.1% in mIoU/IoU@51.2m for stereo and monocular input, respectively. Stereo camera-based SurroundOcc [98] ranks third on FG IoU and mIoU@12.8m because of stereo vision's ability to infer accurate depth at short ranges, where the disparity between the two images is more pronounced.

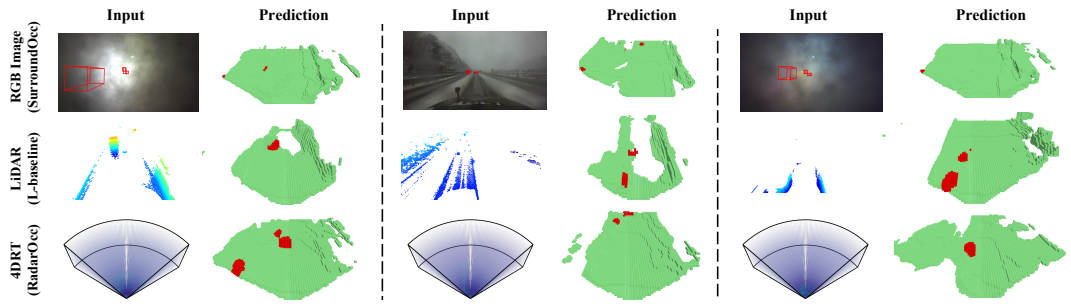


Figure 6.3: Qualitative comparison between RadarOcc, LiDAR-based L-baseline [85] and camera-based SurroundOcc [98] in adverse weathers. Ground truth bounding boxes are shown in RGB images.

Qualitative results under adverse weathers. While we have demonstrated the competitive performance of RadarOcc under normal weather, the key reason behind using radar for perception comes from its unique robustness against adverse weather where LiDAR and cameras fall short. To showcase such an inherent advantage, we provide some examples of qualitative results from different modalities in Fig. 6.3. As can be seen, RadarOcc provide robust 3D occupancy prediction under heavy rain and snow. In contrast, the camera lens are covered by the rain/snow and LiDAR measurements of some objects ahead are missing as water droplets or snowflakes can scatter and absorb the laser beams, leading to worse results. Please see Appendix B for more qualitative results.

6.3 Conclusion

In this chapter, we propose RadarOcc, a novel 3D occupancy prediction approach based on 4DRTs output from 4D imaging radar, enabling robust all-weather perception for autonomous vehicles. We analyse the rationale and challenges of using 4DRTs for 3D occupancy prediction and present tailored solutions to cope with the large, noisy and spherical 4DRTs. Experiments on the K-Radar dataset show RadarOcc's state-of-the-art performance in radar-based 3D occupancy prediction and comparable results to other modalities in normal weathers. Through qualitative analysis, we also exhibit its unique robustness against various adverse weathers. We believe our work could endorse the potential of 4D imaging radar to be an alternative to LiDAR and setup an effective baseline for further research and development of 4D radar-based occupancy perception.

6.4 Limitation and Future Work

As an initial investigation, RadarOcc demonstrate the capability of 4D radar for 3D occupancy prediction and setup an effective baseline for future research and development. However, this work has several limitations as outlined below.

No temporal modelling and forecasting. First, our method maps single-frame 4D radar data to single-frame 3D occupancy prediction without exploiting the temporal information and performing occupancy forecasting. To address this, future research can explore temporal modeling by designing a temporal transformer module that attends to features from previous frames, effectively capturing temporal dynamics. This approach could enhance prediction performance by mitigating frame-level noise and providing more robust, temporally consistent 3D occupancy predictions. To enable 3D occupancy forecasting, future work could focus on developing a world model capable of learning the temporal evolution of the environment [264]. Such a model would predict how the occupancy state changes over time, enabling the system to forecast future occupancy states. This capability is crucial for downstream planning and decision-making modules, as it supports proactive risk assessment and facilitates safer, more informed trajectory planning in dynamic environments.

Lack of fine-grained semantics. Second, due to the lack of point-wise semantic annotations in [52], our task is currently limited to coarse semantics, distinguishing only between foreground and background, which overlooks fine-grained background categories such as road, vegetation, and sidewalks. This limitation reduces the completeness of scene representation, which is critical for high-level decision-making in autonomous driving. To address this, we plan to generate pseudo-semantic labels for fine-grained background categories using foundation models like Grounded SAM [265], leveraging their strong generalization capabilities to create large-scale pseudo-labeled datasets without manual annotations. To mitigate potential noise and inconsistencies introduced by pseudo labels, we will adopt confidence-based filtering methods to selectively use high-fidelity pseudo labels and enhance temporal consistency through smoothing across consecutive frames.

Chapter 7

Conclusion

Robust spatial perception in diverse and complex scenarios is essential for realizing reliable and generalizable mobile autonomy in the wild. However, existing perception approaches relying solely on optical sensors often fail under adverse weather or poor illumination, raising significant risks to the reliabilty. Inspired by biological sensing, this thesis leverages 4D radar as a complementary modality to achieve robust perception. To explore 4D radar's potential for various spatial perception tasks, we propose bespoke methods that address both sensor- and task-specific challenges. We begin by tackling scene flow estimation, introducing self- and cross-modal supervised pipelines that eliminate manual annotations. These methods incorporate novel architectures and loss functions designed to handle radar's inherent sparsity and noise, yielding state-of-the-art results and supporting subtasks such as motion segmentation and odometry. Building upon the success of scene flow estimation, we extend to object-level perception with a radar-based solution for moving object detection and tracking. By focusing on motion segmentation and cluster-based detection, we circumvent the need for explicit 3D bounding boxes and object-type classification—an approach that demonstrates clear advantages when applied to sparse and noisy radar data. Finally, this thesis presents a pioneering method for 3D occupancy prediction using 4D radar tensors (4DRTs) instead of radar point clouds. This design choice preserves weak, yet critical, environmental signals typically lost during point cloud generation. Through techniques like Doppler bin descriptors, sidelobe-aware sparsification, and spherical-based encoding, we achieve competitive performance on challenging datasets, rivaling even LiDAR- and camera-based approaches in various weather conditions. Overall, our research demonstrates that 4D radar can be a robust, cost-effective, and all-weather perception modality, offering multi-level perception insights to bolster mobile autonomous systems.

Despite the progress achieved in this thesis, some limitations remain, both at the task-specific level, which have been at the end of Chapter 3-6, and in the broader context. The full potential of 4D radar has yet to be explored for developing a more robust and cost-effective ultimate perception solution for mobile autonomy. In the following, Section 7.1 outlines the general limitations of this work, while Section 7.2 presents potential future research directions to further advance 4D radar-based spatial perception. Lastly, Section 7.3 discusses the broader impact of the methods and ideas proposed in this thesis.

7.1 Limitations

Limited Dataset and Generalization Challenge. All methods proposed in this thesis—covering scene flow estimation, moving object tracking, and 3D occupancy prediction—are evaluated on relatively small-scale in-house or public datasets [52, 81]. This limitation poses two key challenges. First, the amount of training data is insufficient for learning models with strong generalization capabilities, especially when dealing with the inherent sparsity and noise of radar data. Second, the testing scenarios provided in these datasets are relatively constrained in terms of scene diversity, object types, and environmental conditions. As a result, the generalization ability of the proposed methods to complex, dynamic, and previously unseen environments remains unverified. A potential direction to address this limitation is discussed in Section 7.2.2, where we explore cross-modal conditioned generation to scale up 4D radar training datasets using richer sensor modalities such as cameras and LiDARs.

Lack of Onboard Testing in Real Autonomous Systems. While the methods proposed in this thesis demonstrate strong performance on curated and annotated datasets, they have not been deployed or tested on real mobile autonomous systems, such as self-driving vehicles or ground robots. As a result, the practical reliability of these methods under real-world conditions—such as sensor synchronization drift, platform vibrations, latency, or closed-loop interaction with planning modules—remains invalidated. Real-world deployment introduces challenges beyond offline evaluation, including system integration, real-time inference constraints, and perception-planning coupling, which are critical for safety and robustness in autonomous driving. Verifying these methods on actual platforms would not only provide deeper insights into their real-world reliability but also serve as a compelling demonstration of the potential of 4D radar as a standalone perception modality. Such deployment could accelerate the adoption of radar-centric

perception systems and promote further research and development in this field.

7.2 Future Works

7.2.1 Surrounding 4D Radar Perception

Currently, almost all 4D radar–based perception methods [46], including those presented in this thesis, employ a single front-facing radar sensor. This setup restricts the effective perception area to a limited field of view (FoV) (e.g., horizontal FoV: ±32°, vertical FoV: ± 22° [81]) and constraints the overall performance due the non-negligible sparsity and noise present in single 4D radar. Consequently, 4D radar is mostly treated as a supplement modality to LiDAR and camera providing resilience under adverse weather conditions [52, 266] and offering redundancy for multi-modal fusion [133, 267], rather than functioning as a standalone sensor for comprehensive spatial perception.

While the capability of a single 4D radar is bounded, it would be interesting to explore the integration of multiple 4D radar sensors on a vehicle to overcome these shortcomings. By strategically placing radars at different corners of an autonomous vehicle, we can achieve an omnidirectional spatial coverage of $360°$, enabling comprehensive surrounding perception [249]. Moreover, fusing data from multiple radars would allow us to develop a more competitive alternative to LiDAR-based spatial perception methods. Specifically, merging point clouds from multiple radars significantly increases point density and helps provide more complete measurements of ambient objects. Cross-sensor validation of data from co-located radars can mitigate the impact of noisy points; for instance, targets detected by only one radar but not corroborated by others can be identified as noise. As a result, the performance of 4D radar-based perception, e.g., 3D object detection, could be significantly enhanced given the improved quantity and quality of radar data. Furthermore, despite the additional hardware, a multi-radar configuration remains more cost-effective compared to deploying a single LiDAR. I believe such research efforts could underscore the potential of 4D radar, significantly promoting its adoption as a primary perception sensor in autonomous vehicles.

7.2.2 4D Radar Data Generation

Cross-modal conditioned generation. Many autonomous driving datasets that contain 4D radar data have been released in recent years to support the development of 4D radar-

based spatial perception algorithms [268]. However, the scale of these datasets remains relatively small, with annotated frames (around 10K) significantly fewer than those of popular autonomous driving datasets without radar, such as WOD (390K) [18] and Argoverse 2 (150K) [19]. This shortage in data scale and diversity hinders the training of robust and generalizable models. To address this issue, we draw inspiration from recent advances in vision foundation models, such as Cotracker [269] and LRM [270], which have demonstrated that large-scale synthetic data training, when followed by limited real-data finetuning, can lead to highly performant models. This training paradigm significantly reduces the demand for extensive manual annotation while maintaining strong generalization to real-world scenarios. Following this paradigm, one straightforward approach is to directly use synthetic 4D radar datasets generated through simulation platforms to pre-train radar perception models. However, the availability of high-fidelity 4D radar simulators and large-scale annotated radar datasets remains limited. Motivated by this, we propose to explore cross-modal generative frameworks for synthesizing 4D radar data conditioned on other modalities (e.g., camera or LiDAR), enabling the expansion of existing datasets that lack radar input. Rather than replacing real-world data collection, our goal is to augment training corpora with synthetic radar representations, thereby enhancing the scale and diversity of training samples in a cost-effective and controllable way. We envision that the generated data can support pre-training of 4D radar perception models, which can then be finetuned on smaller real-world datasets to achieve strong generalization.

Novel view synthesis. Unlike academic research, which typically relies on public autonomous vehicles dataset [17, 18, 52, 249] with limited scales (up to thousands of video clips), industrial models require training on at least millions of video sequences to ensure generalization and safety. For instance, Tesla's Full Self-Driving (FSD) models are trained on an extensive dataset comprising petabytes of data size and millions of driving miles, collected from its global fleet of vehicles [271]. This massive data collection introduces new challenges for stable and consistent 4D radar-based model training. One major challenge is the variability in sensor mounting positions across different vehicle models (e.g., Tesla Model X vs. Y), leading to discrepancies in sensors viewpoints that necessitate alignment for standardized training samples. Moreover, when deploying trained models across vehicles with different sensor setups, performance can degrade if models are not adapted to new viewpoints. Many studies have exploited novel view synthesis for camera and LiDAR sensors, generating images [272] or LiDAR point clouds [273] from new perspectives based on existing data. Building on this,

our future research aims to explore novel view synthesis for 4D radar data, enabling the generation of consistent radar representations across different vehicle platforms. Beyond addressing viewpoint discrepancies, this technique can also be applied for data augmentation, enhancing dataset diversity without the need for additional real-world data collection. Ultimately, this approach has the potential to improve the robustness and scalability of 4D radar-based perception systems in real-world applications.

7.3 Broader Impact

The advances presented in this thesis for 4D radar-based spatial perception have the potential to significantly impact the broader field of mobile autonomy, including autonomous vehicles, drones, and field robots. By addressing the limitations of traditional optical sensors—particularly under adverse weather and poor illumination—our methods contribute to more reliable and robust autonomous systems, improving operational safety, reducing failure risks, and enabling consistent perception in scenarios where cameras and LiDARs often underperform.

These approaches enable all-weather perception, allowing autonomous platforms to function reliably in challenging conditions such as rain, fog, and darkness. This capability is essential for the long-term and large-scale deployment of mobile autonomous systems, reducing their dependency on ideal conditions and minimizing human intervention for maintenance. Such resilience supports continuous operations in logistics, agriculture, search and rescue, and urban mobility—domains where robustness and adaptability are critical.

While autonomous driving remains a key application area, the methods developed in this thesis are well suited to other mobile autonomy scenarios that demand reliable perception in complex or degraded environments. In fact, given the generally lower complexity of many such environments (e.g., structured indoor or semi-structured outdoor settings), our approaches are likely to be even more effective and efficient in these domains. Moreover, this thesis demonstrates that 4D radar is not only a strong complement to LiDAR and cameras, but also a promising candidate to become the primary 3D sensing modality in certain autonomous systems. Its advantages in cost, form factor, and environmental robustness make it especially attractive for lightweight and energy-efficient platforms.

Building on these advances, my long-term vision is to create a comprehensive 4D radar-based spatial perception ecosystem for mobile autonomy. This includes

developing solutions across the entire stack—from raw radar signal processing to high-level semantic reasoning—and promoting their integration into real-world platforms. By stimulating further research, encouraging interdisciplinary collaboration, and driving adoption in both industry and academia, I hope to establish 4D radar as a key enabler of safe, reliable, and scalable autonomous systems, transforming industries and improving lives across a range of application domains.

Appendix A

4D Radar Scene Flow Learning using Cross-modal Supervision

This appendix provide supplementary details and experimental results for Chapter 4, which is organized as follows:

- Appendix A.1 illustrates more details about the dataset and the splits we used in our experiments.
- Appendix A.2 lists all symbols used in our approach together with their meanings and dimensions.
- Appendix A.3 introduces more details about our multi-task model architecture.
- Appendix A.4 presents more implementation details on retrieving cross-modal supervision and losses.
- Appendix A.5 provides more detailed experimental results and analysis of our approach.
- Appendix A.6 shows more qualitative scene flow, motion segmentation and ego-motion estimation results.

