DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 957

## Yunshuang Yuan

# Collective Perception: A Fully-Sparse Deep Learning Framework for Multi-Agent Data Fusion

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN-978-3-7696-5369-4

DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 957

# Collective Perception:
# A Fully-Sparse Deep Learning Framework
# for Multi-Agent Data Fusion

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte
Dissertation

von

M. Sc. Yunshuang Yuan

geboren am 25.09.1992 in Hubei, China

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN-978-3-7696-5369-4

Adresse des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften:

Ausschuss Geodäsie (DGK) der Bayerischen Akademie der Wissenschaften
Alfons-Goppel-Straße 11 ● D – 80 539 München
Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100
e-mail post@dgk.badw.de ● http://www.dgk.badw.de

Prüfungskommission:
Vorsitzender: Prof. Dr.-Ing. habil. Jürgen Müller
Referentin: Prof. Dr.-Ing. habil. Monika Sester
Korreferenten: Prof. Dr.-Ing. habil. Christian Heipke
Prof. Dr.-Ing. Markus Fidler

Tag der mündlichen Prüfung: 31.03.2025

Diese Dissertation ist auf dem Server des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften, München unter <http://dgk.badw.de/>
sowie unter Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik
der Leibniz Universität Hannover, ISSN 0174-1454, Nr. 409
unter <https://repo.uni-hannover.de/handle/123456789/19181>, Hannover 2025 elektronisch publiziert

---

© 2025 Ausschuss Geodäsie (DGK) der Bayerischen Akademie der Wissenschaften, München

Alle Rechte vorbehalten. Ohne Genehmigung der Herausgeber ist es auch nicht gestattet,
die Veröffentlichung oder Teile daraus zu vervielfältigen.

ISSN 0065-5325 ISBN-978-3-7696-5369-4

# Abstract

Autonomous driving has garnered significant attention in recent years due to its transformative potential in mobility, including enhanced road safety, reduced traffic congestion, and improved economic efficiency. Equipped with advanced sensors, AI-driven perception, and decision-making capabilities, autonomous driving systems demonstrate the potential to surpass human drivers in efficiency and reliability. However, ego-centric autonomous systems are constrained by limited sensor ranges and occlusions, which hinder their ability to perceive complex and dynamic environments effectively. To address these challenges, collective perception has emerged as a promising solution. By leveraging communication technologies, collective perception enables data sharing among intelligent agents (*e.g.,* autonomous vehicles and smart infrastructure) within multi-agent systems. This data sharing significantly improves situational awareness while reducing uncertainties.

This thesis investigates multi-agent data fusion based on deep-learning algorithms and point cloud data. While recent advancements have tackled problems with occlusions and extended perception ranges, existing collective perception approaches often overlook critical issues, including inefficiencies in data processing and sharing, communication latency, unsynchronized sensor data, and spatial misalignment caused by localization errors. Current methods predominantly rely on dense bird's-eye-view (BEV) maps derived from sparse point cloud data, leading to computational inefficiencies and excessive bandwidth consumption due to the transmission of dense feature maps generated by neural networks. To overcome these limitations, this thesis proposes fully sparse neural networks that leverage the intrinsic sparsity of point clouds, minimizing computational overhead and reducing communication bandwidth requirements. Additionally, the thesis addresses overlooked challenges, such as unsynchronized sensor data, ensuring a more robust approach to collective perception. Furthermore, a novel uncertainty estimation approach is also introduced, enhancing the reliability of perception systems.

Specifically, this dissertation presents two innovative fully-sparse neural networks: *GevBEV* for BEV semantic segmentation and *SparseAlign* for object detection. Together with the data preprocessing and training techniques used for these networks, they are referred to as *framework* in this thesis. The BEV semantic segmentation framework generates BEV maps to guide route planning, while the object detection framework identifies traffic participants to aid collision avoidance.

Unlike previous traditional BEV semantic segmentation frameworks that produce discrete BEV maps, the proposed framework, *GevBEV*, employs spatial Gaussian distributions on the sparse and discrete observation points to interpret the driving space in a continuous manner. This approach allows for precise segmentation details while avoiding unreliable extrapolation in unobserved areas. Any point within the continuous space is supported by evidence from the Gaussian distributions centered on the original observation points. Moreover, *GevBEV* utilizes the Dirichlet distribution and evidential learning to estimate uncertainty, outperforming state-of-the-art frameworks in BEV semantic segmentation while providing enhanced reliability for decision-making and shared-data selection in cooperative autonomous driving.

This thesis also introduces a novel object detection task, *Time-Aligned Cooperative Object Detection* (TA-COOD), which addresses sensor asynchrony. The task involves processing asynchronous sensor data to predict objects at globally aligned timestamps. To tackle this challenge, a novel fully sparse framework, *SparseAlign*, is proposed to efficiently process sequential point cloud data and capture accurate temporal context from the point-wise timestamps of the point clouds. *SparseAlign* has been experimentally demonstrated to significantly outperform state-of-the-art frameworks while requiring fewer computational resources. Ablation studies further demonstrate that the temporal modeling based on point-wise timestamps is crucial for capturing accurate temporal context of objects, leading to more precise predictions for TA-COOD.

**Keywords:** Autonomous driving, collective perception, semantic segmentation, object detection, data fusion, point clouds

## Kurzfassung

Autonomes Fahren hat in den letzten Jahren aufgrund seines transformativen Potenzials in der Mobilität erhebliche Aufmerksamkeit erlangt: dazu zählen verbesserte Verkehrssicherheit, reduzierte Verkehrsstaus und gesteigerte wirtschaftliche Effizienz. Ausgestattet mit fortschrittlichen Sensoren, KI-gesteuerter Wahrnehmung und Entscheidungsfähigkeiten, zeigen autonome Fahrsysteme das Potenzial, menschliche Fahrer in Effizienz und Zuverlässigkeit zu übertreffen. Ego-zentrierte autonome Systeme sind jedoch durch begrenzte Sensorreichweiten und Sichtbehinderungen eingeschränkt, die ihre Fähigkeit, komplexe und dynamische Umgebungen effektiv wahrzunehmen, behindern. Um diese Herausforderungen zu bewältigen, hat sich das Konzept der kollektiven Wahrnehmung als vielversprechende Lösung herausgebildet. Durch den Einsatz von Kommunikationstechnologien ermöglicht die kollektive Wahrnehmung den Datenaustausch und die Datenfusion zwischen intelligenten Agenten (z. B. autonomen Fahrzeugen und intelligenten Infrastrukturen) in Multi-Agenten-Systemen. Dieser Datenaustausch verbessert die Situationswahrnehmung erheblich und reduziert Unsicherheiten.

Diese Dissertation untersucht die Datenfusion in Multi-Agenten-Systemen basierend auf Deep-Learning-Algorithmen und Punktwolkendaten. Trotz bedeutender Fortschritte, beispielsweise in der Behandlung von Sichteinschränkungen und Wahrnehmungsreichweiten, werden in bestehenden kollektiven Wahrnehmungsansätzen oft kritische Probleme übersehen, einschließlich Ineffizienz in der Datenverarbeitung und -weitergabe, Kommunikationslatenz, nicht-synchronisierte Sensordaten und räumliche Fehlanpassungen aufgrund der Lokalisierungsfehler. Aktuelle Methoden stützen sich überwiegend auf dichte Vogelperspektivenkarten (BEV), die aus dünn besetzten Punktwolkendaten abgeleitet werden, was zu Berechnungsineffizienz und einem übermäßigen Bandbreitenverbrauch führt, da dichte Merkmalskarten, die von neuronalen Netzwerken erzeugt werden, übertragen werden. Um diese Einschränkungen zu überwinden, schlägt diese Dissertation Methoden vor, die dünn besetzte Strukturen von Punktwolken ausnutzen, den Rechenaufwand minimieren und die Anforderungen an die Kommunikationsbandbreite verringern. Darüber hinaus werden bislang nicht betrachtete Herausforderungen, wie nicht-synchronisierte Sensordaten, adressiert, um eine robustere Herangehensweise an die kollektive Wahrnehmung zu gewährleisten. Des Weiteren wird ein neuartiger Ansatz zur Unsicherheitsabschätzung eingeführt, der die Zuverlässigkeit der Wahrnehmungssysteme verbessert.

Insbesondere stellt diese Dissertation zwei innovative, vollständig dünn besetzte neuronale Netzwerke vor, *GevBEV* für BEV-semantische Segmentierung und *SparseAlign* für Objekterkennung. Zusammen mit den für diese Netzwerke verwendeten Datenvorverarbeitungs- und Trainingstechniken werden sie in dieser Arbeit als *Framework* bezeichnet. Das BEV-semantische Segmentierungsframework erzeugt BEV-Karten zur Routenplanung, während das Objekterkennungsframework Verkehrsteilnehmer identifiziert, um Kollisionen zu vermeiden.