Besides, we also provide **demo** videos at https://youtu.be/PjKgznDizhI.

A.1 Dataset Details

Dataset separation. As discussed in the main text, the test set annotations of the official *View-of-Delft* (VoD) dataset [81] are withheld for benchmarking. Our task, however, needs to compute custom scene flow metrics with ground truth labels, which demands manually annotated 3D bounding boxes. Thus, for our experiments, we

	Val	*Test*	*Train*
Number of frames	1,296	2,724	4,662
Number of sequences	4	7	13
Is it annotated?	True	True	False

Table A.1: Details of our new splits for the VoD dataset [81].

split the VoD dataset [81] ourselves. In these new splits, we kept the original *Val* set unchanged and divided a part of the original training set into a new *Test* set so that we can generate ground truth scene flow using object annotations for evaluation. The remaining sequences from the original training set are used for our training. To avoid wasting the data from the original testing set, we add its frames to our training data and form a new *Train* set. Note that we remove all annotations from the frames in this *Train* set and use it for self-supervised or cross-modal supervised learning methods. Concrete details of our splits can be found in Tab. A.1.

Data preprocessing. Given sequences of radar point clouds, we first filter out the radar points outside the Field-of-View (FoV) of the camera as only objects partially or fully within the camera FoV were annotated in the dataset originally. Then, we set a z-axis (up-down direction) range [-3m, 3m] and filter points outside of this range as these very low or high points are often ghost targets. During training, we randomly sample 256 radar points for each point cloud to facilitate fast mini-batch learning. During inference, however, we keep the original number of points because our task is to estimate the flow vector of every radar point in the source frame. Finally, for each sequence, we form $L_{seq} - 1$ scene flow input samples by combining pairs of consecutive radar point clouds, where L_{seq} is the length (i.e. number of frames) of the sequence.

Ground truth labelling. As mentioned in the main text, we only annotate ground truth scene flow and motion segmentation labels for the samples from *Val* and *Test* sets. Following [61, 247], we annotate ground truth scene flow by using object annotations (i.e., bounding boxes and track IDs) and ground truth radar ego-motion (calculated from the RTK-GPS/IMU based odometer). For points belonging to the static background, we label their flow vectors with the ground truth radar ego-motion. For foreground objects, we track the ID of each annotated bounding box across consecutive point clouds and compute their rigid transformation *w.r.t* the radar coordinate frame. Each foreground point is assigned a ground truth flow vector, which is summarized by this corresponding rigid transformation. Then, for each foreground point belonging to an

Symbol	Meaning	Dimension	Symbol	Meaning	Dimension
$\mathbf{P}^s$	Input source point cloud	$\mathbb{R}^{N\times(3+C)}$	$\mathbf{P}^t$	Input target point cloud	$\mathbb{R}^{M\times(3+C)}$
$\mathbf{p}_i^s, \mathbf{p}_i^t$	i-th point of $\mathbf{P}^s$, $\mathbf{P}^t$	$\mathbb{R}^{3+C}$	$\mathbf{c}_i^s, \mathbf{c}_i^t$	3D coordinate of point $\mathbf{p}_i^s, \mathbf{p}_i^t$	$\mathbb{R}^3$
$\mathbf{x}_i^s, \mathbf{x}_i^t$	Associated raw features of point $\mathbf{p}_i^s, \mathbf{p}_i^t$	$\mathbb{R}^C$	N	Number of points in the source point cloud	$\mathbb{R}$
M	Number of points in the target point cloud	$\mathbb{R}$	C	Number of channels of the associated raw features	$\mathbb{R}$
$\mathbf{F}$	Scene flow between two input point clouds	$\mathbb{R}^{N\times3}$	$\mathbf{f}_i$	Scene flow vector of the point $\mathbf{p}_i^s$	$\mathbb{R}^3$
$\mathbf{c}_i'$	3D coordinate of point $\mathbf{p}_i^s$ after warped by $\mathbf{f}_i$	$\mathbb{R}^3$	$\mathcal{L}$	Total loss value	$\mathbb{R}$
$\mathcal{L}_{ego}$	Ego-motion loss value	$\mathbb{R}$	$\mathcal{L}_{seg}$	Motion segmentation loss value	$\mathbb{R}$
$\mathcal{L}_{scene}$	Scene flow loss value	$\mathbb{R}$	$\mathbf{E}$	Base backbone features	$\mathbb{R}^{N\times C_e}$
C_e	Number of channels of the backbone features	$\mathbb{R}$	$\hat{\mathbf{F}}^{init}$	Initial scene flow estimation	$\mathbb{R}^{N\times3}$
$\hat{\mathbf{f}}_i^{init}$	Initial scene flow vector estimation of point $\mathbf{p}_i^s$	$\mathbb{R}^3$	$\hat{\mathbf{S}}$	Estimated moving probabilities	$\mathbb{R}^N$
$\hat{s}_i$	Estimated moving probability of point $\mathbf{p}_i^s$	$\mathbb{R}$	$\hat{\mathbf{T}}$	Estimated rigid transformation	$\mathbb{R}^{4\times4}$
$\mathbf{I}_4$	Identity matrix	$\mathbb{R}^{4\times4}$	$\hat{\mathbf{F}}$	Final scene flow estimation	$\mathbb{R}^{N\times3}$
$\hat{\mathbf{f}}_i$	Final scene flow vector estimation of point $\mathbf{p}_i^s$	$\mathbb{R}^3$	T	The length of mini-clips used for temporal update	$\mathbb{R}$
$\mathbf{O}$	Ground truth ego-motion transformation	$\mathbb{R}^{4\times4}$	$\mathbf{T}$	Ground truth rigid transformation	$\mathbb{R}^{4\times4}$
$\mathbf{F}^r$	Ground truth rigid flow components	$\mathbb{R}^{N\times3}$	$\mathbf{f}_i^r$	Ground truth rigid flow component of point $\mathbf{p}_i^s$	$\mathbb{R}^3$
v_i	Radar RRV measurement of point $\mathbf{p}_i^s$	$\mathbb{R}$	v_i^r	RRV component ascribed to the radar ego-motion	$\mathbb{R}$
$\mathbf{u}_i$	The unit radial vector of the point $\mathbf{p}_i^s$	$\mathbb{R}^3$	Δt	Time duration between two frames	$\mathbb{R}$
Δv_i	Absolute radial velocity of the point $\mathbf{p}_i^s$	$\mathbb{R}$	$\mathbf{S}^v$	Pseudo motion segmentation labels from RRV	$\mathbb{R}^N$
s_i^v	Pseudo motion segmentation label for $\mathbf{p}_i^s$ from RRV	$\mathbb{R}$	$\mathbf{S}^{fg}$	Pseudo foreground segmentation labels from LiDAR	$\mathbb{R}^N$
$\mathbf{F}^{fg}$	Pseudo scene flow labels from LiDAR	$\mathbb{R}^{N\times3}$	$\mathbf{f}_i^{fg}$	Pseudo scene flow label for $\mathbf{p}_i^s$ from LiDAR	$\mathbb{R}^3$
$\mathbf{S}^l$	Pseudo motion segmentation labels from LiDAR	$\mathbb{R}^N$	s_i^l	Pseudo motion segmentation label for $\mathbf{p}_i^s$ from LiDAR	$\mathbb{R}$
$\mathbf{S}$	Final pseudo motion segmentation labels	$\mathbb{R}^N$	s_i	Final motion segmentation label for $\mathbf{p}_i^s$	$\mathbb{R}$
$\mathcal{L}_{mot}$	MOT-based scene flow loss value	$\mathbb{R}$	$\mathcal{L}_{opt}$	Optical flow-based scene flow loss value	$\mathbb{R}$
$\mathcal{L}_{self}$	Self-supervised scene flow loss value	$\mathbb{R}$	$\mathbf{W}$	Pseudo optical flow labels	$\mathbb{R}^{N\times2}$
$\mathbf{w}_i$	Pseudo optical flow vector label for point $\mathbf{p}_i^s$	$\mathbb{R}^2$	$\mathbf{m}_i$	Corresponding image pixel of point $\mathbf{p}_i^s$	$\mathbb{R}^2$
λ_{opt}	Weighting parameter for $\mathcal{L}_{opt}$	$\mathbb{R}$	θ	Sensor calibration parameters	-
$\mathcal{D}(\cdot,\cdot,\cdot)$	Point-to-ray distance computing function	-	$\|\|\cdot\|\|_2$	L_2-norm, also known as Euclidean Distance	-

Table A.2: A list of all used symbols with their meanings and dimension in our approach.

object, we assign a ground truth flow vector derived from the rigid transformation of the object. To obtain the ground truth motion segmentation, we first compute the non-rigid flow vector $\mathbf{f}_i^{nr}$ for each point by compensating the ground truth rigid ego-motion as $[\mathbf{f}_i^{nr}\ 1]^\top = [\mathbf{f}_i^{gt}\ 1]^\top - (\mathbf{T}-\mathbf{I}_4)[\mathbf{c}_i\ 1]^\top$. Then, points with $|\mathbf{f}_i^{nr}| > \zeta_{mov} = 5\text{cm}$ are labelled as moving, while the rest are labelled as static.

A.2 Notation

For the convenience of reference, we summarize all symbols used in our approach together with their meanings and dimensions in Tab. A.2.

A.3 Model Architecture Details

Here, we introduce more details about our module design and layer hyperparameters of our model architecture.

Backbone. The input to our backbone is two consecutive 4D radar point clouds, $\mathbf{P}^s \in \mathbb{R}^{N\times(3+C)}$ and $\mathbf{P}^t \in \mathbb{R}^{M\times(3+C)}$. As the foremost step, we use the *set conv* layer [55] to extract local features at different scales. Detailed layer parameters of this multi-scale *set conv* layer are as follows:

syntax: $SC([radii], [nsamples], [dimension])$

where $[radii]$ denotes the grouping radii for multiple scales, $[nsamples]$ denotes the number of local sampled points, and $[dimension]$ are the latent feature sizes.

$$SC([2.0, 4.0, 8.0, 16.0], [4, 8, 16, 32], [[32, 32, 64], [32, 32, 64], [32, 32, 64], [32, 32, 64]]) \rightarrow MLP(256 \rightarrow 256 \rightarrow 256 \rightarrow 256)$$

Note that we do not downsample our radar point clouds because they are already very sparse. After concatenating the global feature vector to point-wise features, we obtain the local-global features for each individual input point cloud: $g_\theta(\mathbf{P}^s) \in \mathbb{R}^{N\times512}$ and $g_\theta(\mathbf{P}^t) \in \mathbb{R}^{M\times512}$.

To propagate the features from the target point cloud to the source, we adopt the *cost volume* layer in [56] for feature correlation. The costs are aggregated in a robust patch-to-patch manner and a MLP was used to learn point-to-point cost dynamically:

$$MLP(512 \times 2 + 3 \rightarrow 512 \rightarrow 512 \rightarrow 512)$$

In patch-to-patch cost aggregation, the number of neighbour points is set to 8. As a results, we can obtain the correlated features $h_\theta(g_\theta(\mathbf{P}^s), g_\theta(\mathbf{P}^t)) \in \mathbb{R}^{N\times512}$.

The flow embedding $FE \in \mathbb{R}^{N\times(512+512+C)}$ can be generated by concatenating the correlated features, the local-global features and the raw input features of $\mathbf{P}^s$. We further feed this flow embedding into another multi-scale *set conv* layer to get $\mathbf{L} \in \mathbb{R}^{N\times256}$:

$$SC([2.0, 4.0, 8.0, 16.0], [4, 8, 16, 32], [[512, 256, 64], [512, 256, 64], [512, 256, 64], [512, 256, 64]]) \rightarrow MLP(256 \rightarrow 256 \rightarrow 256 \rightarrow 256)$$

Another max-pooling operation follows to restore the global feature vector and concatenates it to each point. Finally, we obtain our base backbone features $\mathbf{E} \in \mathbb{R}^{N\times512}$.

Initial flow and motion segmentation head. As the base backbone feature $\mathbf{E}$ includes complementary features extracted from two input point clouds, we simply use the MLP

to implement our initial flow and motion segmentation heads for decoding. The output of the initial flow are 3D flow vectors $\hat{\mathbf{F}}^{init} \in \mathbb{R}^{N\times 3}$:

$$MLP(512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 3)$$

The motion segmentation head estimates per-point moving probabilities $\hat{\mathbf{S}} \in \mathbb{R}^{N}$:

$$MLP(512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 1).$$

Ego-motion head. The purpose of our ego-motion head is to derive a rigid transformation $\hat{\mathbf{T}}$ that can summarize the scene flow rigid component induced by the radar ego-motion. To that end, we leverage the differentiable weighted Kabsch algorithm [183] that solves the ego-motion estimation problem in a close form through:

$$\hat{\mathbf{T}}^{*} = \underset{\hat{\mathbf{T}}\in\mathbb{R}^{4\times 4}}{\arg\min} \sum_{i=1}^{N} w_i ||(\hat{\mathbf{R}}\mathbf{c}_i^s + \hat{\mathbf{t}}) - \mathbf{c}_i^w||_2^2, \tag{A.1}$$

where $\mathbf{c}_i^w = \mathbf{c}_i^s + \hat{\mathbf{f}}_i^{init}$ is the point coordinate warped by the initial scene flow. The sum of all weights is normalized as $\sum_{i=1}^{N} w_i = 1$, and

$$\hat{\mathbf{T}} = \begin{bmatrix} \hat{\mathbf{R}} & \hat{\mathbf{t}} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \tag{A.2}$$

where $\mathbf{R} \in \mathbb{R}^{3\times 3}$ is the rotation matrix and $\mathbf{t} \in \mathbb{R}^{3}$ is the translation vector. In our method, we compute the weights from the estimated moving probabilities $\hat{\mathbf{S}}$ via:

$$w_i = \frac{1-\hat{s}_i}{\sum_{i=1}^{N}(1-\hat{s}_i)}. \tag{A.3}$$

To solve Eq. (A.1), the first step is to compute the centred point coordinates for $\mathbf{C}^s = \{\mathbf{c}_i^s\}_{i=1}^N$ and $\mathbf{C}^w = \{\mathbf{c}_i^w\}_{i=1}^N$. To do that, the weighted centroid of $\mathbf{C}^s$ and $\mathbf{C}^w$ can be derived through:

$$\bar{\mathbf{c}}^s = \sum_{i=1}^{N} w_i \mathbf{c}_i^s, \quad \bar{\mathbf{c}}^w = \sum_{i=1}^{N} w_i \mathbf{c}_i^w \tag{A.4}$$