Im Gegensatz zu traditionellen BEV-semantischen Segmentierungsframeworks, die diskrete BEV-Karten erzeugen, verwendet das vorgeschlagene Framework *GevBEV* räumliche Gaußsche Verteilungen auf den dünn besetzten und diskreten Beobachtungspunkten, um den Fahrraum kontinuierlich zu interpretieren. Dieser Ansatz ermöglicht eine präzise Segmentierung, während unzuverlässige Extrapolationen in unobservierte Bereiche vermieden werden. Jeder Punkt im kontinuierlichen Raum wird aus den auf den ursprünglichen Beobachtungspunkten zentrierten Gaußverteilungen bestimmt. Darüber hinaus nutzt *GevBEV* die Dirichlet-Verteilung und evidenzbasiertes Lernen zur Unsicherheitsabschätzung; dies führt dazu, dass es die aktuellen State-of-the-Art-Modelle in der BEV-semantischen Segmentierung übertrifft und gleichzeitig eine erhöhte Zuverlässigkeit für Entscheidungsfindung und Datenauswahl im kooperativen autonomen Fahren bietet.

Diese Dissertation stellt eine neue Objekterkennungsaufgabe vor, *Time-Aligned Cooperative Object Detection* (TA-COOD), die mit Sensor-Asynchronität umgeht. Die Aufgabe umfasst die Verarbeitung asynchroner Sensordaten, um Objekte zu global ausgerichteten Zeitstempeln vorherzusagen. Zur Bewältigung dieser Herausforderung wird ein neuartiges, vollständig dünn besetztes Framework, *SparseAlign*, vorgeschlagen, das sequentielle Punktwolkendaten effizient verarbeitet und präzisen zeitlichen Kontext aus den punktweisen Zeitstempeln der Punktwolken extrahiert. Es konnte experimentell gezeigt werden, daß *SparseAlign* Framework geringere rechnerische Anforderungen hat und dennoch eine deutlich bessere Leistung als die aktuellen State-of-the-Art-Frameworks bietet. Ablationsstudien zeigen außerdem, dass das zeitliche Modellieren auf Basis der punktweisen Zeitstempel eine wichtige Rolle dabei spielt, den genauen zeitlichen Kontext von Objekten zu erfassen, was zu präziseren Detektionen für TA-COOD führt.

**Schlüsselwörter:** Autonomes Fahren, kollektive Wahrnehmung, semantische Segmentierung, Objekterkennung, Datenfusion, Punktwolken

## List of Acronyms

ADAS  Advance Driver Assistance Systems

AP  Average Precision

BEV  Bird's Eye View

CAV  Connected Autonomous Vehicle

CDV  Cooperative Distributed Vision

CFM  Center Feature Missing

COOD  COoperative Object Detection

CPM  Collective Perception Message

DNN  Deep Neural Network

GT  Ground-Truth

GUI  Graphical User Interface

ICF  Isolated Convolution Field

II  Intelligent Infrastructure

IoU  Intersection over Union

ITS  Intelligent Transportation System

IV  Intelligent Vehicle

SOTA  state-of-the-art

TA-COOD  Time-Aligned COoperative Object Detection

# Contents

**1 Introduction** **15**
- 1.1 The Concept of Collective Perception 15
- 1.2 Intelligent Vehicles (IVs) 17
- 1.3 Safe and Efficient Mobility with IVs and Collective Perception 18
- 1.4 Motivation and Objectives 19
- 1.5 Contributions 20
- 1.6 Structure of the Thesis 20

**2 Background** **23**
- 2.1 A Brief History of Communication for Autonomous Driving 23
- 2.2 Autonomous Driving Systems (ADSs) 24
  - 2.2.1 Sensors and perception 24
  - 2.2.2 Localization 25
  - 2.2.3 Communication 26
- 2.3 Implementation of Collective Perception 28
  - 2.3.1 CPM preparation 28
  - 2.3.2 CPM sharing 29
  - 2.3.3 CPM fusion 29

**3 Theoretical Fundamentals** **31**
- 3.1 Deep Learning Modules 31
  - 3.1.1 Multi-layer perceptrons 31
  - 3.1.2 Convolutional neural network 31
  - 3.1.3 Sparse convolution 33
  - 3.1.4 Attention mechanism and transformers 35
- 3.2 Components of Collective Perception 38
  - 3.2.1 Backbone neural networks 38
  - 3.2.2 Feature fusion 41
  - 3.2.3 Task heads 43
  - 3.2.4 Loss functions 45
- 3.3 Pose Graph Optimization for Pose Alignment 50
- 3.4 Evaluation Metrics 51
  - 3.4.1 Average Precision (AP) 51
  - 3.4.2 Intersection over Union (IoU) 52
  - 3.4.3 Calibration plot 54

**4 Related Work 57**
4.1 Representation of Driving Spaces . . . . . . . . . . 57
4.2 Ego-based Perception Systems . . . . . . . . . . 58
  4.2.1 Object Detection . . . . . . . . . . 58
  4.2.2 BEV Semantic Segmentation . . . . . . . . . . 59
  4.2.3 Temporal modeling of sequential data . . . . . . . . . . 59
  4.2.4 Uncertainty estimation . . . . . . . . . . 60
4.3 Collective Perception . . . . . . . . . . 61
  4.3.1 Model efficiency . . . . . . . . . . 62
  4.3.2 Sensor asynchrony . . . . . . . . . . 62
  4.3.3 Datasets for collective perception . . . . . . . . . . 62

**5 Datasets 65**
5.1 Simulation Datasets . . . . . . . . . . 65
  5.1.1 OPV2V . . . . . . . . . . 65
  5.1.2 OPV2Vt . . . . . . . . . . 65
5.2 Real Datasets . . . . . . . . . . 67
  5.2.1 V2Vreal . . . . . . . . . . 67
  5.2.2 DairV2X . . . . . . . . . . 68
  5.2.3 DairV2Xt . . . . . . . . . . 68

**6 Framework Design 73**
6.1 Formalization of Collective Perception . . . . . . . . . . 73
6.2 CoSense3D: a Efficient Framework for Collective Perception . . . . . . . . . . 74
6.3 Standardized Modules of CoSense3D . . . . . . . . . . 78
  6.3.1 Overview . . . . . . . . . . 78
  6.3.2 Data Augmentation . . . . . . . . . . 79
  6.3.3 Optional modules . . . . . . . . . . 80
  6.3.4 Spatial Fusion . . . . . . . . . . 81
  6.3.5 Object detection head . . . . . . . . . . 81
  6.3.6 BEV Semseg head . . . . . . . . . . 82
  6.3.7 Target assigner . . . . . . . . . . 83
6.4 Efficiency of CoSense3D . . . . . . . . . . 85
  6.4.1 Efficient data processing and training . . . . . . . . . . 85
  6.4.2 Experimental analysis of training efficiency . . . . . . . . . . 86

**7 Gaussian-based Evidential BEV Semantic Segmentation (GevBEV) 91**
7.1 Definition of GevBEV . . . . . . . . . . 91
7.2 Architecture of the Framework GevBEV . . . . . . . . . . 93
  7.2.1 Overview . . . . . . . . . . 93
  7.2.2 Spatial fusion . . . . . . . . . . 94
  7.2.3 Heads . . . . . . . . . . 94

7.2.4 CPM selection . . . 96
7.3 Experiments . . . 97
7.3.1 State-of-the-art models for comparison . . . 97
7.3.2 Ablation studies . . . 97
7.3.3 Implementation details . . . 98
7.4 Evaluation Metrics . . . 98
7.4.1 IoU . . . 98
7.4.2 Calibration plot . . . 99
7.4.3 Notations . . . 100
7.5 Results . . . 100
7.5.1 Qualitative analysis . . . 100
7.5.2 Quantitative Analysis . . . 104
7.5.3 Uncertainty-based information selection for collective perception . . . 109
7.5.4 Summary . . . 110

**8 Time-Aligned Cooperative Object Detection Using Fully Sparse Neural Network 111**
8.1 Definition of TA-COOD . . . 111
8.2 Architecture of the Framework SparseAlign . . . 112
8.2.1 Overview . . . 112
8.2.2 Sparse UNet (SUNet) . . . 112
8.2.3 Temporal alignment (TAM) . . . 114
8.2.4 Pose alignment (PAM) . . . 115
8.2.5 Spatial alignment (SAM) . . . 116
8.2.6 Detection heads . . . 117
8.3 Experiments . . . 117
8.3.1 Comparative models . . . 117
8.3.2 Training . . . 117
8.4 Results . . . 118
8.4.1 Qualitative results . . . 118
8.4.2 Comparison to SOTA models . . . 119
8.4.3 Training efficiency . . . 120
8.4.4 Ablation study . . . 121
8.4.5 Communication latency and sensor asynchrony . . . 121
8.4.6 Localization errors . . . 123
8.4.7 CPM sizes . . . 124
8.5 Summary . . . 125

**9 Summary and Conclusion 127**

**10 Outlook 129**

**List of Figures 131**

**List of Tables** 135

**References** 137

# 1 Introduction

## 1.1 The Concept of Collective Perception

### Benefits of cooperative work

In the context of biological evolution, cooperation proves advantageous within the harsh framework of competitive selection. Cooperation refers to the process by which a group of organisms collaborates and acts in pursuit of a common or mutually beneficial goal. For instance, certain ant species engage in collective decision-making when identifying new nesting sites. Similarly, scout bees communicate information regarding potential sites through pheromone signaling, enabling the entire colony to collectively evaluate and choose the most suitable location. In another example, orcas exhibit cooperative behavior during hunting, utilizing intricate vocalizations and coordinated strategies to efficiently capture prey.

Mimicking animal behaviors, swarm robots initially embraced the concept of cooperation in numerous research endeavors by the end of the last century (Dudek et al., 1993). This collaborative framework, wherein multiple robots work in unison, facilitates the execution of complex tasks, reducing the necessity for sophisticated individual robotic design. Such a collective design paradigm enhances the economic efficiency, scalability, and resilience of the overall robotic system, making it considerably more robust against potential system-wide failures.

However, as highly intelligent organisms, animals often build more intricate systems of cooperation. As per Poole et al. (1998), an Intelligent Agent (IA) is a system that operates intelligently within an environment by reasoning based on its knowledge, analyzing and interacting with the environment accordingly. In the case of a robotic artificial agent, knowledge is acquired through sensors that capture environmental data, which is subsequently processed into a format suitable for guiding appropriate actions.

A seminal contribution to the field of cooperative intelligent robotics is the Cooperative Distributed Vision (CDV) system introduced by Matsuyama (1999). Matsuyama posits that intelligence is not confined to a central processing unit, such as the brain, but rather emerges from dynamic interactions with the surrounding environment. Accordingly, the CDV system presents an intelligent framework grounded in these dynamic interactions among distributed vision-enabled agents. This framework consists of three key modules: perception, action, and communication. Utilizing the images captured by each agent's camera, the perception module analyzes the environment and identifies moving objects within the images. Through communication and sharing the detected objects, the agents collaboratively adjust their orientations towards target objects, thereby improving their observational capabilities and understanding of the environment.

### Cooperation in Intelligent Transportation Systems (ITSs)

The idea of cooperation extends to transportation systems, where every participant within traffic is considered an individual agent. These agents may have different departure points and destinations, yet they frequently traverse the same road segments. By collaboratively sharing real-time traffic conditions, overall traffic safety and efficiency can be significantly enhanced, benefiting all traffic participants involved. As intelligence is integrated into these agents, this collaborative system evolves into an ITS, a comprehensive transportation and management system that efficiently integrates advanced technologies including information, communication, sensor, control and computer technology into the entire transportation management system. It creates a comprehensive,

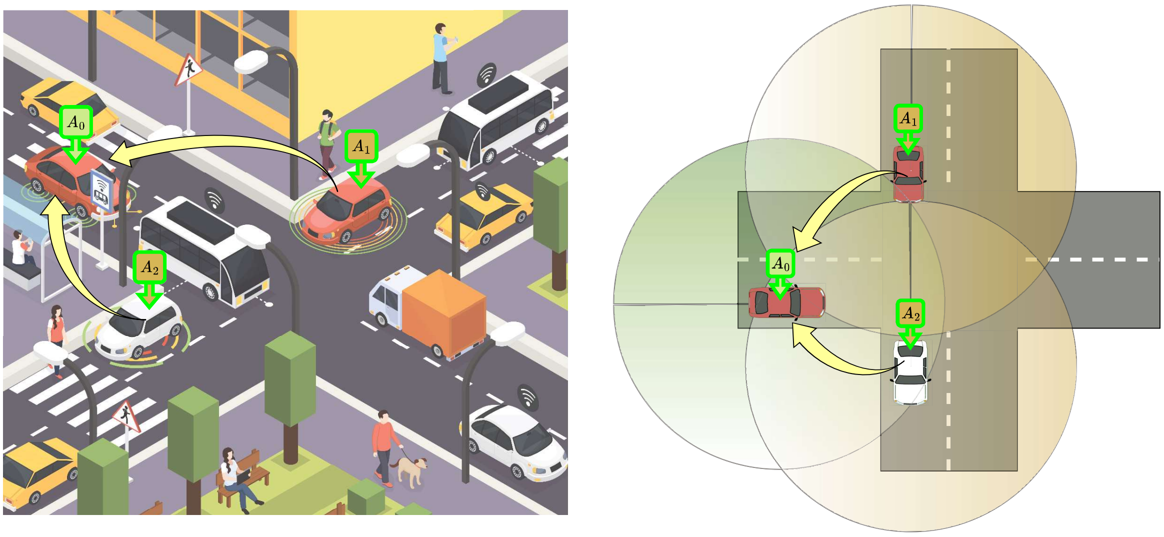


*Figure 1.1: Collective perception scenario*[1]

real-time, accurate, and highly efficient system that plays a role in a wide range and in all aspects of transportation. Within this system, agents collaboratively enhance each other's capabilities, particularly in terms of sensory functions and field-of-view (FoV). This cooperative interaction enables more effective environmental perception, optimizing overall system performance. As per European Telecommunications Standards Institute (ETSI, 2023), such cooperative components, including IA, Intelligent Infrastructures (II) that contribute to the operation and functionality of the ITS, are called ITS-stations. These stations are interconnected and facilitate information sharing.

### Collective perception: understanding the environment with cooperation

Perception is the process of analyzing, identifying and interpreting the sensory information to understand the information and the environment. It is the cornerstone for comprehending the behavior of traffic participants and anticipating their future actions, a crucial step towards realizing an effective ITS. Through cooperative work, the intelligent traffic agents interact with each other and exchange sensory information. By analyzing the information collected from different agents, the perception system achieves a more comprehensive understanding of the environment. Therefore, collective perception is defined as the process of perception based on the collective sensory information. As the example shows in Figure 1.1, the view of vehicle V0 is occluded by the bus and this vehicle is not able to observe the traffic situations at the intersection. However, by utilizing the information shared by vehicles V1 and V2, vehicle V0 is able to expand its perceptual range. This extended perception enables vehicle V0 to more effectively capture and interpret the ongoing situations at the intersection, thereby facilitating the formulation of safer and more efficient driving decisions.

Officially, the technical protocol outlined by ETSI (2023) defines Collective Perception as an integral component of ITS that involves sharing safety-relevant information about the current context of the ITS-station's environment. The data shared among ITS-stations, encapsulated within a Collective Perception Message (CPM), serves the purpose of enhancing environmental awareness through cooperation. This thesis adopts this definition and concentrates on ITS-stations, specifically intelligent vehicles and infrastructures.

---

[1] Left image in the figure is a modified version of the image designed by macrovector/Freepik, `https://www.freepik.com/free-vector/autonomous-vehicle-isometric_6375496.htm`

## 1.2 Intelligent Vehicles (IVs)

Intelligent Vehicles (IVs) are vehicles equipped with intelligent driving systems that enable them to operate in either partial or fully autonomous modes. With advanced modern sensors and information and computation technologies, IVs are becoming increasingly intelligent. In transportation systems, the cooperation of IVs with the IIs has offered significant potential for improving road safety, enhancing efficiency by minimizing travel time, promoting fuel economy, reducing greenhouse gas emissions, and ensuring environmental sustainability, all while maintaining passenger comfort. Due to these advantages, Original Device Manufacturers (ODMs) globally have invested substantial resources in the development of IVs, aiming to alleviate human drivers from some or all driving responsibilities. The Society of Automotive Engineers (SAE, Table 1.1) classifies IV automation into six levels, based on the degree of human intervention required in driving tasks. The first three levels necessitate constant human involvement, whereas the final three levels allow the vehicle to assume full control under specific or all conditions.

| Level | Description | Examples |
|---|---|---|
| 0 | No Automation | |
| 1 | Driver Assistance | adaptive cruise control |
| 2 | Partial Automation | vehicle can control both steering and acceleration/deceleration simultaneously |
| 3 | Conditional Automation | vehicle can manage most aspects of driving, but a human driver might need to intervene if prompted |
| 4 | High Automation | vehicle can perform all driving functions under certain conditions or environments |
| 5 | Full Automation | vehicle can perform all driving functions in all conditions without human intervention |

*Table 1.1: Levels of Automation in Autonomous Driving (SAE J3016 SAE International (2021))*

By the end of 2023, the automation level (ref. Table 1.1) of the mainstream IVs equipped with Advanced Driver Assistance Systems (ADASs) has remained stagnant, predominantly hovering at levels 2 and 3 for an extended period. Notably, only a few companies had ventured into commercializing higher-level autonomous vehicles allowing for limited driverless operation in specific conditions. For instance, Waymo and Baidu Apollo successfully launched their level-4 Robotaxi services to be operated in San Francisco and Shanghai, respectively. In the long run, the automotive industry aims to develop vehicles being entirely autonomous (SAE, level 5), freeing humans from the act of driving. These vehicles, known as Self-Driving Vehicles (SDVs) or Autonomous Vehicles (AVs), possess the capability to drive independently and optimize decision-making. This transition holds promise for significantly safer driving practices and a reduction in traffic accidents.

However, the intricate real-world driving environment poses technological, regulatory, and safety challenges for achieving full automation. For instance, low-light conditions during nighttime driving may impair the visibility provided by camera-based systems. This limitation can be addressed through the use of LiDAR, a sensor that determines object distances by emitting and receiving laser signals. Nevertheless, both LiDAR and camera systems may encounter difficulties under adverse weather conditions, such as snow, rain, and fog. Moreover, all sensors are constrained by limitations in sensing range and resolution. Addressing these challenges is a critical prerequisite for the realization of fully autonomous driving systems.