We subtract the centroid coordinates from $\mathbf{C}^s$ and $\mathbf{C}^w$ and obtain the centered point coordinates $\tilde{\mathbf{C}}^s \in \mathbb{R}^{N\times 3}$ and $\tilde{\mathbf{C}}^w \in \mathbb{R}^{N\times 3}$. A weighted covariance matrix $\mathbf{H}_w \in \mathbb{R}^{3\times 3}$ can then be formulated as:

$$\mathbf{H}_w = \tilde{\mathbf{C}}^{s\top} diag(w_1, w_2, ..., w_N) \tilde{\mathbf{C}}^w. \tag{A.5}$$

The optimal rotation matrix $\hat{\mathbf{R}}^*$ is solved by:

$$\hat{\mathbf{R}}^{*} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} \mathbf{U}, \tag{A.6}$$

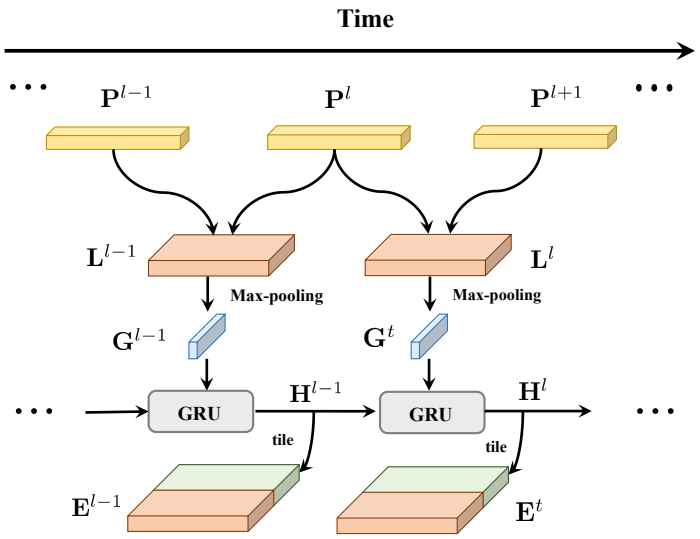


Figure A.1: Temporal update module. We concatenate the updated hidden state $\mathbf{H}$ to each point feature when activating this module, while we use the global feature vector $\mathbf{G}$ directly when the updating module is disabled. Note that the superscript (e.g., l) denotes the temporal order index and we discard it in other places.

where the singular value decomposition (SVD) is used in $\mathbf{H}_w = \mathbf{U}\sum\mathbf{V}$ and $d = sign(det(\mathbf{V}\mathbf{U}^\top))$ is the correction value to avoid special reflection cases. As the final step, we compute the optimal translation vector as:

$$\hat{\mathbf{t}}^* = \bar{\mathbf{c}}^w - \hat{\mathbf{R}}^*\bar{\mathbf{c}}^s. \tag{A.7}$$

Temporal update module. This module is embedded in our backbone module and its use in our method is optional. If enabled, it can propagate information from previous frames to the current frame. As a reminder, before we generate our base backbone feature $\mathbf{E}$, we get the local features $\mathbf{L} \in \mathbb{R}^{N\times 256}$ with another multi-scale *set conv* layer. The global feature vector is obtained by $\mathbf{G} \in \mathbb{R}^{256} = MAX(\mathbf{L})$, where MAX is the max-pooling operation along the channel axis. Our temporal update module operates on $\mathbf{G}$ and applies a GRU network [237] to update it as a hidden state temporally, as seen in Fig. A.1.

A.4 Cross-Modal Supervision Details

In this section, we provide some implementation details on cross-modal supervision retrieving and loss formulation.

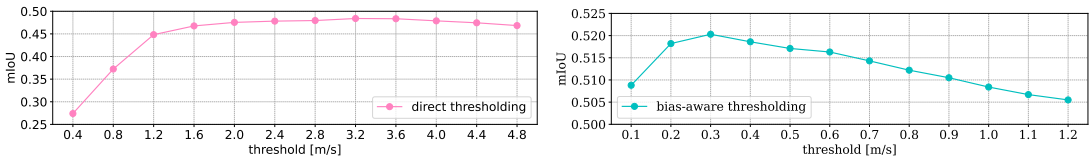


Figure A.2: The impact of threshold η_v when applying direct and bias-aware thresholding to generate pseudo motion segmentation labels $\mathbf{S}^v$ with the ego-motion and RRV measurements. Here, mIoU is computed between the ground truth motion segmentation labels and the generated pseudo one on the *Val* set.

Strategy for generate pseudo label $\mathbf{S}^v$. As introduced in the main text, we propose to generate a pseudo motion segmentation label $\mathbf{S}^v$ with the odometry information and radar RRV measurements. As a reminder, the pseudo label $\mathbf{S}^v$ is created by thresholding the ego-motion compensated RRV Δv_i, i.e., the absolute radial velocity. Next, we will discuss our specific thresholding strategy.

Given per-point Δv_i, it is intuitive to generate a motion segmentation mask by labelling points with larger values than a fixed threshold as moving points. However, this straightforward strategy suffers from the temporal offset between the well-curated keyframes and original radar point clouds. Even though the coordinates of radar points are transformed to the timestamp of keyframes using high-frequency odometry information, their RRV measurements still correspond to the original timestamps. As a result, there is often a non-trivial global bias in Δv_i, which disturbs the identification of real moving points given a fixed threshold. For example, almost all points could be classified as moving when a large bias exists. To mitigate this issue, we propose to *normalize* all Δv_i by subtracting their mean value μ and then classify points by thresholding. The pseudo motion segmentation label $\mathbf{S}^v = \{s_i^v \in \{0, 1\}\}_{i=1}^N$ can then be created with a fixed threshold η_v, where 0 represents stationary points. We determine the value of η_v via grid searching on the *Val* set, as seen in Fig. A.2.

To validate if our bias-aware strategy helps to improve the quality of pseudo motion segmentation labels $\mathbf{S}^v$, we compare this advanced strategy to the direct (i.e. not bias-aware) one. As the pseudo motion segmentation label generation is affected by the threshold η_v, we iteratively change its value and observe how the mIoU between the ground truth and pseudo motion segmentation labels varies with the threshold. Fig. A.2 shows the comparison between direct thresholding and bias-aware thresholding on the *Val* set. Without considering the global bias of residuals, direct thresholding can only give a maximum mIoU of 48.4%, while our proposed bias-aware thresholding produces

a mIoU of 52%[1]. The results demonstrate the effectiveness of our advanced strategy to tackle the temporal misalignment of ego-motion and RRV measurements. Moreover, as shown in the top of Fig. A.2, the mIoU increases to near 50% only when the threshold is large enough (> 1.0m/s), which further proves the presence of a global bias in Δv_i. For the best quality of pseudo label $\mathbf{S}^v$, we set $\eta_v = 0.3$ in all our experiments according to the results on the *Val* set exhibited in Fig. A.2.

LiDAR multi-object tracking. As another active ranging sensor, LiDAR can detect targets and measure their 3D positions by emitting pulsed lasers [274]. Compared with 4D radar, it can restore the geometric structure well with denser point data under satisfactory weather conditions. Some recent works [71, 72, 200] perform 3D multi-object tracking (MOT) on LiDAR point clouds in a *tracking-by-detection* paradigm and show prominent results. Inspired by the progress, we propose to extract supervision signals from LiDAR by running 3D MOT algorithms on LiDAR point clouds. The method we employ is called AB3DMOT [71], which first detects 3D bounding boxes with a pretrained PointRCNN [195] model and tracks objects based on a 3D Kalman filter. To implement this method, we first use the OpenPCDet[2] library and adopt the pretrained PointRCNN model to perform the object detection stage. The x, y, z range is set to [(0, 70.4), (-40, 40), (-3, 1)] meters respectively and the number of points is set as 4096 following the original paper [195]. Similar to the authors of the VoD dataset [81], we only detect three classes of object, i.e. *pedestrian, cyclist* and *car*, as foreground objects. For the tracking part, we use the official code[3] and take the detection results as input. Specifically, we set the minimal birth length Bir_{min} as 4 and the maximum death length Age_{max} as 8. The centre distance threshold for object association is 2 meters. As a result, the 3D MOT algorithm can give us a set of estimated bounding boxes and their track IDs for each frame.

Optical flow loss $\mathcal{L}_{opt}$. Different from LiDAR or radar point clouds, RGB images contain explicit object appearance and texture information, which provide rich semantics cues for inferring pixel-level 2D motions between frames, i.e., optical flow estimation. As a 2D perspective projection of 3D scene flow, optical flow describes the motion of points on the image plane. Thus it can also be used to partially supervise the scene flow estimation of points, whose perspective projection is within the camera field-of-view (FoV). Thanks to the advance in data-driven optical flow estimation [238, 275], we

[1]Note that tangentially moving targets are not distinguished in the pseudo label $\mathbf{S}^v$ because radar only measures the relative radial velocity.

[2]https://github.com/open-mmlab/OpenPCDet

[3]https://github.com/xinshuoweng/AB3DMOT

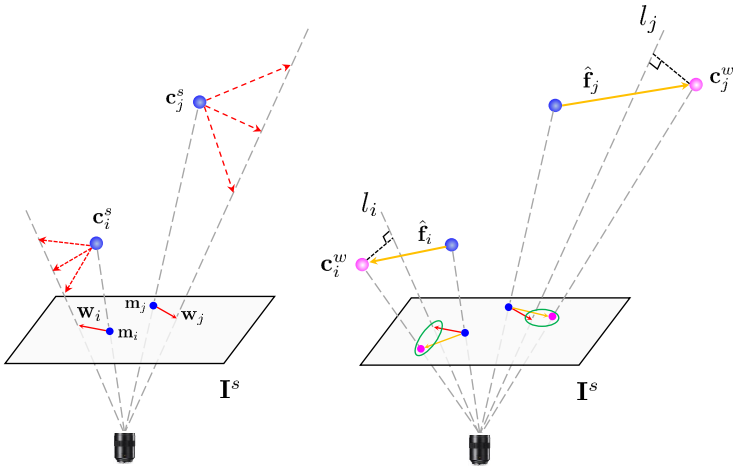


Figure A.3: Left figure: Blue denotes two points (i.e., $\mathbf{c}_i^s$, $\mathbf{c}_j^s$) from the source point cloud $\mathbf{P}^s$ and their perspective projection $\mathbf{m}_i, \mathbf{m}_j$ on the corresponding image $\mathbf{I}^s$. Red arrows denote pseudo optical flow labels $\mathbf{w}_i, \mathbf{w}_j$ and their possible 3D projections (dashing line). Right figure: Yellow denotes the predicted scene flow vectors $\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j$ and their corresponding optical flow after perspective projection. Magenta denotes warped points $\mathbf{c}_i^w, \mathbf{c}_j^w$ and their perspective projection on image $\mathbf{I_s}$. l_i and l_j denote the corresponding rays of pixels warped by pseudo optical flow labels.

can use a pretrained model to infer pseudo optical flow labels efficiently. For one pair of $\mathbf{P}^s$ and $\mathbf{P}^t$, we can input their synchronized monocular images $\mathbf{I}^s$ and $\mathbf{I}^t$ into the network and obtain an optical flow map. We adopt the RAFT-S [238] pretrained model[4] to estimate our optical flow. The number of iterations is set as 12 in practice. After drawing per-point optical flow vector $\mathbf{W} = \{\mathbf{w}_i \in \mathbb{R}^2\}_{i=1}^N$, we formulate our optical flow loss to constrain our scene flow prediction in the perspective view. In the main text, due to the depth-unawareness during perspective projection, we propose to minimize the point-to-ray distance instead of the flow divergence in pixel scale. Our motivation is further illustrated in Fig. A.3, where a larger distance between the warped point $\mathbf{c}_j^w$ and the expected ray l_j results in a smaller pixel-scale loss after perspective projection for a farther point $\mathbf{c}_j^s$. In practice, we discard point-to-ray distances lower than 0.25m when computing the loss to mitigate the impact of optical flow estimation errors.

Self-supervised loss. Motivated by self-supervised scene flow works [56, 59–61, 124], we also utilize self-supervised loss functions to complement our cross-modal supervised learning. Specifically, we leverage three self-supervised loss functions from [124] and keep their default hyperparameter settings. The first one is called soft Chamfer loss,

[4]https://github.com/princeton-vl/RAFT

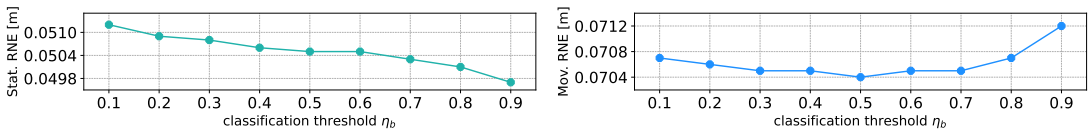


Figure A.4: Impact of the classification threshold η_b on the *Val* set. The value is changed from 0.1 to 0.9 by a step of 0.1.

which constrains scene flow by pulling mutual pseudo correspondences between the target point cloud $\mathbf{P}^t$ and the warped source point cloud $\mathbf{P}^w$ close to each other. In practice, the noise from outliers is mitigated and small errors led by low-resolution are discarded, to make the loss *soft* to intractable radar point clouds. The second loss is spatial smoothness loss, which is used for local consistency in scene flow estimation. In particular, a distance-based weight is computed at first and then the spatial smoothness of the predicted scene flow is enforced accordingly. The third loss is called radial displacement loss, where the radial projections of estimated scene flow are constrained by the RRV measurements.

A.5 More Experimental Analysis

Impact of the classification threshold η_b. As an important hyperparameter, η_b is used to threshold the estimated moving probabilities $\hat{\mathbf{S}}$ to generate the binary motion segmentation mask during inference, which further determines the flow vectors of which points are refined with the ego-motion estimation. To select its optimal value, we run a series of evaluations on the *Val* set with the trained CMFlow model and show the results in Fig. A.4. As we increase the threshold value, the performance on Stat. RNE gets improved continuously, however the optimal threshold for Mov. RNE is 0.5. Considering that the scene flow accuracy on both moving and static points is crucial, we aim to seek a trade-off between Stat. RNE and Mov. RNE. We thus set the $\eta_b = 0.5$ as the optimal value for our experiments.

Impact of the mini-clip length T. When activating our temporal update module in the backbone, we can update the global feature vector temporally to propagate information from previous frames. However, training with long sequences in a brute-force way will lead to non-negligible over-fitting. To mitigate the issue, we split long training sequences into many mini-clips with a length of T and train mini-batches of them. During inference, the hidden state is re-initialized after T frames to emulate the training conditions. Here, we exhibit how to select the optimal T value in our experiments,

T	EPE [m]↓	AccS↑	AccR↑	RNE [m]↓
1	0.132	0.263	0.552	0.053
3	0.128	0.280	0.560	0.051
5	**0.122**	**0.295**	**0.579**	**0.049**
7	0.131	0.264	0.560	0.053
10	0.129	0.260	0.551	0.052
15	0.142	0.203	0.488	0.057

Table A.3: Impact of the mini-clip length for temporal feature update. The results shown above are obtained by evaluating our CMFlow (T) method on the *Val* set. The best T (**bold**) are used for experiments on the *Test* set.

as seen in Tab. A.3. As we increase the value of T, the performance on the *Val* set improves because temporal information from more previous frames can be used for the current frame. However, when the value of T exceeds 5, the performance starts to degrade and becomes even worse than the one without temporal update when $T = 15$. We attribute this to two reasons. First, as we discuss above, longer clips can exacerbate the over-fitting issue. Second, using long-term information from earlier frames will introduce more noise and disturb the current feature extraction.