From the perspective of perception, the challenges in achieving full automation stem from insufficient sensory data and limitations in knowledge acquisition within the driving environment. While SAE Level 4 IVs are capable of managing most driving scenarios, they struggle with "corner cases"—rare but recurrent situations encountered on many driving routes that constitute a small portion of overall driving mileage. Some of these corner cases are caused by insufficient observations. For instance, a pedestrian crossing the road might be occluded by the building or infrastructures. Equipped with 36 sensors to ensure redundancy in sensory information, Baidu's Apollo RT6 might still face challenges in overcoming occlusion. As all sensors are mounted on the same vehicle, they are subject to limitations such as occlusion and restricted sensing range, underscoring the critical importance of better sensor placement. Instead, collecting the perceptual information from neighboring AIs and IIs that are placed at vantage positions could reduce blind spots, extend the FoV and enhance perception capabilities of the ego vehicle, achieving a more comprehensive understanding of its surroundings.

## 1.3 Safe and Efficient Mobility with IVs and Collective Perception

As the population and economy expand, the demand for mobility grows substantially, amplifying the complexity of transportation needs within constrained time and space. This surge poses significant challenges to developing effective traffic solutions without compromising safety and comfort. Instead of building new traffic infrastructures such as broader and better road networks, optimized travel and driving decisions on the existing roads are more economical and sustainable to meet the growing mobility demands.

Safer driving operations contribute to a decreasing number of accidents, subsequently mitigating congestion. It also reduces traveling time by optimized route selection based on real-time traffic information. Additionally, efficient negotiation among traffic participants leads to an overall enhanced traffic flow, offering benefits to all involved parties.

However, assuring driving operations necessitates continuous driver focus throughout the trip. This attentiveness enables drivers to consistently observe and comprehend their surroundings, enabling sound and timely driving decisions. However, human drivers might unintentionally disregard traffic rules due to a lack of awareness or while operating a vehicle under subconscious or even unconscious conditions, such as intoxicated conditions under the influence of alcohol, or fatigued after extended travel periods. These misconducts of drivers contribute to 88% of traffic accidents in Germany (Statistisches Bundesamt, 2017). As a compensation for human drivers against their imperfectness in driving, modern IVs are equipped with ADASs to simplify their driving tasks and assist drivers in responding correctly to emergent situations. For example, Adaptive Cruise Control (ACC) adjusts the vehicle's speed—accelerating, decelerating, and occasionally stopping—in response to the movements of nearby objects. This technology relieves drivers from the constant need to monitor their vehicle's speed and the actions of surrounding vehicles, consequently lowering the risk of fatigue-related driving incidents. Automatic Emergency Braking (AEB) aids drivers in imminent collision risks and promptly responding to such situations. Benefiting from these assistance functionalities, fatal accidents have been decreasing in recent years, despite an overall increase in the total number of accidents. (Statistisches Bundesamt, 2023). With higher levels of vehicle automation enabled by collective perception, observations of the driving environment have the potential to become more comprehensive and precise, thereby enhancing overall safety.

A higher automation level requires sufficient knowledge for a comprehensive understanding of the driving environment. Achieving this depth of understanding necessitates communication to expand the knowledge base of IVs in both macro and micro perspectives. This ensures not only safer operations but also optimized routing and enhanced traffic efficiency through negotiations. In instances where instantaneous decisions for subsequent driving operations are necessary, a comprehensive understanding of nearby object behaviors on a micro scale is vital. Through communication with nearby objects and adjacent infrastructures, the ego-vehicle can expand its observation range, minimizing occlusions, and enhancing environmental comprehension. Real-time traffic information reception and negotiations with other networked vehicles in macro scale allow for route optimization among these involved vehicles.

## 1.4 Motivation and Objectives

Collective perception mainly contains three stages: data preparation, sharing and fusion. Previous works (Chen et al., 2019a; Marvasti et al., 2020b; Xu et al., 2022c; Wang et al., 2020a; Xu et al., 2022b,a) have made significant progress in collective perception by sharing dense Bird's Eye View (BEV) feature maps which are extracted by Deep Neural Networks (DNN). They have proven that preparing and sharing the intermediate-processed feature maps outperforms the frameworks that share the fully processed information, such as the detected bounding boxes of objects. These detected results contain errors which directly influence the accuracy of the fusion results. In addition, the size of intermediate-processed data is adjustable. With a good data selection strategy, sharing the intermediate-processed data is more communication-efficient than the bulky raw data (*e.g.,* point clouds) sharing. Therefore, this thesis focuses on preparing, sharing and fusing the intermediate-processed data.

However, different from sharing the BEV feature maps which demands substantial communication bandwidth and high computational resources for processing, this thesis concentrates on more efficient sparse frameworks that utilize the sparsity of point-cloud data. Typically, previous works employ 2D or 3D dense convolutions to extract features from point clouds. As the detection range of point clouds expands, the computational demands of these frameworks increase quadratically or cubically. Exploring the sparsity feature of point clouds, this thesis attempts to explore fully-sparse frameworks that are more computational- and communication-efficient to solve the object detection and BEV semantic segmentation problem. Beyond the framework design, training these models is also computationally demanding because the data from multiple agents should be processed at the same time. Thus, this thesis also explores efficient training methods.

After the intermediate-processed data are shared and fused, they can be used for different perception tasks. Object detection and semantic segmentation are two primary perception tasks. Object detection focuses on identifying and locating other traffic participants within the driving scene, providing data that can be used to predict their behavior and avoid potential collisions. In contrast, semantic segmentation assigns each data point—such as pixels in camera images or points in LiDAR-generated point clouds—a semantic label to distinguish different types of surfaces and objects. Since IVs typically operate on approximately 2D road surfaces, sensor data can be projected into BEV space to produce BEV semantic segmentation results. This projection aids route planning by offering a detailed overview of the static environment, particularly the occupancy status of drivable surfaces, which is crucial for safe path planning. Together, object detection and BEV semantic segmentation provide complementary information about the dynamic and static elements of the driving environment, making them the primary perception tasks in this work.

In BEV semantic segmentation tasks, estimating the uncertainty of the predictive results is essential for enabling the model to recognize its own limitations. However, prior research has not focused on uncertainty estimation. This thesis aims to address this gap by modeling uncertainty and evaluating the reliability of the generated predictive semantic segmentation results. Also different from previous works that make predictions over the unobserved areas, this thesis takes advantage of sparse operations and only learns and predicts based on the observed areas to prevent invalid predictions which may lead to safety problems. For example, an unobserved road area which is occupied by other vehicles might also be classified as an empty drivable road surface because it is close to an observed road surface. This might lead to traffic accidents.

For the object detection task, the detection accuracy is very sensitive to the localization errors and time asynchrony. Because the objects (vehicles) are highly dynamic. In addition to framework design for vehicle detection, this thesis addresses the fusion challenges posed by localization errors and time asynchrony. While previous studies have attempted to mitigate the influence of localization errors (Wang et al., 2020a; Xu et al., 2022a; Yuan et al., 2022; Yuan and Sester, 2022) and communication delays (Xu et al., 2023; Yu et al., 2023), none have addressed the issue of asynchronous sensor timestamps. To bridge this gap, this work introduces a novel benchmark, time-aligned cooperative object detection (TA-COOD), which aims to align objects observed by asynchronous sensors and predict the final object position at a globally synchronized timestamp.

## 1.5 Contributions

The contributions of this thesis are summarized as follows:

1. This work proposes a novel probabilistic model for BEV semantic segmentation, named *GevBEV*. It interprets the 2D driving space as a probabilistic BEV map with point-based spatial Gaussian distributions, from which one can draw density values as the evidence parameters for the categorical Dirichlet distribution at any new sample point in the continuous driving space. The experimental results show that GevBEV not only provides more reliable uncertainty quantification but also outperforms the previous works on the benchmarks OPV2V and V2V4Real of BEV map interpretation for cooperative perception in simulated and real-world driving scenarios, respectively. A critical factor in cooperative perception is the data transmission size through the communication channels. GevBEV helps reduce communication overhead by selecting only the most important information to share from the learned uncertainty, reducing the average information communicated by 87% with only a slight performance drop.

2. To explore the influence of the asynchronous sensor ticking time on the collective perception, this work propose to predict a bounding box for each object at the globally aligned timestamp and propose the benchmark TA-COOD. For this purpose, two new datasets *OPV2Vt* and *DairV2Xt* are generated by interpolating the data frames of the OPV2V (Xu et al., 2022c) and DairV2X (Yu et al., 2022). Based on these two datasets, an efficient spatial-temporal fusion model *SparseAlign* is designed to process the sequential point cloud data to generate the TA-COOD result. The experimental results confirmed the superior efficiency and performance of the proposed fully sparse framework compared to the state-of-the-art dense models. More importantly, they show that the point-wise observation timestamps of the dynamic objects are crucial for modeling the object temporal context and recovering their time-related locations.

3. Fully sparse frameworks are explored for both BEV semantic segmentation and object detection. To this purpose, efficient 3D backbone networks and fusion methods are designed. The experimental results on the synthetic dataset OPV2V (Xu et al., 2022c) and the real datasets V2Vreal (Xu et al., 2023) and DairV2X (Yu et al., 2022) all show that the proposed frameworks outperform the state-of-the-art with large margins.

## 1.6 Structure of the Thesis

In Chapter 2, the background for autonomous driving and collective perception is presented. The chapter begins with a brief history of pioneering work in autonomous driving and early developments in collective perception. Next, it discusses the current state of research in this field, focusing on the three main components of V2X technology: perception sensors, vehicle localization, and communication systems. In the last section, the three main stages of collective perception are introduced.