Performance of fully-supervised methods. In our main experiments, we are interested in if our performance could catch up or surpass that of fully-supervised methods when more unannotated data is available for training. As a reminder, we select the state-of-the-art fully-supervised method, PV-RAFT [175], for comparison. Here for completeness, we also evaluate another five fully-supervised methods and compare them together with PV-RAFT [175] in Tab. A.4. Note that we use all available annotated samples in *Train* set and label their ground truth scene flow with object annotations (c.f. Appendix A.1) for fully-supervised training. Under the same training setting, PV-RAFT [175] achieves the best results among fully-supervised methods. This further proves its state-of-the-art performance and supports our choice of PV-RAFT as the compared fully-supervised method when more unannotated training data is available.

Impact of self-supervised loss. As a complement to our cross-modal supervision, the self-supervised loss can exploit the inherent spatio-temporal relationship and constraints in the input data to bootstrap scene flow learning. Here, we analyse its impact by ablating it from our method. The results are shown in Tab. A.5. With the self-supervised loss, the performance of CMFlow improves in all metrics. This demonstrates that combining self-supervised and cross-modal supervision provides complementary learning cues and

Method	EPE [m]↓	AccS↑	AccR↑	RNE [m]↓
FlowNet3D [55]	0.201	0.169	0.379	0.081
PointPWC-Net [56]	0.196	0.177	0.391	0.079
FlowStep3D [57]	0.286	0.061	0.185	0.115
PV-RAFT [175]	**0.126**	**0.258**	**0.587**	**0.051**
3DFlow [276]	0.277	0.104	0.294	0.111
Bi-PointFlowNet [277]	0.242	0.164	0.350	0.097

Table A.4: Comparison between fully-supervised radar scene flow methods on the *Test* set. All methods are trained using the same annotated training samples with ground truth scene flow.

Method	EPE [m]↓	AccS↑	AccR↑	RNE [m]↓
w.o. $\mathcal{L}_{self}$	0.152	0.221	0.494	0.061
w. $\mathcal{L}_{self}$	**0.141**	**0.233**	**0.499**	**0.057**
Gain	-0.011	+0.012	+0.005	-0.004

Table A.5: Analysis of the impact of self-supervise loss. For the top row, we ablate the self-supervised loss during training.

leads to more accurate scene flow estimation.

Impact of LiDAR modality. In our cross-modal supervised learning pipeline, LiDAR supervision is used in two ways to support our model training. One is to generate a pseudo motion segmentation label $\mathbf{S}^l$, which is further used to obtain a more reliable label $\mathbf{S}$ after fusing with $\mathbf{S}^v$. Another is to provide the pseudo scene flow label $\mathbf{F}^{fg}$ used to formulate the loss $\mathcal{L}_{mot}$ that supervises the final scene flow. Here, we analyze the impact of LiDAR modality in these two ways. As seen in Tab. A.6, the LiDAR modality contributes to our improvement gain in both aspects. First (c.f. row (b)), using the pseudo label $\mathbf{S}^l$ can effectively complement the label $\mathbf{S}^v$ where tangentially moving targets are not distinguished. Second (c.f. row (c)), the scene flow loss $\mathcal{L}_{mot}$ can constrain the scene flow vectors of identified moving points in $\mathbf{S}^l$. When using the LiDAR modality in both ways, an even larger improvement can be achieved. This demonstrates that each supervision component in our framework is significant and correlates to each other compactly.

	L (seg)	L (flow)	EPE [m]↓	AccS↑	AccR↑	RNE [m]↓
(a)			0.159	0.216	0.458	0.064
(b)	✓		0.156	0.221	0.467	0.063
(c)		✓	0.152	0.217	0.477	0.061
(d)	✓	✓	**0.141**	**0.223**	**0.499**	**0.057**

Table A.6: Ablation study for LiDAR modality. L (seg) denotes that the LiDAR supervision is used for generating pseudo motion segmentation labels, while L (flow) denotes that the LiDAR supervision is used to formulate the scene flow loss.

A.6 More Qualitative Results

Scene flow estimation. More qualitative scene flow results are shown in Fig. A.5. It can be seen that CMFlow can accurately estimate scene flow vectors in diverse driving scenarios. After warping the source point cloud with the estimated scene flow, both static background and multiple dynamic objects can be aligned well between two frames.

Motion segmentation. Additional qualitative results for the motion segmentation task can be seen in Fig. A.6. Without any ground truth labels used for training, our method results in accurate motion segmentation in dynamic environments. Different moving objects (e.g., car, cyclist, pedestrian) can be segmented from the static background well.

Ego-motion estimation. We show more qualitative odometry results in Fig. A.7. As a byproduct of our method, the estimated rigid ego-motion transformation can effectively support the odometry task in complex urban driving scenarios. Compared to the baseline ICP [239] that directly solves the transformation with all points, CMFlow can distinguish and mitigate the impact of moving points, and thus has better odometry performance in dynamic environments.

Figure A.5: Qualitative scene flow results in seven *Test* scenes. From left to right: 1) radar points from the source frame projected to the corresponding RGB image (points are coloured by distance from the sensor), 2) two input radar point clouds, the source one (pink) and the target one (green), 3) the source point cloud warped by our predicted scene flow and the target radar point cloud, 4) the source point cloud warped by ground truth scene flow and the target one. We mark regions of interest in amber and apply the zooming-in operation for them.

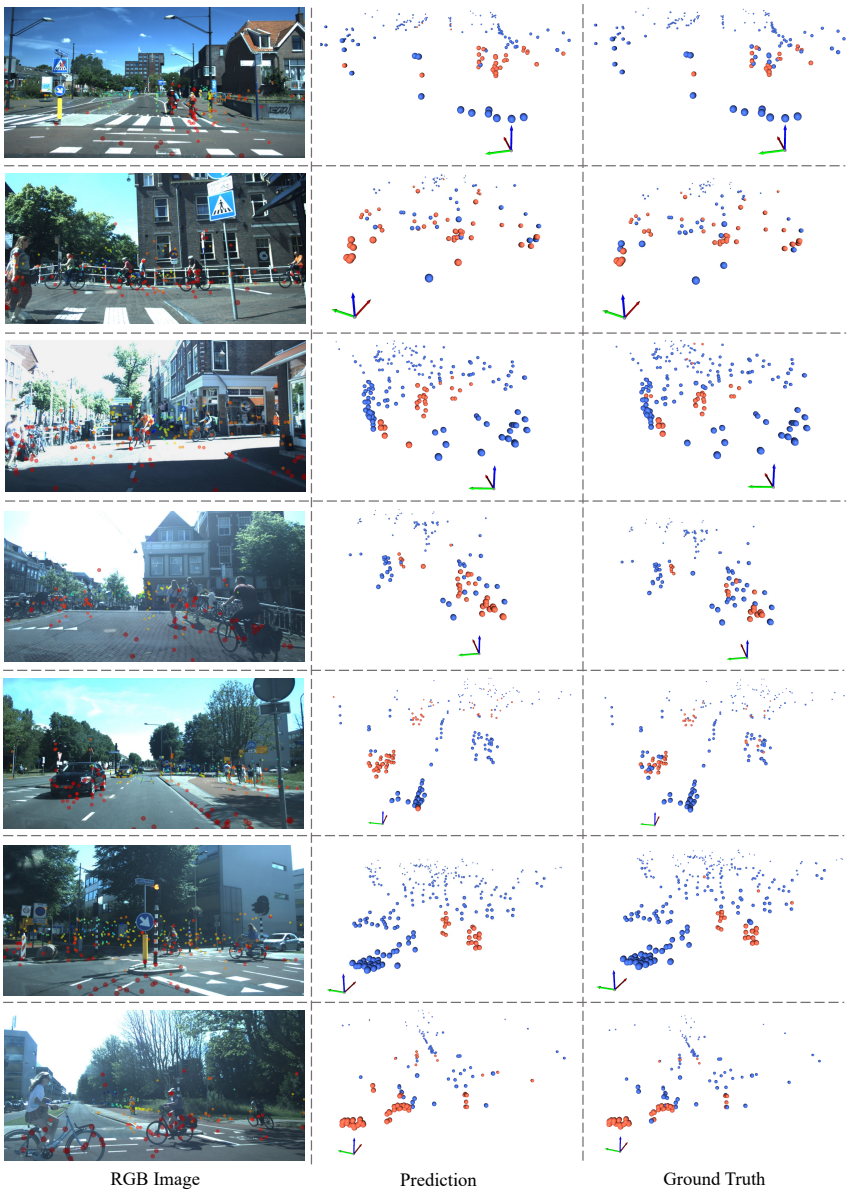


Figure A.6: Visualization of motion segmentation. The left column shows the corresponding image with radar points (coloured by range) projected onto it. In the middle and right columns, moving points are shown in orange while static points are shown in blue.

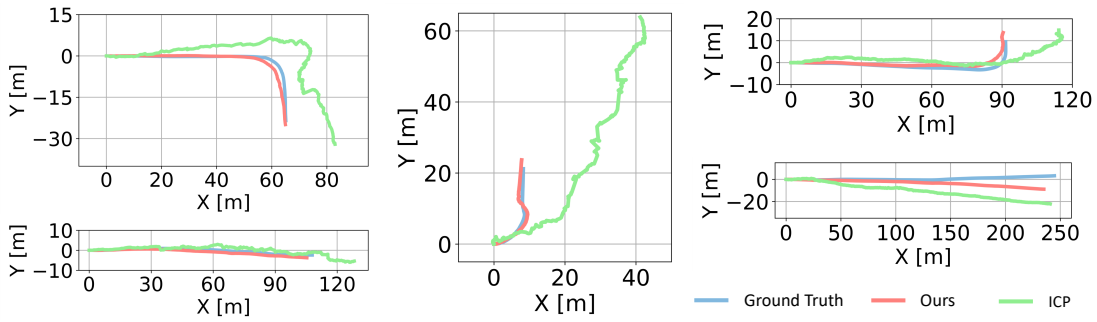


Figure A.7: Qualitative odometry results in five *Test* sequences. To plot the ego-vehicle trajectory, inter-frame ego-motion transformations are accumulated temporally. Please see the supplementary video for dynamic trajectory results.

Appendix B

Robust 3D Occupancy Prediction with 4D Imaging Radar

This appendix include the supplementary details and experimental results for Chapter 6, which is organized as follows:

- Section B.1 illustrates more details on our experiment setup, including ground truth generation, dataset statistics, evaluation area and computation resources we used for our experiments.
- Section B.2 introduces implementation details of different components in `RadarOcc`.
- Section B.3 gives more experimental results, visualization and failure case of `RadarOcc`.

Besides, please refer to our demo video for more qualitative results: `https://www.youtube.com/watch?v=sa4gejcWMvk`.

B.1 Experiment Setup Details

Ground truth generation. Our pipeline of 3D occupancy annotation is similar to those in [85, 98, 99, 263]. First, we split each LiDAR point cloud from a sequence into the background and foreground part with the help of 3D bounding box annotations. For the background, we superimpose all LiDAR points by transforming them into a united world coordinate using their extrinsic. For the foreground part, we track the same instances (indicated by the same tracking IDs) across the sequence and transform LiDAR points association to them into the coordinates of their bounding boxes. In this way, sparse LiDAR point clouds can be significantly densified and the occupancy labels

can be more realistic. Note that K-Radar [52] only annotates the objects in the front of the car. To avoid the interference of moving objects in the back, we only use the front part of each LiDAR sweep for ground truth generation. Second, we transform the background and objects point sets into the current frame coordinate system with respect to the ego-pose of the current frame and the objects' pose. Lastly, we concatenate the background and objects points at the current frame and voxelized the merged point cloud to generate the occupancy labels. In cases where voxels are overlaid or boundaries are not clear, we use the majority voting to decide voxel-wise semantics (foreground vs. background).

Dataset statistics. In adverse weather conditions (*e.g.,* fog, rain and snow), water droplets or snowflakes can scatter or absorb LiDAR beams, reducing the effective range of LiDAR and inducing noise in the data. To ensure the high fidelity of our occupancy labels, we select 24 sequences collected in decent weather conditions from K-Radar [52] for annotation and leave the rest sequences collected in poor weathers unannotated, which can only be used for qualitative analysis. We split the annotated 24 sequences into the training, validation and test sets with a ratio of 17:2:5, resulting in 11,333, 1,059 and 2,878 frames, respective. Over 0.5 billion voxels are obtained from all annotated frames, among which free, background and foreground class accounts for 92.3%, 7.4% and 0.3% individually.

Evaluation area. As claimed in the main chapter, we only report the evaluation results within the area where the horizontal FoV (hFov) of all sensors overlap. This scheme can lead to a more fair comparison as it avoids assessing the hallucinated voxels beyond hFoV for modalities like radar and camera, whose hFoVs cannot fully cover our defined RoI volume ahead of the car. Specifically, the overleap hFoV of K-Radar [52] sensor suite is 107°, symmetrically distributed around the front axis. The ratio between the final evaluation area and our RoI is calculated as: $1 - \cot\left(107^\circ/2\right)/4 \approx 0.812$.

Computation resources. All of our experiments are conducted on a Ubuntu server equipped with 2 Nvidia RTX 3090 - 24GB GPUs, an Intel i9-10980XE CPU @ 3.00GHz and a 64GB RAM. The training of our method `RadarOcc` uses 17.98GB VRAM, and takes approximately 16.7 hours.

License for K-Radar. The K-Radar dataset [52] is published under the CC BY-NC-ND License, and all K-Radar codes [1] are published under the Apache License 2.0.

[1] https://github.com/kaist-avelab/K-Radar

B.2 Implementation Details of RadarOcc

Data volume reduction. The volume size of input raw 4DRT $\mathbf{V}$ is $256 \times 107 \times 37 \times 64$ $(R \times A \times E \times D)$. By encoding the Doppler bins for each spatial location into 8-channel descriptors, we reduce the size of 4DRTs by $\times \frac{D}{8}$, leading to a 3D spatial data volume with a size of $(256 \times 107 \times 37) \times 8$ with the Doppler axis as the feature dimension. For sidelobe-aware spatial sparsifying, we select the top-N_r ($N_r = 250$) elements per range. The resulting lightweight sparse RT $\mathbf{T}$ per frame is $\sim$5MB. Please refer to Sec. B.3.1 for how we select the optimal N_r.