In Chapter 3, the theoretical foundations relevant to this thesis are presented, covering deep learning modules, components of collective perception, and the localization correction technique. Additionally, commonly used evaluation metrics for object detection and semantic segmentation are discussed.

In Chapter 4, related work about autonomous driving and collective perception is reviewed. The chapter begins by examining limitations in previous driving-space representation methods, including object detection and semantic segmentation. Following this, it introduces perception algorithms based solely on ego-vehicle sensors. Collective perception, as an extension of ego-based perception, presents new challenges such as spatial misalignment of data from multiple IAs due to localization errors, communication delays, and sensor asynchrony, as well as higher computational demands and significant data collection costs. The final section of the chapter discusses state-of-the-art algorithms developed to address these challenges.

Chapter 5 introduces the datasets used in this work, comprising two simulated datasets and two real-world datasets. Chapter 6 begins with a formal mathematical definition of collective perception and then presents an efficient development tool designed for this purpose. Using this tool, comparative experiments in cooperative object detection are conducted to investigate agent-based training methods—which calculate gradients only for specific agents—to reduce computational resource consumption.

In Chapters 7 and 8, the main proposed frameworks for cooperative object detection (COOD), including TA-COOD and BEV semantic segmentation, are explained, respectively. Each task is explained with the detailed architecture of the models, the experiment configurations, and the final evaluation of the results generated by these models.

In the last two chapters, a summary of this work is given in the conclusion part, and an outlook for collective perception is discussed.

# 2 Background

This chapter begins with a brief overview of the historical development of communication technologies relevant to autonomous driving. Next, it introduces the perception system for autonomous vehicles, focusing on the sensors that enable its functionality. Incorporating data sharing among IAs, the perception system could enhance its performance in terms of perception accuracy and reliability, by increasing its field-of-view and decreasing occlusion. The perception system can improve its accuracy and reliability, achieving an expanded field of view and reduced occlusion. However, such cooperation hinges on dependable localization and communication technologies. Section 2.2.2 and Section 2.2.3 provide an overview of these technologies, underpinning the assumptions underlying the experiments conducted in this thesis. Finally, this chapter defines the three implementation stages of collective perception, namely CPM preparation, sharing and fusion.

## 2.1 A Brief History of Communication for Autonomous Driving

Dating back to the 1980s, the German Ernst Dickmanns, widely regarded as the pioneer of AV, achieved a milestone by developing VaMoRs (Versuchsfahrzeug für autonome Mobilität und Rechnersehen by Dickmanns and Zapp (1987)), which was a Mercedes-Benz van installed with an integrated computer and was able to autonomously navigate on traffic-free streets, reaching a speed of 96 kilometers per hour. However, Ernst Dickmanns was not the originator of the concept of autonomous driving. This idea was first envisioned in 1925 by an American inventor Francis P. Houdina, who built what is believed to be the first radio-operated driverless automobile, a Chandler motor car controlled by the radio sent from a second following car. These radio-controlled cars can be considered as driverless cars but not strictly as autonomous cars because of the indirect intervention of humans in the following car. However, it can be regarded as the pioneer of Connected Autonomous Vehicle (CAV) for cooperative driving.

Starting from Futurama (Geddes, 1940), a remotely-controlled car exhibited in 1939's World's Fair, the idea of guiding the driverless car with electric cables beneath the driving tracks was adopted and explored in the following decades by many laboratories, such as RCA Labs (Zworykin et al., 1957), Ohio State University's Communication and Control Systems Laboratory, and the United Kingdom's Transport and Road Research Laboratory. In 1986, Bosch-Blaupunkt introduced one of the earliest on-board navigation systems EVA (Pilsak, 1986) and Carnegie-Mellon University built an autonomous system NAVLAB (Pomerleau, 1988) which utilizes sonar, camera, and laser range finder to find obstacles and achieve self-localization and navigation.

Along with the development of AVs, the route-guidance system system (RGS) via Vehicle-to-Infrastructure (V2I) communication (Wolf and Begun, 1940; Rosen et al., 1970) and its extension to Vehicle-to-Vehicle (V2V) communication (Tsugawa, 1992) was also evolving. The V2I system aims to remotely provide guidance information to the drivers to find correct routes. To enhance flexibility in information exchange and reduce reliance on infrastructure, V2V communication was introduced. V2V enables vehicles to communicate directly with each other anytime, anywhere, offering a more adaptable and decentralized approach to information sharing.

Based on the advancements and accomplishments of V2I and V2V technologies alongside the evolution of vehicle automation, large projects such as PROMETHEUS and DRIVE in Europe (Gillan, 1989), the IVHS (Intelligent Vehicle Highway System) Program in the United States (Sweeney, 1993), and SSVS (Super Smart Vehicle System) in Japan (Tsugawa et al., 1991) were launched in the 1990s, seeking to further explore the potential of intelligent traffic agents and their interaction by leveraging advanced technologies such as modern microelectronics, sensor technology, telecommunications, and informatics. The common objective of these projects was enhancing safety, efficiency, economic viability, comfort, and environmental friendliness.

## 2.2 Autonomous Driving Systems (ADSs)

Beginning with early explorations of driverless and communicative vehicles in the previous century, autonomous driving systems have since undergone rapid evolution, advancing significantly in intelligence and capability in the current century. This can be attributed to the swift progress in sensor systems, computing hardware, and the state-of-the-art algorithms harnessed by Artificial Intelligence (AI) (Poole et al., 1998). Since the initial adoption of Radar for emergency brake assistance in the 1950s (Middlehurst, 2017), the evolution of modern AVs has incorporated a diverse array of sensors within their perception systems. These include cameras, LiDAR (Light Detection and Ranging) sensors, and infrared depth sensors, all of which collectively supply comprehensive environmental data. Over the course of this progression, the introduction of the Global Navigation Satellite Systems (GNSS) has significantly enhanced the localization and navigation capabilities of vehicles. Alongside GNSS, technologies such as the Inertial Navigation Systems (INS) (Zhang et al., 2012; Wahyudi et al., 2018; Kourabbaslou et al., 2019), Real-Time Kinematic (RTK) (Uradziński, 2011; Li et al., 2018), and Simultaneous Localization and Mapping (SLAM) (Durrant-Whyte and Bailey, 2006) are employed to further refine localization accuracy.

With advanced sensors and the localization and navigation systems, the AI-based egocentric-ADS has successfully passed the driving test in some specified scenarios. To enhance safety further, communication technologies for Vehicular Ad Hoc Networks (VANETs) have been introduced to alleviate problems with occlusions and extend the observation range. This thesis concentrates on the LiDAR-based collective perception system with considering the challenges introduced by sensor synchronization, localization errors, and communication latency. In the following, technologies related to these issues and the reason behind the choices of experimental settings in Chapter 7 and Chapter 8 are introduced.

### 2.2.1 Sensors and perception

Perception is the process of understanding the environment via sensing. By processing the sensed data, the perception system should interpret and represent the data into a higher semantic level so that this information can be utilized for making driving decisions. To make a correct decision, the vehicle should know its own position relative to the map for navigation and the positions of other traffic participants to avoid collisions.

Among all available sensors, cameras stand as the predominant sensors for ADSs due to their ability to capture rich environmental data and their cost-effectiveness. Several cameras are mounted on the vehicle to capture RGB images of the surrounding environment. These images are processed by deep-learning (DL) models and interpreted into information of higher semantic level that is easier to be used by the computer for decision making. For instance, these DL models can identify the traffic participants within the 2D images. Combining the geometries, the calibration parameters, of several cameras, and the 3D locations of these detected participants can be computed. With this information, the ego-vehicle can avoid collisions. By interpreting road geometry from the images and assessing the detected objects' relative positions in relation to the ego-vehicle, the autonomous driving system is able to plan the trajectory for its subsequent actions.

While camera-based perception systems serve as the "eyes" for driving tasks, their accuracy in locating objects, particularly those at a distance, remains limited, because the camera-based approach lacks inherent depth information and relies on subsequent algorithmic estimations. It indirectly queries 3D information either by estimating the depth information or by the geometries of several cameras; therefore, it is more sensitive to inaccuracies arising from sensor calibration. These errors can propagate and potentially magnify during the algorithmic process of projecting 2D points into 3D space. Additionally, their effectiveness notably diminishes in inadequate lighting conditions. In contrast, a LiDAR sensor actively sends and receives laser beam signals for detecting distances relative to the obstacles, provides accurate measurements in a few hundred meters, and operates independently of lighting conditions. Moreover, LiDAR directly captures real-world 3D information without the algorithmic projection from 2D to 3D. Infrared depth sensors can also be utilized as compensation for cameras, but their accuracy and detection range fall far behind the LiDARs. Due to these advantages, LiDAR can be regarded as essential for safer autonomous driving. Therefore, this thesis focuses on building perception frameworks that process point cloud data captured by LiDARs for the two perception tasks: Bird's Eye View (BEV) semantic segmentation and object detection.

Taking the point clouds as input, object detection aims to find the pre-defined target objects in the driving environment that might cause collisions. Because the on-ground vehicles are driving approximately on the 2D BEV surfaces, for simplicity, these surfaces are interpreted by BEV semantic segmentation as a 2D BEV semantic map. Each pixel on this map is assigned a semantic label corresponding to its classification. Based on these semantic labels, the IVs are able to navigate themselves in the 2D driving environment.