Range-wise self-attention. In our spherical-based feature encoding, the range-wise self-attention is performed on the non-empty elements per range, *i.e.,* $t_i \in \mathbb{R}^{N_r \times (8+2)} (i = 1, 2, \ldots, R)$, where $N_r = 250$. The 8-channel Doppler descriptors are considered as the input features while the azimuth and elevation indices are converted to positional embeddings with lookup tables [258]. Specifically, we use two layers of multi-head attention with the embedding dimension set as 32, number of heads as 4 and dropout probability to be 0.1 The output is re-organized to a sparse tensor with a dimension of $RN_r \times (32 + 3)$, where the range, azimuth and elevation index is stored for each non-empty element.

3D sparse convolution. We utilize the `spconv` library [278] to implement the sparse convolution layers for our spherical-based feature encoding. This encoding process has two types of operation: *3D Submanifold Convolution* and *3D Sparse Convolution*. 3D submanifold convolution only convolves the active spatial locations without altering the sparsity pattern and spatial resolution, while 3D sparse convolution performs convolution on all active locations, expanding the sparsify pattern and allows for spatial resolution change. We leverage the submanifold convolution as the first and last layer and apply three sparse convolution layers in-between. We set the stride as 2 for the last two of 3D sparse convolution to reduce the spatial dimension. As a result, we obtain a 3D dense feature volume $\mathbf{F} \in \mathbb{R}^{64 \times 27 \times 10 \times C_f} (C_f = 192)$, where the spatial dimension is decreased by $\times 4$.

Deformable self-attention. Given feature volume $\mathbf{F}$, we use 3D deformable self-attention [260] to augment its spatial features. Two attention layers are used and the number of sampling points of each query is set to 8. Each attention layer has 8 heads and apply a dropout of a rate of 0.1 to the output features. The refined feature volume $\mathbf{F}_r$ has the same dimension as $\mathbf{F}$, *i.e.,* $64 \times 27 \times 10 \times C_f$.

Spherical-to-Cartesian feature aggregation. To aggregate features extracted in the

spherical coordinates, we build learnable voxel queries $\mathbf{H} = \{h^q\}_q$ with a dimension of $14 \times 128 \times 128 \times C_f$ defined in the LiDAR Cartesian coordinates system. By transforming the 3D points p^q corresponding to our voxel queries h^q into the radar spherical coordinates, we construct a list of 3D reference points $\Phi(p^q)$. Then, the deformable cross attention is used to aggregate the spherical features to Cartesian by considering the spherical volume $\mathbf{F}_r$ as the key and value of attention and the voxel queries $\mathbf{H}$ as the query of the attention. Just as the self-attention module, we use the 3D version of the deformable attention [260], with the same network settings. The dimension of the output Cartesian feature $\mathbf{G}$ have the same size as the learnable queries $\mathbf{H}$, which is $14 \times 128 \times 128 \times C_f$.

3D occupancy decoding. Given the Cartesian voxel features $\mathbf{G}$, we firstly apply the 3D version of ResNet-18 [14] for decoding, resulting in 4 level of feature maps, with $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$ of the voxel spatial shape and $80, 160, 320, 640$ for feature dimension respectively. These multi-level features are then upsampled back to the target spatial space $H \times W \times L$ using 3D FPN [261], leading to the final features $\mathbf{G}_d$ with a dimension of $14 \times 128 \times 128 \times 4C_f$. Lastly, we use an MLP with the hidden dimension of [64,64] to reduce the feature channel and predict the occupancy probabilities which are normalized by a *softmax* layer. The output is denoted as $\tilde{\mathbf{O}} \in \{0,1\}^{H \times W \times L \times (C+1)}$.

Training loss. The overall loss function $\mathcal{L}$ used to train our network can be written as:

$$\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{LS} + \mathcal{L}_{scal}^{geo} + \mathcal{L}_{scal}^{sem} \tag{B.1}$$

Given the ground truth denoted as $\hat{\mathbf{O}} = \{\hat{o}_i \in \{c_0, c_1, \ldots, c_C\}\}_{i=1}^{N_o} (N_o = H \times W \times L)$ and the output $\tilde{\mathbf{O}}$, the cross-entropy loss $\mathcal{L}_{CE}$ can be calculated as:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{N_o} \sum_{c=c_0}^{c_C} w_c \hat{o}_{i,c} \log(\tilde{o}_{i,c}) \tag{B.2}$$

where N_o is the number of voxels, c and i indexes classes and voxels, $\tilde{o}_{i,c}$ is the predicted logit for i-th voxel on the class c. $\hat{o}_{i,c} = 1$ if $\hat{o}_i = c$; else, $\hat{o}_{i,c} = 0$. To balance different classes, we use w_c for each class calculated as the inverse of the class frequency in K-Radar [52]. Please refer to [262] and [96] for more details on the lovasz-softmax loss $\mathcal{L}_{LS}$ and scene-class affinity loss $\mathcal{L}_{scal}^{geo}$ and $\mathcal{L}_{scal}^{sem}$ we used in Eq. B.1.

Training details. We train RadarOcc with 10 epochs using Adam optimizer with a learning rate of 3e-4. The batch size is 1 for each GPU. We follow [85] to use loss normalization to balance the weight of the 4 different losses, and cosine annealing [279] with $\frac{1}{3}$ warm-up ratio is used at the start of the training.

	IoU (%)			mIoU (%)			BG IoU (%)			FG IoU (%)			fps
N_r	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	
850	-	-	-	-	-	-	-	-	-	-	-	-	CUDA OOM
650	52.5	43.9	30.6	34.4	**27.2**	19.7	52.1	43.7	30.4	16.7	**10.7**	8.9	2.9
450	53.9	44.3	30.9	**36.8**	26.9	**19.9**	53.7	44.0	30.6	**19.9**	9.7	**9.2**	3.1
250	**54.1**	**45.1**	**31.9**	34.0	25.7	19.1	**53.7**	**44.8**	**31.6**	14.2	6.7	6.6	3.3
50	52.7	44.5	31.9	32.6	25.8	19.4	52.6	44.3	31.5	12.5	7.3	7.3	3.6

Table B.1: Impact of the number of selected top elements per range (*i.e.*, N_r) in our sidelobe-aware spatial sparsifying. The results are reported on the validation set. Best result is shown in **bold**.

B.3 Additional Experiment Results

B.3.1 Impact of the Number of Reserved Top Elements N_r

In Sec. 6.1.3, we propose a sidelobe-aware spatial sparsification technique that selects the top-N_r elements for each individual range rather than the entire dense radar tensor (RT). There is indeed a trade-off between preserving critical measurements and filtering noise/compressing the radar tensor in this process. Excessive compression/filtering may result in the loss of weak reflections, while insufficient compression/filtering increases computational costs and retains some level of noise.

To identify the optimal balance, we conducted a series of experiments varying the number of selected top elements for each range, *i.e.*, N_r, and assessed performance and inference speed on the validation set. The results, presented in Table B.1, indicate that `RadarOcc` achieves the best results in half of all metrics on our validation set when N_r = 250. Both higher and lower values of lead to suboptimal results, suggesting that N_r = 250 strikes the best balance between retaining critical signals and filtering noise. Additionally, the inference speed at N_r = 250 is relatively higher compared to configurations with larger values. Therefore, we select N_r = 250 for `RadarOcc`'s evaluation on our testing set.

B.3.2 Impact of the Number of Reserved Doppler Bins N_d

To investigate the effect of the number of preserved top values (*i.e.,* N_d) among Doppler bins for each spatial location, we conducted a series of experiments by varying N_d. As shown in Table B.2, the change in N_d does not significantly impact our results. For both efficiency and performance, we chose $N_d = 3$ for our method based on the validation

N_d	IoU @ 51.2m (%)	mIoU @ 51.2m (%)
1	30.9	18.7
2	28.8	**19.4**
3	**31.9**	19.1
4	31.1	18.9
5	30.1	18.8

Table B.2: Impact of the number of reserved Doppler bins for each spatial location (*i.e.*, N_d). The results are reported on the validation set. Best result is shown in **bold**.

		IoU (%)			mIoU (%)			BG IoU (%)			FG IoU (%)		
	Method	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m	12.8m	25.6m	51.2m
(a)	Ours	**48.8**	**39.1**	30.4	**34.3**	**28.5**	**22.6**	**47.9**	**38.2**	29.4	**20.7**	**18.7**	**15.8**
(b)	Ours w/o RWA	48.6	39.0	**30.7**	32.8	27.7	22.0	47.4	38.0	**29.6**	18.1	16.3	14.2

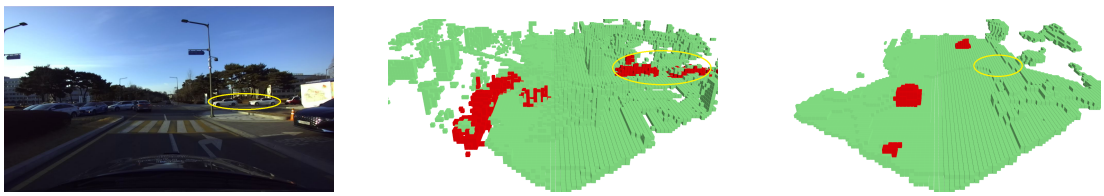
Table B.3: Ablation studies on range-wise self-attention designs of `RadarOcc`.

set performance.

This can be explained by the fact that K-Radar [52] wraps around overflow values in Doppler measurements due to the limited Doppler measurement range. For example, Doppler speeds of 3.0 m/s and 6.0 m/s are measured within the range of -1.92 to 1.92 m/s as 3.0 - 3.84 = -0.84m/s and 6.0 - 3.84$\times$2 = -1.68m/s, respectively. This ambiguity means the information from the Doppler axis only marginally improves our model. Consequently, changing hardly affects our performance. Table. 6.2 in **??** also shows that our baseline without Doppler bin descriptor (w/o DBD), which only uses mean power, reflects this minimal impact. However, we believe our Doppler bin encoding method could bring more improvement with other radar sensors that have a larger measurement range.

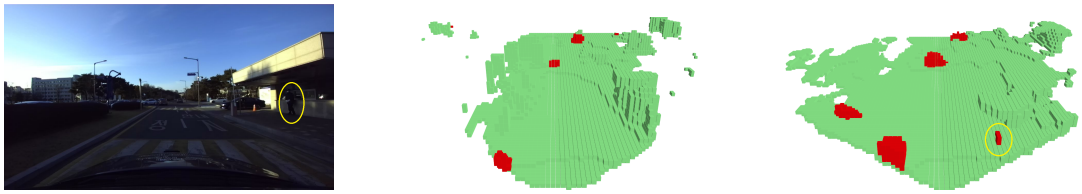
B.3.3 Impact of Range-wise Self-attention

To verify the effectiveness of the range-wise self-attention used in our sphercial-based feature encoding (*c.f.* Sec. 6.1.4), we experiment by removing it from the network and show the results in Tab. B.3. It can be seen that `RadarOcc` improves the performance on most metrics by adding the range-wise self-attention. In particular on FG IoU, the relative gain is 14.4%, 14.7% and 11.3% for the 3D volume of 12.8m, 25.6m and 51.2m, respectively. We credit this to the ability of range-wise self-attention to further suppress the sidelobe noises appearing around the foreground objects.



(a) Camera View (b) Ground truth (c) RadarOcc prediction

Figure B.1: Example of failure case due to insufficient resolution and decreased Signal-to-Noise Ratio (SNR) at far distances. The white cars parked at the far right are not well predicted.



(a) Camera View (b) 32 line LiDAR prediction (c) RadarOcc prediction

Figure B.2: Example of `RadarOcc` outperforming 32-line LiDAR on objects with low radar cross-section: the pedestrain is recognized.

B.3.4 Qualitative Results under Adverse Weather

To better show the qualitative results of `RadarOcc` and baseline methods based on other modalities, we make some video demos under different weather conditions and integrate them into a demo video: `https://www.youtube.com/watch?v=sa4gejcWMvk`. We recommend our audience to watch the video for a better understanding of our work.

B.3.5 Example of Failure Cases

We observed some failure cases of `RadarOcc` due to some reasons, such as insufficient resolution and decreased Signal-to-Noise Ratio (SNR) at far distances. An example of such failure cases is exhibited in Fig. B.1. We hope this could shed the light on future research in this field.

B.3.6 Object with Low Radar Cross-section

In our method, we address objects with low radar cross-section (RCS) from two key perspectives:

Input perspective. We utilize 4D radar tensor (4DRT) data instead of radar point clouds

for 3D occupancy prediction. This approach avoids the loss of weak signal returns that can occur during the point cloud generation process, *e.g.*, those filtered out by the CFAR detection, preserving more measurements from low RCS objects compared to radar point clouds.

Method perspective. Our sidelobe-aware spatial sparsifying technique selects the top-elements for each individual range rather than the entire dense RT. As shown in Fig. 6.2, this method retains critical measurements scattered across different ranges, including both low and high RCS objects. This contrasts with percentile-based methods, which often concentrate on elements corresponding to high RCS objects, thereby missing important data from low RCS objects.

As a result, our method is effective in recognizing objects with low RCS, such as pedestrians, when predicting 3D occupancy. Figure B.2 shows an example where `RadarOcc` successfully handles low-RCS objects while 32-line LiDAR not.

Bibliography

[1] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*, volume 200. Springer, 2008.

[2] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[3] European Commission. On the road to automated mobility: An EU strategy for mobility of the future, 2018. [Online; accessed 19-January-2025].

[4] Daniel J Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[5] Eric Krotkov, Douglas Hackett, Larry Jackel, Michael Perschbacher, James Pippine, Jesse Strauss, Gill Pratt, and Christopher Orlowski. The darpa robotics challenge finals: Results and perspectives. *The DARPA robotics challenge finals: Humanoid robots to the rescue*, pages 1–26, 2018.

[6] Ravi Raj and Andrzej Kos. A comprehensive study of mobile robot: History, developments, applications, and future research perspectives. *Applied Sciences*, 12(14):6951, 2022.

[7] Mary B Alatise and Gerhard P Hancke. A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846, 2020.

[8] Lars Kunze, Nick Hawes, Tom Duckett, Marc Hanheide, and Tomáš Krajník. Artificial intelligence for long-term robot autonomy: A survey. *IEEE Robotics and Automation Letters*, 3(4):4023–4030, 2018.

[9] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):187–210, 2018.

[10] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816, 2021.

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2016.

[12] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III*, pages 234–241. Springer, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[17] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 11621–11631, 2020.

[18] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2446–2454, 2020.

[19] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*, 2023.

[20] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 954–960, 2018.

[21] Ke Wang, Sai Ma, Junlan Chen, Fan Ren, and Jianbo Lu. Approaches, challenges, and applications for deep visual odometry: Toward complicated and emerging areas. *IEEE Transactions on Cognitive and Developmental Systems*, 14(1):35–49, 2020.

[22] Dongjae Lee, Minwoo Jung, Wooseong Yang, and Ayoung Kim. LiDAR odometry survey: Recent advancements and remaining challenges. *Intelligent Service Robotics*, 17(2):95–118, 2024.

[23] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, 2021.

[24] Hamid Laga, Laurent Valentin Jospin, Farid Boussaid, and Mohammed Bennamoun. A survey on deep learning techniques for stereo-based depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):1738–1764, 2020.

[25] Xinzhu Ma, Wanli Ouyang, Andrea Simonelli, and Elisa Ricci. 3d object detection from images for autonomous driving: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[26] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision*, 131(8):1909–1963, 2023.

[27] Peng Zhang, Xin Li, Liang He, and Xin Lin. 3d multiple object tracking on autonomous driving: A literature review. *arXiv preprint arXiv:2309.15411*, 2023.

[28] Diego M Jiménez-Bravo, Álvaro Lozano Murciego, André Sales Mendes, Héctor Sánchez San Blás, and Javier Bajo. Multi-object tracking in traffic environments: A systematic literature review. *Neurocomputing*, 494:43–55, 2022.

[29] Yuxing Xie, Jiaojiao Tian, and Xiao Xiang Zhu. Linking points with labels in 3d: A review of point cloud semantic segmentation. *IEEE Geoscience and Remote Sensing Magazine*, 8(4):38–59, 2020.

[30] Siddiqui Muhammad Yasir and Hyunsik Ahn. Deep learning-based 3d instance and semantic segmentation: A review. *arXiv preprint arXiv:2406.13308*, 2024.

[31] Zhiqi Li, Nan Xiang, Honghua Chen, Jianjun Zhang, and Xiaosong Yang. Deep learning for scene flow estimation on point clouds: A survey and prospective trends. In *Computer Graphics Forum*, volume 42, page e14795. Wiley Online Library, 2023.

[32] Sundaram Muthu, Ruwan Tennakoon, Reza Hoseinnezhad, and Alireza Bab-Hadiashar. A survey of CNN-based techniques for scene flow estimation. *IEEE Access*, 2023.

[33] Huaiyuan Xu, Junliang Chen, Shiyu Meng, Yi Wang, and Lap-Pui Chau. A survey on occupancy perception for autonomous driving: The information fusion perspective. *Information Fusion*, 114:102671, 2025.

[34] Yanan Zhang, Jinqing Zhang, Zengran Wang, Junhao Xu, and Di Huang. Vision-based 3d occupancy prediction in autonomous driving: A review and outlook. *arXiv preprint arXiv:2405.02595*, 2024.

[35] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[36] Li-Hua Wen and Kang-Hyun Jo. Deep learning-based perception systems for autonomous driving: A comprehensive survey. *Neurocomputing*, 489:255–270, 2022.

[37] Mario Bijelic, Tobias Gruber, and Werner Ritter. Benchmarking image sensors under adverse weather conditions for autonomous driving. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 1773–1779. IEEE, 2018.

[38] Yuxiao Zhang, Alexander Carballo, Hanting Yang, and Kazuya Takeda. Perception and sensing for autonomous vehicles under adverse weather conditions: A survey. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:146–177, 2023.

[39] Dario Floreano and Claudio Mattiussi. *Bio-inspired artificial intelligence: Theories, methods, and technologies*. MIT Press, 2008.

[40] Yoseph Bar-Cohen. *Biomimetics: Nature-based innovation*. CRC Press, 2016.

[41] Gareth Jones and Marc W Holderied. Bat echolocation calls: adaptation and convergent evolution. *Proceedings of the Royal Society B: Biological Sciences*, 274(1612):905–912, 2007.

[42] Kyle C Newton, Andrew B Gill, and Stephen M Kajiura. Electroreception in marine fishes: Chondrichthyans. *Journal of Fish Biology*, 95(1):135–154, 2019.

[43] Jürgen Hasch, Eray Topak, Raik Schnabel, Thomas Zwick, Robert Weigel, and Christian Waldschmidt. Millimeter-wave technology for automotive radar sensors in the 77 GHz frequency band. *IEEE Transactions on Microwave Theory and Techniques*, 60(3):845–860, 2012.

[44] Sandeep Rao. Introduction to mmWave sensing: FMCW radars. *Texas Instruments (TI) mmWave Training Series*, pages 1–11, 2017.

[45] Shunqiao Sun and Yimin D Zhang. 4D automotive radar sensing for autonomous vehicles: A sparsity-oriented approach. *IEEE Journal of Selected Topics in Signal Processing*, 15(4):879–891, 2021.

[46] Lili Fan, Junhao Wang, Yuanmeng Chang, Yuke Li, Yutong Wang, and Dongpu Cao. 4d mmwave radar for autonomous driving perception: A comprehensive survey. *IEEE Transactions on Intelligent Vehicles*, 2024.

[47] Texas Instruments. mmWave Radar Sensors – Overview. `https://www.ti.com/sensors/mmwave-radar/overview.html`, 2024. Accessed: 2024-02-22.

[48] Andrew Kramer, Kyle Harlow, Christopher Williams, and Christoffer Heckman. ColoRadar: The direct 3D millimeter wave radar dataset. *The International Journal of Robotics Research*, 41(4):351–360, 2022.

[49] Louis L Scharf and Cédric Demeure. *Statistical signal processing: Detection, estimation, and time series analysis*. Prentice Hall, 1991.

[50] Stephen Blake. OS-CFAR theory for multiple targets and nonuniform clutter. *IEEE Transactions on Aerospace and Electronic Systems*, 24(6):785–790, 1988.

[51] Yuwei Cheng, Jingran Su, Mengxin Jiang, and Yimin Liu. A novel radar point cloud generation method for robot environment perception. *IEEE Transactions on Robotics*, 38(6):3754–3773, 2022.

[52] Dong-Hee Paek, Seung-Hyun Kong, and Kevin Tirta Wijaya. K-radar: 4D radar object detection for autonomous driving in various weather conditions. *Advances in Neural Information Processing Systems*, 35:3819–3829, 2022.

[53] Dong-Hee Paek, Seung-Hyun Kong, and Kevin Tirta Wijaya. Enhanced k-radar: Optimal density reduction to improve detection performance and accessibility of 4D radar tensor-based object detection. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 1–6. IEEE, 2023.

[54] David K Barton. *Radar system analysis and modeling*. Artech House, 2004.

[55] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning scene flow in 3D point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019.

[56] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Fuxin Li. PointPWC-Net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 88–107, 2020.

[57] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. FlowStep3D: Model unrolling for self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4114–4123, 2021.

[58] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5692–5703, 2021.

[59] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11177–11185, 2020.

[60] Ruibo Li, Guosheng Lin, and Lihua Xie. Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15577–15586, 2021.

[61] Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Björn Ommer, and Andreas Geiger. SLIM: Self-supervised LiDAR scene flow and motion segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 13126–13136, 2021.

[62] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. TNT: Target-driven trajectory prediction. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 895–904. PMLR, 2021.

[63] Nachiket Deo, Eric Wolff, and Oscar Beijbom. Multimodal trajectory prediction conditioned on lane-graph traversals. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 203–212. PMLR, 2022.

[64] Minghan Wei and Volkan Isler. Energy-efficient path planning for ground robots by and combining air and ground measurements. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 766–775. PMLR, 2020.

[65] Anjian Li, Liting Sun, Wei Zhan, Masayoshi Tomizuka, and Mo Chen. Prediction-based reachability for collision avoidance in autonomous driving. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7908–7914. IEEE, 2021.

[66] Jiahao Lin, Hai Zhu, and Javier Alonso-Mora. Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments. In *Proceedings of*

the IEEE International Conference on Robotics and Automation (ICRA), pages 2682–2688. IEEE, 2020.

[67] Hui Zhang, Hongzhe Jin, Zhangxing Liu, Yubin Liu, Yanhe Zhu, and Jie Zhao. Real-time kinematic control for redundant manipulators in a time-varying environment: Multiple-dynamic obstacle avoidance and fast tracking of a moving object. *IEEE Transactions on Industrial Informatics*, 16(1):28–41, 2019.

[68] Clément Petres, Yan Pailhas, Pedro Patron, Yvan Petillot, Jonathan Evans, and David Lane. Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, 23(2):331–341, 2007.

[69] Ryo Yonetani, Tatsunori Taniai, Mohammadamin Barekatain, Mai Nishimura, and Asako Kanezaki. Path planning using neural A* search. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 12029–12039. PMLR, 2021.

[70] Hiroaki Inotsume, Takashi Kubota, and David Wettergreen. Robust path planning for slope traversing under uncertainty in slip prediction. *IEEE Robotics and Automation Letters*, 5(2):3390–3397, 2020.

[71] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D multi-object tracking: A baseline and new evaluation metrics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366, 2020.

[72] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3D object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11784–11793, 2021.

[73] Hsu-Kuang Chiu, Jie Li, Rareş Ambruş, and Jeannette Bohg. Probabilistic 3D multi-modal, multi-object tracking for autonomous driving. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 14227–14233. IEEE, 2021.

[74] Tobias Fischer, Yung-Hsu Yang, Suryansh Kumar, Min Sun, and Fisher Yu. CC-3DT: Panoramic 3D object tracking via cross-camera fusion. In *Proceedings of the Conference on Robot Learning (CoRL)*. PMLR, 2022.

[75] Hou-Ning Hu, Yung-Hsu Yang, Tobias Fischer, Trevor Darrell, Fisher Yu, and Min Sun. Monocular quasi-dense 3D object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1992–2008, 2022.

[76] Fangqiang Ding, Changhong Fu, Yiming Li, Jin Jin, and Chen Feng. Automatic failure recovery and re-initialization for online UAV tracking with joint scale and aspect ratio optimization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5970–5977. IEEE, 2020.

[77] Abhijeet Shenoi, Mihir Patel, JunYoung Gwak, Patrick Goebel, Amir Sadeghian, Hamid Rezatofighi, Roberto Martin-Martin, and Silvio Savarese. JRMOT: A real-time 3D multi-object tracker and a new large-scale dataset. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10335–10342. IEEE, 2020.

[78] Aleksandr Kim, Aljoša Ošep, and Laura Leal-Taixé. EagerMOT: 3D multi-object tracking via sensor fusion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 11315–11321. IEEE, 2021.

[79] Yihan Zeng, Chao Ma, Ming Zhu, Zhiming Fan, and Xiaokang Yang. Cross-modal 3D object detection and tracking for auto-driving. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3850–3857. IEEE, 2021.

[80] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. PnPNet: End-to-end perception and prediction with tracking in the loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11553–11562, 2020.

[81] Andras Palffy, Ewoud Pool, Srimannarayana Baratam, Julian F. P. Kooij, and Dariu M. Gavrila. Multi-class road user detection with 3+1D radar in the view-of-delft dataset. *IEEE Robotics and Automation Letters*, 7(2):4961–4968, 2022.

[82] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[83] Tesla. Tesla AI Day 2022. https://www.youtube.com/watch?v=ODSJsviD_SU, 2022. Accessed: 2024-04-08.

[84] Ben Agro, Quinlan Sykora, Sergio Casas, and Raquel Urtasun. Implicit occupancy flow fields for perception and prediction in self-driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1379–1388, 2023.

[85] Xiaofeng Wang, Zheng Zhu, Wenbo Xu, Yunpeng Zhang, Yi Wei, Xu Chi, Yun Ye, Dalong Du, Jiwen Lu, and Xingang Wang. Openoccupancy: A large scale benchmark for surrounding semantic occupancy perception. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 17850–17859, 2023.

[86] Xiaoyu Tian, Tao Jiang, Longfei Yun, Yucheng Mao, Huitong Yang, Yue Wang, Yilun Wang, and Hang Zhao. Occ3d: A large-scale 3D occupancy prediction benchmark for autonomous driving. *Advances in Neural Information Processing Systems*, 36, 2024.

[87] Yiming Li, Zhiding Yu, Christopher Choy, Chaowei Xiao, Jose M Alvarez, Sanja Fidler, Chen Feng, and Anima Anandkumar. VoxFormer: Sparse voxel transformer for camera-based 3D semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9087–9098, 2023.

[88] Ran Cheng, Christopher Agia, Yuan Ren, Xinhai Li, and Liu Bingbing. S3CNet: A sparse semantic scene completion network for LiDAR point clouds. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 2148–2161. PMLR, 2021.

[89] Luis Roldao, Raoul de Charette, and Anne Verroust-Blondet. LMSCNet: Lightweight multiscale 3D semantic completion. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 111–119, 2020.

[90] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep LiDAR point cloud segmentation via learning contextual shape priors from scene completion. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 3101–3109, 2021.

[91] Pengfei Li, Yongliang Shi, Tianyu Liu, Hao Zhao, Guyue Zhou, and Ya-Qin Zhang. Semi-supervised implicit scene completion from sparse LiDAR. *arXiv preprint arXiv:2111.14798*, 2021.

[92] Christoph B Rist, David Emmerichs, Markus Enzweiler, and Dariu M Gavrila. Semantic scene completion using local deep implicit functions on LiDAR data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7205–7218, 2021.

[93] Xinhao Liu, Moonjun Gong, Qi Fang, Haoyu Xie, Yiming Li, Hang Zhao, and Chen Feng. LiDAR-based 4D occupancy completion and forecasting. *arXiv preprint arXiv:2310.11239*, 2023.

[94] Zhaoyang Xia, Youquan Liu, Xin Li, Xinge Zhu, Yuexin Ma, Yikang Li, Yuenan Hou, and Yu Qiao. SCPNet: Semantic scene completion on point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17642–17651, 2023.

[95] Tarasha Khurana, Peiyun Hu, David Held, and Deva Ramanan. Point cloud forecasting as a proxy for 4D occupancy forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1116–1124, 2023.

[96] Anh-Quan Cao and Raoul De Charette. MonoScene: Monocular 3D semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3991–4001, 2022.

[97] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Tri-perspective view for vision-based 3D semantic occupancy prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9223–9232, 2023.

[98] Yi Wei, Linqing Zhao, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. Surroundocc: Multi-camera 3D occupancy prediction for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 21729–21740, 2023.

[99] Wenwen Tong, Chonghao Sima, Tai Wang, Li Chen, Silei Wu, Hanming Deng, Yi Gu, Lewei Lu, Ping Luo, Dahua Lin, et al. Scene as occupancy. In *Proceedings*

of the IEEE International Conference on Computer Vision (ICCV), pages 8406–8415, 2023.

[100] Yunpeng Zhang, Zheng Zhu, and Dalong Du. Occformer: Dual-path transformer for vision-based 3D semantic occupancy prediction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9433–9443, 2023.

[101] Zhiyu Tan, Zichao Dong, Cheng Zhang, Weikun Zhang, Hang Ji, and Hao Li. OVO: Open-vocabulary occupancy. *arXiv preprint arXiv:2305.16133*, 2023.

[102] Yuanhui Huang, Wenzhao Zheng, Borui Zhang, Jie Zhou, and Jiwen Lu. Self-occ: Self-supervised vision-based 3D occupancy prediction. *arXiv preprint arXiv:2311.12754*, 2023.