### 2.2.2 Localization

Exact localization of IAs is a prerequisite for successfully aligning and fusing the features from different IAs. Consequently, localization errors pose a significant challenge within the collective perception framework. It is imperative that this framework minimizes the impact of such errors as much as possible. The following discussion provides an overview of the current state of localization technologies to underpin the assumptions related to localization in the experimental context.

As an essential component for vehicle navigation systems, the GNSS-based localization accuracy ranges from a few meters to over 20 meters (Tan and Huang, 2006) depending on the visible number of satellites and the weather and environmental conditions. It suffices for human drivers to ascertain their location and plan their route. Yet, for the autonomous driving systems, centimeter-level precision, or at least accuracy within a few decimeters (∼30cm) is essential to ensure accurate positioning within road lanes. To attain this level of precision, techniques such as INS, RTK, and SLAM can be used on top of GNSS.

INS operates based on the principles of inertia, utilizing sensors to track a vehicle's motion. It can be used to correct the positioning and navigation information by fusing GNSS information with the velocity and orientation measured by high-accuracy gyroscopes and accelerometers. However, the errors of INS drift over time. Similarly, RTK also combines GNSS information with additional correction data to improve the localization accuracy. This correction data, which includes precise satellite positions, is transmitted from base stations with known locations. However, a significant limitation of RTK is its dependence on communication with these base stations within a restricted range, due to limited signal strength. Consequently, RTK is often ineffective in areas where signal obstruction occurs, such as tunnels, areas under dense tree canopies, and regions affected by multi-path errors caused by nearby buildings (Uradziński, 2011). Differently, SLAM constructs real-time local maps autonomously by matching the information of consecutive frames of data measured by onboard sensors, enabling the vehicle to navigate based on its relative position within these maps. When provided with a global reference map, the vehicle determines its global location by aligning and registering the local map into the global map. Sensors such as short-range Radar (Ward and Folkesson, 2016), Localizing Ground-Penetrating Radar (LGPR) (Cornick et al., 2016; Ort et al., 2020), camera (Parra et al., 2010; Li et al., 2013; Suhr et al., 2017) as well as LiDAR (Wei et al., 2020; Levinson and Thrun, 2010; Elbaz et al., 2017; Wang et al., 2019) can be used to improve the localization accuracy. In general, LiDAR-based solutions achieve the best performance due to their high-accuracy measurements. For instance, by registering the road geometric elements, such as curbs and markings detected from point clouds data, into the global feature map of the environment, the lateral and longitudinal errors can be reduced to less than 30 cm. Similarly, matching deep features that are generated by DL models can also achieve the localization accuracy under 30 cm (Elbaz et al., 2017; Wang et al., 2019).

In this thesis, it is assumed that state-of-the-art algorithms are employed to deliver localization with centimeter-level accuracy, facilitating effective data fusion for collective perception.

### 2.2.3 Communication

To facilitate communication among traffic agents, various communication technologies are employed based on specific scenarios. Among these scenarios, vehicles play the main role with their misconduct emerging as the most crucial factor contributing to traffic accidents (Statistisches Bundesamt, 2017). In the pursuit of enhancing road safety and optimizing traffic services, the concept of Vehicular Ad Hoc Networks (VANETs) was initially introduced in 2001. This innovative approach facilitates communication and networking among vehicles, with autonomous vehicles within this network referred to as Connected Autonomous Vehicles (CAV). Under the framework of VANET, there are three types of communication scenarios, namely Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Everything (V2X).

V2V communication, also called Inter-Vehicle Communication (IVC), allows information exchange among vehicles. For instance, they can share accident, congestion, and speed limit information to improve traffic safety and efficiency (Arena and Pau, 2019). V2I communication handles the interaction between the vehicles and the RSUs, hence also named as Roadside-to-Vehicle Communication (RVC). The communication in this scenario is bidirectional. From the vehicle side to the roadside, the traffic information in a broader range and better view-of-sight can be shared. Take a busy traffic intersection as an example; infrastructure equipped with high-mounted cameras can provide more comprehensive observations of the overall traffic conditions compared to relying solely on the on-board cameras of vehicles. Sharing the information about the traffic situation from the infrastructure significantly improves the vehicles' ability to understand this environment and make better driving decisions accordingly. In the opposite communication direction, the RSUs can collect information from the vehicles, update the cloud database in real-time, and adjust the supervision and management of the traffic accordingly (Arena and Pau, 2019), benefiting all end users in the VANET. By integrating V2V and V2I communications, while also taking into account vulnerable road users and other connectable devices, V2X technology enables vehicles to communicate with any traffic entities that may influence or be influenced by vehicular activity (Lee et al., 2017). For instance, communication with vulnerable road users, such as pedestrians and cyclists, can prevent collisions with them (Lin et al., 2015).

| **Range** | **tech.** | **standard** | **DTR(bps)** | **range(m)** | **latency(ms)** |
|---|---|---|---|---|---|
| Short | Bluetooth | IEEE 802.15.1 | 1M–3M | 10 | 100 |
| | BLE | IEEE 802.15.1 | 1M | 50 | 6 |
| | ZigBee | IEEE 802.15.4 | 20K–250K | 75–100 | 30 |
| | UWB | IEEE 802.15.3 | 480M | 75 | 0.1 |
| Medium | DSRC/WAVE | IEEE 802.11p/1609.4 | 27M | 300 | 100 |
| Long | LTE | 3GPP Rel. 14 | 17.7M | 100– >5K | < 10 |
| | 5G-NR | 3GPP Rel. 15 | >1G | 50– >5K | < 1 |

*Table 2.1: VANET communication technologies classified with communication ranges.*

To implement the aforementioned communication scenarios, various communication technologies are employed. Based on their communication ranges, these technologies can be categorized into short, medium, and long ranges (Ahangar et al., 2021). Table 2.1 includes examples of exemplary standardized technologies within each range class. The key features that are relevant to the data fusion process of collective perception are listed.

Communication **latency** refers to the time delay that occurs between the moment a signal is transmitted and when it is received. A low latency is important for perception systems because it requires the shared information to be as fresh as possible to accurately localize the dynamic objects in the driving environment. If the latency is too high, the data arriving at the goal agent would be outdated. This introduces challenges to predict accurate locations of the dynamic objects in real-time. To ensure low latency, the data transfer rate (**DTR**) also plays a critical role. It reflects the number of bits that can be transferred per second. The DTR should match or be larger than the communication workload to prevent congestion in the communication channels. Once the communication traffic is jammed, the data transfer might consume much more time than the lowest latency the communication can provide. At last, the communication **range** is provided in the table as it holds significance for the respective use cases. For example, Bluetooth may not be suitable for V2I communication, as vehicles are typically situated at a longer distance from the infrastructure elements than the communication range of Bluetooth. In Table 2.1, the listed DTRs and ranges are the higher boundaries; the latency is the lowest transmission time required if the communication channels are operating in the most efficient mode. The standardized protocols are also given as reference to retrieve more details regarding the corresponding techniques.

In general, the technologies with very short communication ranges, such as Bluetooth and BLE (Bluetooth 4.0), are not suitable for inter-agent communication. However, they can be used for intra-vehicle communication of on-board sensors to simplify the complexity of wiring in the vehicle. ZigBee and Ultra Wide Band (UWB) signals can reach relatively longer distances, having the potential for inter-agent data transmission. Especially, when multi-hop communication is applied, more distant agents are interconnectable. However, ZigBee is not suitable for crowded areas because of its low DTR. In contrast, UWB uses a wide bandwidth from 3.1 to 10.6 GHz (Ahmed et al., 2015), enabling heavy data exchange. Besides, it can penetrate obstacles, assisting the localization in dense environments (Martín et al., 2020). As a medium range communication technology, Dedicated Short-Range Communication (DSRC) under the standard IEEE 802.11p is modified from WiFi technology (IEEE 802.11). Wireless Access in Vehicular Environment (WAVE) under IEEE 1609.4 is an extension of DSRC, supporting the co-existence of safety and non-safety applications over DSRC channels. They are specially designed to enable reliable communication for both V2V and V2I, providing information exchange, such as traffic signals and accident alerts (Zeng et al., 2009; Kenney, 2011), in a wide area network. Despite increased DTR, DSRC/WAVE can hardly be used in dense traffic because of congestion issues caused by the absence of optimized channel access control (G. and R., 2018). Besides, with DSRC's range limit of 300 m, the vehicles running at high speed need to change network topologies frequently, leading to handover problems (Abboud et al., 2016). To solve these problems and facilitate large-scale V2X communication, the Third Generation Partnership Project (3GPP) group has defined the long range C-V2X technology in the technical report 3GPP Release 14 [1]. It introduced the LTE-based C-V2C technology (Moradi-Pari et al., 2023). However, with the rapid advancement of information and computer technologies, intelligent agents now have the capability to process large volumes of data, enhancing their perception performance. Consequently, communication technologies must also be capable of managing massive data exchanges to effectively support these agents. To this end, 5G technology is introduced in the latter releases of 3GPP standards. For instance, 5G-NR increases the DTR to Gigabyte level

---

[1] 3GPP Rel.14: TR21.914. `https://www.3gpp.org/specifications-technologies/releases/release-14`

and reduces the latency to less than 1 ms, enabling more advanced collective perception systems by sharing richer and more diverse information.