[103] Chubin Zhang, Juncheng Yan, Yi Wei, Jiaxin Li, Li Liu, Yansong Tang, Yueqi Duan, and Jiwen Lu. Occnerf: Self-supervised multi-camera occupancy prediction with neural radiance fields. *arXiv preprint arXiv:2312.09243*, 2023.

[104] Antonin Vobecky, Oriane Siméoni, David Hurych, Spyridon Gidaris, Andrei Bursuc, Patrick Pérez, and Josef Sivic. POP-3D: Open-vocabulary 3D occupancy prediction from images. *Advances in Neural Information Processing Systems*, 36, 2024.

[105] Qihang Ma, Xin Tan, Yanyun Qu, Lizhuang Ma, Zhizhong Zhang, and Yuan Xie. COTR: Compact occupancy transformer for vision-based 3D occupancy prediction. *arXiv preprint arXiv:2312.01919*, 2023.

[106] Junyi Ma, Xieyuanli Chen, Jiawei Huang, Jingyi Xu, Zhen Luo, Jintao Xu, Weihao Gu, Rui Ai, and Hesheng Wang. Cam4DOcc: Benchmark for camera-only 4D occupancy forecasting in autonomous driving applications. *arXiv preprint arXiv:2311.17663*, 2023.

[107] Shunqiao Sun, Athina P Petropulu, and H Vincent Poor. MIMO radar for advanced driver-assistance systems and autonomous driving: Advantages and challenges. *IEEE Signal Processing Magazine*, 37(4):98–117, 2020.

[108] Prashant P Gandhi and Saleem A Kassam. Analysis of CFAR processors in nonhomogeneous background. *IEEE Transactions on Aerospace and Electronic Systems*, 24(4):427–445, 1988.

[109] Ziqi Pang, Zhichao Li, and Naiyan Wang. SimpleTrack: Understanding and rethinking 3D multi-object tracking. *arXiv preprint arXiv:2111.09621*, 2021.

[110] Aleksandr Kim, Guillem Brasó, Aljoša Ošep, and Laura Leal-Taixé. PolarMOT: How far can geometric relations take us in 3D multi-object tracking? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 41–58. Springer, 2022.

[111] Michael Meyer and Georg Kuschk. Automotive radar dataset for deep learning based 3D object detection. In *Proceedings of the European Radar Conference (EuRAD)*, pages 129–132, 2019.

[112] Julien Rebut, Arthur Ouaknine, Waqas Malik, and Patrick Pérez. Raw high-definition radar for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17021–17030, June 2022.

[113] Lianqing Zheng, Zhixiong Ma, Xichan Zhu, Bin Tan, Sen Li, Kai Long, Weiqi Sun, Sihan Chen, Lu Zhang, Mengyue Wan, et al. TJ4DRadSet: A 4D radar dataset for autonomous driving. In *Proceedings of the IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 493–498. IEEE, 2022.

[114] Minseong Choi, Seunghoon Yang, Seungho Han, Yeongseok Lee, Minyoung Lee, Keun Ha Choi, and Kyung-Soo Kim. MSC-RAD4R: ROS-based automotive dataset with 4D radar. *IEEE Robotics and Automation Letters*, 2023.

[115] Jianzhu Huai, Binliang Wang, Yuan Zhuang, Yiwen Chen, Qipeng Li, and Yulong Han. Snail radar: A large-scale diverse benchmark for evaluating 4d-radar-based slam. *The International Journal of Robotics Research*, page 02783649251329048, 2024.

[116] Yuwei Cheng, Jingran Su, Hongyu Chen, and Yimin Liu. A new automotive radar 4d point clouds detector by using deep learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8398–8402. IEEE, 2021.

[117] Baowei Xu, Xinyu Zhang, Li Wang, Xiaomei Hu, Zhiwei Li, Shuyue Pan, Jun Li, and Yongqiang Deng. RPFA-Net: A 4D radar pillar feature attention network

for 3D object detection. In *Proceedings of the IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3061–3066. IEEE, 2021.

[118] Jianan Liu, Qiuchi Zhao, Weiyi Xiong, Tao Huang, Qing-Long Han, and Bing Zhu. SMURF: Spatial multi-representation fusion for 3D object detection with 4D imaging radar. *IEEE Transactions on Intelligent Vehicles*, 2023.

[119] Zhijun Pan, Fangqiang Ding, Hantao Zhong, and Chris Xiaoxuan Lu. Moving object detection and tracking with 4D radar point cloud. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[120] Seung-Hyun Kong, Dong-Hee Paek, and Sangjae Cho. RTNH+: Enhanced 4D radar object detection network using combined CFAR-based two-level preprocessing and vertical encoding. *arXiv preprint arXiv:2310.17659*, 2023.

[121] Shouyi Lu, Guirong Zhuo, Lu Xiong, Xichan Zhu, Lianqing Zheng, Zihang He, Mingyu Zhou, Xinfei Lu, and Jie Bai. Efficient deep-learning 4D automotive radar odometry method. *IEEE Transactions on Intelligent Vehicles*, 2023.

[122] Jun Zhang, Huayang Zhuge, Zhenyu Wu, Guohao Peng, Mingxing Wen, Yiyao Liu, and Danwei Wang. 4DRadarSLAM: A 4D imaging radar SLAM system for large-scale environments based on pose graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 8333–8340. IEEE, 2023.

[123] Guohao Peng, Heshan Li, Yangyang Zhao, Jun Zhang, Zhenyu Wu, Pengyu Zheng, and Danwei Wang. Transloc4d: Transformer-based 4d radar place recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17595–17605, 2024.

[124] Fangqiang Ding, Zhijun Pan, Yimin Deng, Jianning Deng, and Chris Xiaoxuan Lu. Self-supervised scene flow estimation with 4D automotive radar. *IEEE Robotics and Automation Letters*, pages 1–8, 2022.

[125] Fangqiang Ding, Andras Palffy, Dariu M. Gavrila, and Chris Xiaoxuan Lu. Hidden gems: 4D radar scene flow learning using cross-modal supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–10, 2023.

[126] Fangqiang Ding, Xiangyu Wen, Yunzhou Zhu, Yiming Li, and Chris Xiaoxuan Lu. Radarocc: Robust 3d occupancy prediction with 4d imaging radar. *Advances in Neural Information Processing Systems*, 37:101589–101617, 2024.

[127] Yang Liu, Feng Wang, Naiyan Wang, and Zhao-Xiang Zhang. Echoes beyond points: Unleashing the power of raw radar data in multi-modality fusion. *Advances in Neural Information Processing Systems*, 36, 2024.

[128] Bin Tan, Zhixiong Ma, Xichan Zhu, Sen Li, Lianqing Zheng, Sihan Chen, Libo Huang, and Jie Bai. 3D object detection for multi-frame 4D automotive millimeter-wave radar point cloud. *IEEE Sensors Journal*, 2022.

[129] Michael Meyer and Georg Kuschk. Deep learning based 3D object detection for automotive radar and camera. In *Proceedings of the 16th European Radar Conference (EuRAD)*, pages 133–136. IEEE, 2019.

[130] Li Wang, Xinyu Zhang, Baowei Xv, Jinzhao Zhang, Rong Fu, Xiaoyu Wang, Lei Zhu, Haibing Ren, Pingping Lu, Jun Li, et al. Interfusion: Interaction-based 4D radar and LiDAR fusion for 3D object detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12247–12253. IEEE, 2022.

[131] Li Wang, Xinyu Zhang, Jun Li, Baowei Xv, Rong Fu, Haifeng Chen, Lei Yang, Dafeng Jin, and Lijun Zhao. Multi-modal and multi-scale fusion 3D object detection of 4D radar and LiDAR for autonomous driving. *IEEE Transactions on Vehicular Technology*, 2022.

[132] Lianqing Zheng, Sen Li, Bin Tan, Long Yang, Sihan Chen, Libo Huang, Jie Bai, Xichan Zhu, and Zhixiong Ma. RCFusion: Fusing 4D radar and camera with bird's-eye view features for 3D object detection. *IEEE Transactions on Instrumentation and Measurement*, 2023.

[133] Weiyi Xiong, Jianan Liu, Tao Huang, Qing-Long Han, Yuxuan Xia, and Bing Zhu. LXL: LiDAR-excluded lean 3D object detection with 4D imaging radar and camera fusion. *IEEE Transactions on Intelligent Vehicles*, 2023.

[134] Haoyi Zhang, Kai Wu, Rongkang Chen, Zihao Wu, Yong Zhong, and Weihua Li. TL-4DRCF: A two-level 4D radar-camera fusion method for object detection in adverse weather. *IEEE Sensors Journal*, 2024.

[135] Qiao Yan and Yihan Wang. MVFAN: Multi-view feature assisted network for 4D radar object detection. In *Proceedings of the International Conference on Neural Information Processing (ICONIP)*, pages 493–511. Springer, 2023.

[136] Xinyu Zhang, Li Wang, Jian Chen, Cheng Fang, Lei Yang, Ziying Song, Guangqi Yang, Yichen Wang, Xiaofei Zhang, and Jun Li. Dual radar: A multi-modal dataset with dual 4D radar for autonomous driving. *arXiv preprint arXiv:2310.07602*, 2023.

[137] Jianning Deng, Gabriel Chan, Hantao Zhong, and Chris Xiaoxuan Lu. See beyond seeing: Robust 3D object detection from point clouds via cross-modal hallucination. *arXiv preprint arXiv:2309.17336*, 2023.

[138] Hang Cui, Junzhe Wu, Jiaming Zhang, Girish Chowdhary, and William R. Norris. 3D detection and tracking for on-road vehicles with a monovision camera and dual low-cost 4D mmWave radars. In *Proceedings of the IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2931–2937. IEEE, 2021.

[139] Jun Zhang, Huayang Zhuge, Yiyao Liu, Guohao Peng, Zhenyu Wu, Haoyuan Zhang, Qiyang Lyu, Heshan Li, Chunyang Zhao, Dogan Kircali, et al. NTU4DRadLM: 4D radar-centric multi-modal dataset for localization and mapping. In *Proceedings of the IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 4291–4296. IEEE, 2023.

[140] Guirong Zhuoins, Shouyi Lu, Lu Xiong, Huanyu Zhouins, Lianqing Zheng, and Mingyu Zhou. 4DRVO-Net: Deep 4D radar-visual odometry using multi-modal and multi-scale adaptive fusion. *IEEE Transactions on Intelligent Vehicles*, 2023.

[141] Yuan Zhuang, Binliang Wang, Jianzhu Huai, and Miao Li. 4D IRIOM: 4D imaging radar inertial odometry and mapping. *IEEE Robotics and Automation Letters*, 2023.

[142] Xingyi Li, Han Zhang, and Weidong Chen. 4D radar-based pose graph SLAM with ego-velocity pre-integration factor. *IEEE Robotics and Automation Letters*, 2023.

[143] Zeyu Han, Jiahao Wang, Zikun Xu, Shuocheng Yang, Lei He, Shaobing Xu, Jianqiang Wang, and Keqiang Li. 4d millimeter-wave radar in autonomous driving: A survey. *arXiv preprint arXiv:2306.04242*, 2023.

[144] Xiangyuan Peng, Miao Tang, Huawei Sun, Kay Bierzynski, Lorenzo Servadei, and Robert Wille. 4d mmwave radar for sensing enhancement in adverse environments: Advances and challenges. *arXiv preprint arXiv:2503.24091*, 2025.

[145] Sundar Vedula, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480, 2005.

[146] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–7, 2007.

[147] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 739–751, 2008.

[148] Jan Čech, Jordi Sanchez-Riera, and Radu Horaud. Scene flow estimation by growing correspondence seeds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3129–3136, 2011.

[149] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1377–1384, 2013.

[150] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. SphereFlow: 6 DoF scene flow from RGB-D pairs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3526–3533, 2014.

[151] Julian Quiroga, Frédéric Devernay, and James Crowley. Local/global scene flow estimation. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 3850–3854, 2013.

[152] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James Crowley. Dense semi-rigid scene flow estimation from RGBD images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 567–582, 2014.

[153] Simon Hadfield and Richard Bowden. Kinecting the dots: Particle based scene flow from depth sensors. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2290–2295, 2011.

[154] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.

[155] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 614–630, 2018.

[156] Zhaoyang Lv, Kihwan Kim, Alejandro Troccoli, Deqing Sun, James M Rehg, and Jan Kautz. Learning rigidity in dynamic scenes with a moving camera for 3D motion field estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 468–484, 2018.

[157] Huaizu Jiang, Deqing Sun, Varun Jampani, Zhaoyang Lv, Erik Learned-Miller, and Jan Kautz. SENSE: A shared encoder network for scene-flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3195–3204, 2019.

[158] Yi-Ling Qiao, Lin Gao, Yu-Kun Lai, Fang-Lue Zhang, Mingzhe Yuan, and Shihong Xia. SF-Net: Learning scene flow from RGB-D images with CNNs. In *Proceedings of the British Machine Vision Conference (BMVC)*, page 281, 2018.

[159] Liang Liu, Guangyao Zhai, Wenlong Ye, and Yong Liu. Unsupervised learning of scene flow estimation fusing with local rigidity. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 876–882, 2019.

[160] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1983–1992, 2018.

[161] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 36–53, 2018.

[162] Junhwa Hur and Stefan Roth. Self-supervised monocular scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7396–7405, 2020.

[163] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Rigid scene flow for 3d LiDAR scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1765–1770, 2016.

[164] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 261–270, 2020.

[165] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Advances in Neural Information Processing Systems*, 34:7838–7851, 2021.

[166] Yancong Lin and Holger Caesar. Icp-flow: Lidar scene flow estimation with icp. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15501–15511, 2024.

[167] Xueqian Li, Jianqiao Zheng, Francesco Ferroni, Jhony Kaesemodel Pontes, and Simon Lucey. Fast neural scene flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9878–9890, 2023.

[168] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.

[169] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3D point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630, 2019.

[170] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2530–2539, 2018.

[171] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7962–7971, 2019.

[172] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. HPLFlowNet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3254–3263, 2019.

[173] Ruibo Li, Chi Zhang, Guosheng Lin, Zhe Wang, and Chunhua Shen. RigidFlow: Self-supervised scene flow learning on point clouds by local rigidity prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16959–16968, 2022.

[174] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene flow on point clouds guided by optimal transport. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 527–544, 2020.

[175] Yi Wei, Ziyi Wang, Yongming Rao, Jiwen Lu, and Jie Zhou. PV-RAFT: Point-voxel correlation fields for scene flow estimation of point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6954–6963, 2021.

[176] Guanting Dong, Yueyi Zhang, Hanlin Li, Xiaoyan Sun, and Zhiwei Xiong. Exploiting rigidity constraints for LiDAR scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12776–12785, 2022.

[177] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. FlowNet3D++: Geometric losses for deep scene flow estimation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 91–98, 2020.