## 2.3 Implementation of Collective Perception

### 2.3.1 CPM preparation

Collective perception can be achieved through a three-step process: preparation, sharing, and fusion of CPMs. In the preparation phase, any information that may benefit recipients is identified and prepared for dissemination. However, due to limited communication bandwidth, it is essential to prioritize and share only the most critical information. This raises the question: *what should be shared?* An effective strategy is therefore needed to identify and select the most important information for transmission.

| Level | Name | Possible processing procedures |
|---|---|---|
| 1 | Data acquisition | LiDAR, camera, radar |
| 2 | Preprocessing | down-sampling, transformation |
| 3 | Features | normals, curvatures, deep features |
| 4 | Analysis | object detection, semantic segmentation, tracking |
| 5 | Understanding | threat/safety assessment |
| 6 | Decision | navigation, steer, brake, speed |

*Table 2.2: Definition of different data processing levels*

Inspired by Hall and Llinas (1997), the content of CPMs can be classified according to their processing levels as shown in Table 2.2. As the information is distilled to a higher processing level, it contains less detail and gets smaller in data size. In the data acquisition level, raw data from various sensors—including LiDAR, cameras, and radar—can be shared to provide recipients with complete sensory information from cooperating sources. However, sharing raw data typically demands substantial communication bandwidth. For example, the raw point cloud data captured by LiDARs may contain millions of 3D points. At a capturing frequency of 10 Hz, it generates tens of Megabytes of data per second, saturating the communication channels.

After data acquisition at level 1, the raw data can be down-sampled to a smaller size and transformed to required coordinates before sharing. For example, by receiving the location of an ego-vehicle, the cooperative vehicle can transform the point cloud data to the ego-vehicle's coordinate system or down-sample the point cloud to voxels (Chen et al., 2019a) before sharing. Also, the data can be further processed to obtain features in level 3, such as normals and curvatures, calculated from the neighborhood points in the point cloud or so-called deep features learned by a Deep Neural Network (DNN) (Marvasti et al., 2020a; Chen et al., 2019b; Wang et al., 2020a). Starting from level 4, the data become lightweight, only requiring little communication resources.

Despite the success of early works (Allig et al., 2019; Aoki et al., 2020; Niels et al., 2019) on improving perception performance by sharing high-processing level messages, they might be error-prone and task-specific, and cannot be generalized for use on other tasks. Instead, the mid-level messages, especially the learned deep features (Marvasti et al., 2020a; Chen et al., 2019a; Xu et al., 2022c; Wang et al., 2020a), are proven to be beneficial in balancing the perception performance and the communication bandwidth requirements.

This thesis takes LiDAR as a focus because of its high accuracy in measuring 3D environments and better reliability under different light conditions. The CPMs are prepared at level 3. More specifically, the local point cloud data captured by LiDAR is processed by a DNN to extract deep features for sharing.

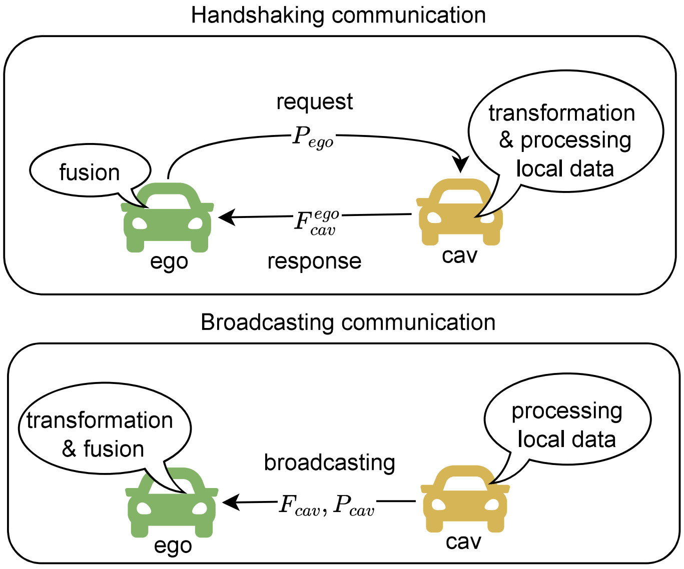


*Figure 2.1: CPM sharing strategy.*

### 2.3.2 CPM sharing

As illustrated in Figure 2.1, there are mainly two strategies for sharing the prepared CPMs: handshaking and broadcasting communication. By handshaking, the ego vehicle first sends a request CPM containing its own pose $P_{ego}$ to the cooperative vehicle. Then the cooperative vehicle transforms its local raw data into the coordinate system of the ego-vehicle and processes the transformed data $F^{ego}_{cav}$ for sharing. Instead, in the broadcasting communication, the cooperative vehicle directly sends its local processed data $F_{cav}$ and its pose $P_{cav}$ to the ego-vehicle. The coordinate transformation of the data is accomplished at the recipient. The handshaking approach simplifies data fusion but places a higher demand on communication resources, while the broadcasting strategy is more communication-efficient but requires feature transformation at the fusion stage. This thesis employs both strategies, assuming ideal communication with 100% throughput rather than simulating the actual communication process.

### 2.3.3 CPM fusion

To fuse the received CPMs into the ego-vehicle's local coordinate system, two main challenges must be addressed: inaccurate poses of IVs (Wang et al., 2020a; Xu et al., 2022a; Yuan et al., 2022; Yuan and Sester, 2022) and time asynchrony due to communication latency (Xu et al., 2023; Yu et al., 2023) and asynchronous sensor clocks. Pose errors make it difficult to accurately align features from cooperative vehicles within the ego-vehicle's coordinate system, which can degrade cooperative perception performance and even negatively impact the ego-vehicle's own local perception. Furthermore, time asynchrony results in outdated CPMs that do not accurately reflect the real-time positions of the detected target objects. This thesis explores all these challenges while developing and evaluating the perception frameworks.

# 3 Theoretical Fundamentals

This thesis employs deep learning techniques to construct the collective perception framework. Accordingly, the relevant deep learning modules are first introduced in Section 3.1. Subsequently, the main network components, task heads, and loss functions utilized in this work are described in detail in Section 3.2. Additionally, the pose graph optimization algorithm is discussed in Section 3.3, as it plays a critical role in enabling localization correction within the framework. Finally, the commonly used evaluation metrics for object detection and BEV semantic segmentation tasks are defined in Section 3.4.

## 3.1 Deep Learning Modules

### 3.1.1 Multi-layer perceptrons

Multi-Layer Perceptrons (MLPs), also called feed-forward neural networks, consist of fully connected neurons with a kind of activation function, such as *sigmoid*, *tanh*, *ReLU* (Rectified Linear Unit), and *softmax*. A layer of fully connected neurons is called a fully connected layer in which each input neuron is connected to each output neuron. A MLP contains one input layer, at least one hidden layer, and one output layer. An example of MLP with two fully connected layers is shown in Figure 3.1. The outcome of each neuron in the hidden layers and the output layer is the weighted sum of the outcomes of the previous layer. In addition, a constant bias value can be added to this sum, as the $b_1$ and $b_2$ shown in Figure 3.1. Mathematically, the MLP can be expressed with

$$\mathbf{y} = f^3(f^2(f^1(\mathbf{x}))) \tag{3.1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and the output vector, respectively. $f^*$ is the function of the fully connected layer which can be formulated as

$$h_j^l = \sigma(\sum_i (x_i w_{ij}) + b_j) \tag{3.2}$$

where $x_i$ is the outcome value from the $i$th neuron in the previous $l-1$th layer, $w_{ij}$ is the weight between the $i$th previous neuron and the $j$th neuron in the current $l$th layer, $b_j$ is the bias at the $l$th layer, $\sigma$ is the activation function, and $h_j^l$ is the outcome value of the $j$th neuron in the $l$th layer.

MLPs are of extreme importance for building neural networks. In this work, they are used for encoding the input point features and position and time embeddings for transformers.