[178] Haiyan Wang, Jiahao Pang, Muhammad A Lodhi, Yingli Tian, and Dong Tian. FESTA: Flow estimation via spatial-temporal attention for scene point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14173–14182, 2021.

[179] Ruibo Li, Guosheng Lin, Tong He, Fayao Liu, and Chunhua Shen. HCRF-Flow: Scene flow from point clouds with continuous high-order CRFs and position-aware flow embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 364–373, 2021.

[180] Victor Zuanazzi, Joris van Vugt, Olaf Booij, and Pascal Mettes. Adversarial self-supervised scene flow estimation. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 1049–1058, 2020.

[181] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:60–76, 2018.

[182] Ivan Tishchenko, Sandro Lombardi, Martin R Oswald, and Marc Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 150–159, 2020.

[183] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[184] Fangqiang Ding, Zhijun Pan, Yimin Deng, Jianning Deng, and Chris Xiaoxuan Lu. Self-supervised scene flow estimation with 4D automotive radar. *IEEE Robotics and Automation Letters*, 7(3):8233–8240, 2022.

[185] Mingliang Zhai, Bing-Kun Bao, and Xuezhi Xiang. Dmrflow: 4d radar scene flow estimation with decoupled matching and refinement. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

[186] Jialong Wu, Marco Braun, Dominic Spata, and Matthias Rottmann. Tars: Traffic-aware radar scene flow estimation. *arXiv preprint arXiv:2503.10210*, 2025.

[187] Yufei Liu, Xieyuanli Chen, Neng Wang, Stepan Andreev, Alexander Dvorkovich, Rui Fan, and Huimin Lu. Self-supervised diffusion-based scene flow estimation and motion segmentation with 4d radar. *IEEE Robotics and Automation Letters*, 2025.

[188] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[189] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020.

[190] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 107–122. Springer, 2020.

[191] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. MOTS: Multi-object tracking and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7942–7951, 2019.

[192] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixé, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8844–8854, 2022.

[193] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3D proposal generation and object detection from view aggregation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[194] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3D object detection. *arXiv preprint arXiv:1908.09492*, 2019.

[195] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019.

[196] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-time 3D object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[197] Weijing Shi and Raj Rajkumar. Point-GNN: Graph neural network for 3D object detection in a point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1711–1719, 2020.

[198] Erkan Baser, Venkateshwaran Balasubramanian, Prarthana Bhattacharyya, and Krzysztof Czarnecki. Fantrack: 3D multi-object tracking with feature association

network. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 1426–1433. IEEE, 2019.

[199] Johannes Pöschmann, Tim Pfeifer, and Peter Protzel. Factor graph based 3D multi-object tracking in point clouds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10343–10350. IEEE, 2020.

[200] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris M Kitani. GNN3DMOT: Graph neural network for 3D multi-object tracking with 2D-3D multi-feature learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6499–6508, 2020.

[201] Nuri Benbarka, Jona Schröder, and Andreas Zell. Score refinement for confidence-based 3D multi-object tracking. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8083–8090. IEEE, 2021.

[202] Jan-Nico Zaech, Alexander Liniger, Dengxin Dai, Martin Danelljan, and Luc Van Gool. Learnable online graph representations for 3D multi-object tracking. *IEEE Robotics and Automation Letters*, 7(2):5103–5110, 2022.

[203] Tao Wen, Yanyong Zhang, and Nikolaos M Freris. PF-MOT: Probability fusion based 3D multi-object tracking for autonomous vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 700–706. IEEE, 2022.

[204] Tara Sadjadpour, Jie Li, Rares Ambrus, and Jeannette Bohg. ShaSTA: Modeling shape and spatio-temporal affinities for 3D multi-object tracking. *arXiv preprint arXiv:2211.03919*, 2022.

[205] Mohamed Chaabane, Peter Zhang, J Ross Beveridge, and Stephen O'Hara. DEFT: Detection embeddings for tracking. *arXiv preprint arXiv:2102.02267*, 2021.

[206] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 474–490. Springer, 2020.

[207] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[208] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6247–6257, 2020.

[209] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[210] Jen-Hao Cheng, Sheng-Yao Kuan, Hou-I Liu, Hugo Latapie, Gaowen Liu, and Jenq-Neng Hwang. Centerradarnet: Joint 3d object detection and tracking framework using 4d fmcw radar. In *2024 IEEE International Conference on Image Processing (ICIP)*, pages 998–1004. IEEE, 2024.

[211] Matthias Zeller, Daniel Casado Herraez, Jens Behley, Michael Heidingsfeld, and Cyrill Stachniss. Radar tracker: Moving instance tracking in sparse and noisy radar point clouds. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16170–16177. IEEE, 2024.

[212] Jianan Liu, Guanhua Ding, Yuxuan Xia, Jinping Sun, Tao Huang, Lihua Xie, and Bing Zhu. Which framework is suitable for online 3d multi-object tracking for autonomous driving with automotive 4d imaging radar? In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 1258–1265. IEEE, 2024.

[213] Dong-In Kim, Dong-Hee Paek, Seung-Hyun Song, and Seung-Hyun Kong. Bayesian approximation-based trajectory prediction and tracking with 4d radar. *arXiv preprint arXiv:2502.01357*, 2025.

[214] Luis Roldao, Raoul De Charette, and Anne Verroust-Blondet. 3D semantic scene completion: A survey. *International Journal of Computer Vision*, 130(8):1978–2005, 2022.

[215] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1754, 2017.

[216] Shice Liu, Yu Hu, Yiming Zeng, Qiankun Tang, Beibei Jin, Yinhe Han, and Xiaowei Li. See and think: Disentangling semantic scene completion. *Advances in Neural Information Processing Systems*, 31, 2018.

[217] Jiahui Zhang, Hao Zhao, Anbang Yao, Yurong Chen, Li Zhang, and Hongen Liao. Efficient semantic scene completion network with spatial group convolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 733–749, 2018.

[218] Jie Li, Yu Liu, Xia Yuan, Chunxia Zhao, Roland Siegwart, Ian Reid, and Cesar Cadena. Depth based semantic scene completion with position importance aware loss. *IEEE Robotics and Automation Letters*, 5(1):219–226, 2019.

[219] Jie Li, Yu Liu, Dong Gong, Qinfeng Shi, Xia Yuan, Chunxia Zhao, and Ian Reid. RGBD-based dimensional decomposition residual network for 3D semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7693–7702, 2019.

[220] Pingping Zhang, Wei Liu, Yinjie Lei, Huchuan Lu, and Xiaoyun Yang. Cascaded context pyramid for full-resolution 3D semantic scene completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7801–7810, 2019.

[221] Jie Li, Kai Han, Peng Wang, Yu Liu, and Xia Yuan. Anisotropic convolutional networks for 3D semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3351–3359, 2020.

[222] Xiaokang Chen, Kwan-Yee Lin, Chen Qian, Gang Zeng, and Hongsheng Li. 3D sketch-aware semantic scene completion via semi-supervised structure prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4193–4202, 2020.

[223] Yingjie Cai, Xuesong Chen, Chao Zhang, Kwan-Yee Lin, Xiaogang Wang, and Hongsheng Li. Semantic scene completion via integrating instances and scene in-the-loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 324–333, 2021.

[224] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9297–9307, 2019.

[225] Julien Rebut, Arthur Ouaknine, Waqas Malik, and Patrick Pérez. Raw high-definition radar for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17021–17030, 2022.

[226] Guoqiang Zhang, Haopeng Li, and Fabian Wenger. Object detection and 3D estimation via an FMCW radar using a fully convolutional network. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4487–4491. IEEE, 2020.

[227] Yizhou Wang, Zhongyu Jiang, Yudong Li, Jenq-Neng Hwang, Guanbin Xing, and Hui Liu. RODNet: A real-time radar object detection network cross-supervised by camera-radar fused object 3D localization. *IEEE Journal of Selected Topics in Signal Processing*, 15(4):954–967, 2021.

[228] Xu Dong, Pengluo Wang, Pengyue Zhang, and Langechuan Liu. Probabilistic oriented object detection in automotive radar. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 102–103, 2020.

[229] Ao Zhang, Farzan Erlik Nowruzi, and Robert Laganiere. RadDet: Range-azimuth-doppler based radar object detection for dynamic road users. In *Proceedings of the 18th Conference on Robots and Vision (CRV)*, pages 95–102. IEEE, 2021.

[230] Andras Palffy, Jiaao Dong, Julian F. P. Kooij, and Dariu M. Gavrila. CNN-based road user detection using the 3D radar cube. *IEEE Robotics and Automation Letters*, 5(2):1263–1270, 2020.

[231] Bence Major, Daniel Fontijne, Amin Ansari, Ravi Teja Sukhavasi, Radhika Gowaikar, Michael Hamilton, Sean Lee, Slawomir Grzechnik, and Sundar Subramanian. Vehicle detection with automotive radar using deep learning on range-azimuth-doppler tensors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[232] Ruihan Liu, Xiaoyi Wu, Xijun Chen, Liang Hu, and Yunjiang Lou. 4d-rolls: 4d radar occupancy learning via lidar supervision. *arXiv preprint arXiv:2505.13905*, 2025.

[233] Long Yang, Lianqing Zheng, Wenjin Ai, Minghao Liu, Sen Li, Qunshu Lin, Shengyu Yan, Jie Bai, Zhixiong Ma, and Xichan Zhu. Metaocc: Surround-view 4d radar and camera fusion framework for 3d occupancy prediction with dual training strategies. *arXiv preprint arXiv:2501.15384*, 2025.

[234] Lianqing Zheng, Jianan Liu, Runwei Guan, Long Yang, Shouyi Lu, Yuanzhe Li, Xiaokai Bai, Jie Bai, Zhixiong Ma, Hui-Liang Shen, et al. Doracamom: Joint 3d detection and occupancy prediction with multi-view 4d radars and cameras for omnidirectional perception. *arXiv preprint arXiv:2501.15394*, 2025.

[235] P. J. Besl and Neil D. McKay. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[236] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[237] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST)*, pages 103–111, 2014.

[238] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 402–419. Springer, 2020.

[239] P. J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[240] Yuexin Ma, Tai Wang, Xuyang Bai, Huitong Yang, Yuenan Hou, Yaming Wang, Yu Qiao, Ruigang Yang, and Xinge Zhu. Vision-centric BEV perception: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[241] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7(3):652–674, 2022.

[242] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.

[243] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, volume 96, pages 226–231, 1996.

[244] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015.

[245] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[246] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12697–12705, 2019.

[247] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable scene flow from point clouds in the real world. *IEEE Robotics and Automation Letters*, 7(2):1589–1596, 2021.

[248] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[249] Felix Fent, Fabian Kuttenreich, Florian Ruch, Farija Rizwin, Stefan Juergens, Lorenz Lechermann, Christian Nissler, Andrea Perl, Ulrich Voll, Min Yan, and Markus Lienkamp. MAN TruckScenes: A multimodal dataset for autonomous trucking in diverse conditions. *arXiv preprint arXiv:2407.07462*, 2024.

[250] Stefan Andreas Baur, Frank Moosmann, and Andreas Geiger. LISO: LiDAR-only self-supervised 3D object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 253–270. Springer, 2024.

[251] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[252] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018.

[253] R Kwok and C Haas. Effects of radar side-lobes on snow depth retrievals from Operation IceBridge. *Journal of Glaciology*, 61(227):576–584, 2015.

[254] Peter Tait. *Introduction to radar target recognition*, volume 18. IET, 2005.

[255] Ming Nie, Yujing Xue, Chunwei Wang, Chaoqiang Ye, Hang Xu, Xinge Zhu, Qingqiu Huang, Michael Bi Mi, Xinchao Wang, and Li Zhang. PARTNER: Level up the polar representation for LiDAR 3D object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3801–3813, 2023.

[256] Qi Chen, Sourabh Vora, and Oscar Beijbom. Polarstream: Streaming object detection and segmentation with polar pillars. *Advances in Neural Information Processing Systems*, 34:26871–26883, 2021.

[257] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9939–9948, 2021.

[258] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[259] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[260] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[261] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[262] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4413–4421, 2018.

[263] Yiming Li, Sihang Li, Xinhao Liu, Moonjun Gong, Kenan Li, Nuo Chen, Zijun Wang, Zhiheng Li, Tao Jiang, Fisher Yu, et al. SSCBench: A large-scale 3D semantic scene completion benchmark for autonomous driving. *arXiv preprint arXiv:2306.09001*, 2023.

[264] Wenzhao Zheng, Weiliang Chen, Yuanhui Huang, Borui Zhang, Yueqi Duan, and Jiwen Lu. OccWorld: Learning a 3D occupancy world model for autonomous driving. *arXiv preprint arXiv:2311.16038*, 2023.

[265] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

[266] Zenseact. Zenseact Open Dataset (ZOD), 2024. Available at: `https://zod.zenseact.com/`.

[267] Yingjie Wang, Jiajun Deng, Yao Li, Jinshui Hu, Cong Liu, Yu Zhang, Jianmin Ji, Wanli Ouyang, and Yanyong Zhang. Bi-LRFusion: Bi-directional LiDAR-radar fusion for 3D dynamic object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13394–13403, 2023.

[268] Jingzhong Li, Yuyi Wang, Lin Yang, Jun Lin, Gaoqiang Kang, Zhen Shi, Yuxuan Chen, Yue Jin, and Kanta Akiyama. L-RadSet: A long-range multimodal dataset with 4D radar for autonomous driving and its application. *IEEE Transactions on Intelligent Vehicles*, 2024.

[269] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 18–35. Springer, 2024.

[270] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

[271] Wikipedia Contributors. Tesla autopilot. `https://en.wikipedia.org/wiki/Tesla_Autopilot`, 2025. Accessed: 2025-02-09.

[272] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7467–7477, 2020.

[273] Tang Tao, Longfei Gao, Guangrun Wang, Yixing Lao, Peng Chen, Hengshuang Zhao, Dayang Hao, Xiaodan Liang, Mathieu Salzmann, and Kaicheng Yu. LiDAR-NeRF: Novel lidar view synthesis via neural radiance fields. In *Proceedings of the 32nd ACM International Conference on Multimedia (ACM MM)*, pages 390–398, 2024.

[274] You Li and Javier Ibanez-Guzman. LiDAR for autonomous driving: The principles, challenges, and trends for automotive LiDAR and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61, 2020.

[275] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018.

[276] Guangming Wang, Yunzhe Hu, Zhe Liu, Yiyang Zhou, Masayoshi Tomizuka, Wei Zhan, and Hesheng Wang. What matters for 3D scene flow network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 38–55, 2022.

[277] Wencan Cheng and Jong Hwan Ko. Bi-PointFlowNet: Bidirectional learning for point cloud based scene flow estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 108–124, 2022.

[278] SpConv Contributors. SpConv: Spatially sparse convolution library. `https://github.com/traveller59/spconv`, 2022.

[279] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.