### 3.1.2 Convolutional neural network

Convolutional Neural Networks (LeCun, 1989), or CNNs, are a class of neural networks that are specialized for processing data with grid-like topology (LeCun et al., 2015), such as images. Compared to MLPs, CNNs contain a series of convolution layers that are more efficient as each convolution operation only covers a portion of the input data. More specifically, each convolution layer performs a series of convolution operations between two matrices, where one matrix is the restricted portion of the input matrix $I$, also called receptive field (Figure 3.2 green or dashed blue box over input grid) and one matrix is a set of learnable parameters that are arranged in grid $K$, also called kernel or filter (Figure 3.2 red box). Mathematically, this operation can be expressed as

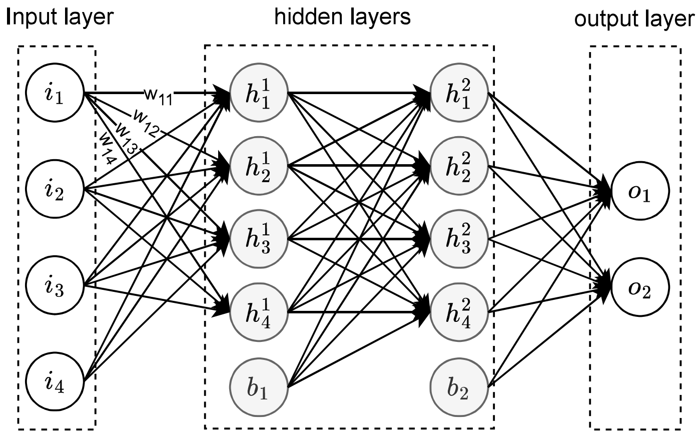
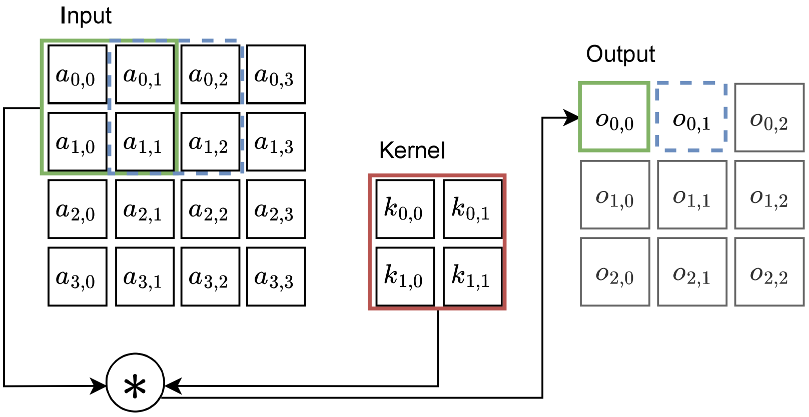


*Figure 3.1: Multi-Layer Perceptron*



$$o_{0,0} = a_{0,0} \cdot k_{0,0} + a_{0,1} \cdot k_{0,1} + a_{1,0} \cdot k_{1,0} + a_{1,1} \cdot k_{1,1}$$

$$o_{0,1} = a_{0,1} \cdot k_{0,0} + a_{0,2} \cdot k_{0,1} + a_{1,1} \cdot k_{1,2} + a_{1,1} \cdot k_{1,1}$$

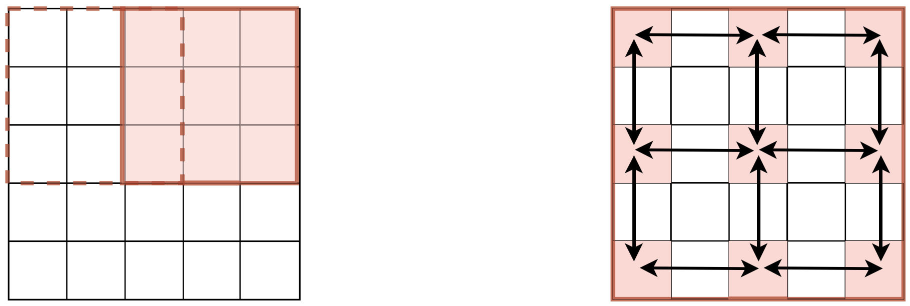
$$\vdots$$

*Figure 3.2: An example of a two-dimensional convolution*

$$o_{i,j} = (K * I)(i,j) = \sum_m \sum_n a_{i+m,i+n} \cdot k_{m,n} \tag{3.3}$$

where $a_{i+m,j+n}$ and $k_{m,n}$ are the elements in the input matrix $I$ and kernel matrix $K$, respectively. The indices $i$ and $j$ indicate the convolution location. Since the same operation with Equation (3.3) is applied at each input location, this series of operations can be computed in parallel so that less processing time is required. In Figure 3.2, the input data is a $4 \times 4 \times 1$ tensor, and the kernel is $2 \times 2 \times 1$, where the last dimension is the number of channels, also called depth. In real use cases, the kernel is smaller in size than the input data but normally has more channels. For example, an input image normally is composed of three (RGB) channels. One may use more than three kernels to perform convolution over each channel of the input image and take the summation of the outputs over all channels as the final output of each kernel. As a result, the final output has the dimension of $W \times H \times C$, where $W$ and $H$ are the width and height of the output tensor, respectively. $C$ is the number of kernels or output channels.

To reduce computational demands, the convolutional kernel can move with larger step sizes instead of a step size of one, as shown in Figure 3.2. This step size is referred to as the *stride*.



*(a) Kernel size 3, stride 2 and dilation 1.*

*(b) Kernel size 3, stride 1 and dilation 2.*

*Figure 3.3: Variants of convolution kernels.*

Figure 3.3a illustrates an example of a convolutional kernel with a size of 3 and a stride of 2, where the kernel window shifts forward by two pixels at each step (the dashed red box moves to the new position shown by the solid red box). In addition to adjusting the stride size, the kernel elements can be arranged sparsely across the input to increase the receptive field without increasing computational load. This variant, known as *dilated convolution*, involves dilating the kernel at a specified ratio. Figure 3.3b depicts a kernel of size 3, stride 1, and dilation 2, where the kernel elements (red squares) are spaced out and operate on every second input element. The black arrows indicate the dilation ratio, which is two in the example.

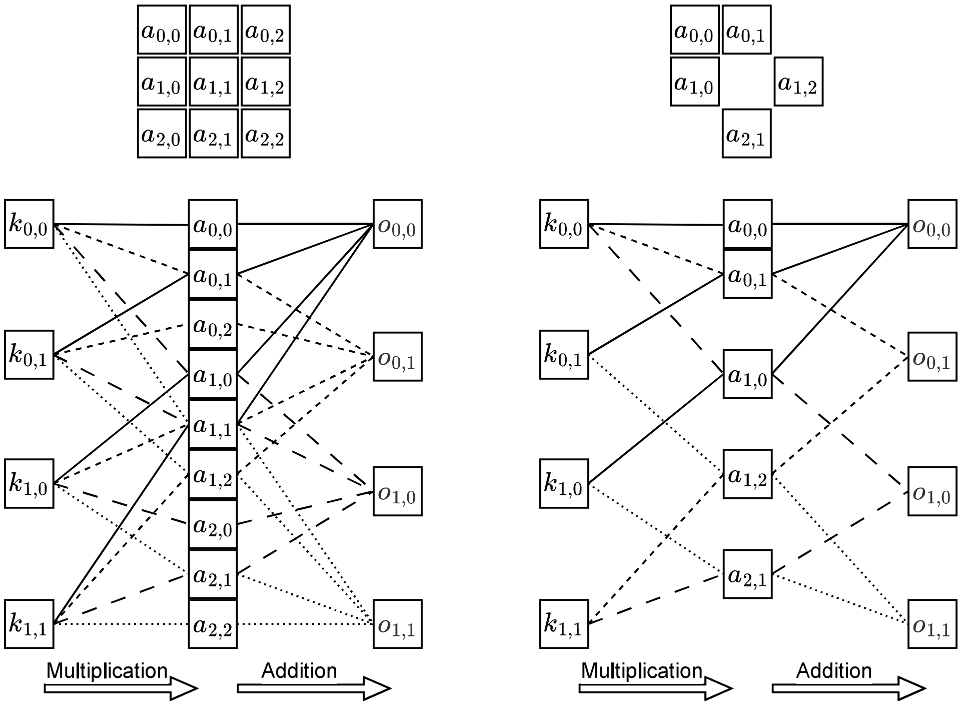
On top of each convolutional layer, which is a linear operation, an activation function is always attached to introduce non-linearity so that the convolution layers can learn more complex non-linear features. *ReLU* and its variants, such as *leaky ReLU*, are the most commonly used activation functions because of their computational simplicity and representational sparsity. In addition, the CNNs may also contain normalization layers, which re-center and re-scale the output data after each convolutional layer to accelerate and stabilize the training process of the network. To simplify the explanation, when the term *convolutional layer* appears in the following chapters, it refers to the convolution layer attached with the normalization layer and the activation function.

To apply the convolutional layers on LiDAR data, the 3D point clouds captured by LiDAR should be rasterized into small units, which are called voxels. This process of rasterization is called voxelization. The outcome is a 4D tensor of shape $X \times Y \times Z \times C$, where $X$,$Y$,$Z$ are the discretized lengths along the $x$, $y$ and $z$ axes in the 3D world, respectively. The input channel $C$ normally contains the aggregated features from the points in each corresponding voxel. For example, the averaged coordinate and intensity values of the points in that voxel. Since the point cloud data are very sparse, dense 3D convolutions over the 4D tensor ($X \times Y \times Z \times C$) waste a lot of computational resources due to unnecessary calculations in empty spaces. Therefore, this thesis uses fully sparse convolutions.

### 3.1.3 Sparse convolution

The convolution layers can be extended to any dimension. However, the computational demands increase exponentially as the dimension increases. Modern computers are able to process 3D convolutions only on very limited input data size, such as cropped point clouds in a limited field of view or discretized with larger voxel size. To increase the computational efficiency, sparse convolution (Graham and der Maaten, 2017) is introduced to deal with the convolution operations over point clouds that are represented with sparse tensors of shape $N \times C$, where $N$ is the number of voxels and $C$ the number of features for each voxel.

The conversion from dense convolution to sparse convolution is simply a technical trick that avoids unnecessary calculations over empty spaces. Figure 3.4 shows the differences. The left subfigure Figure 3.4a demonstrates atomic operations (multiplication and addition) of the dense



*(a) Dense convolution* *(b) Sparse convolution*

*Figure 3.4: Atomic operations of convolutions*

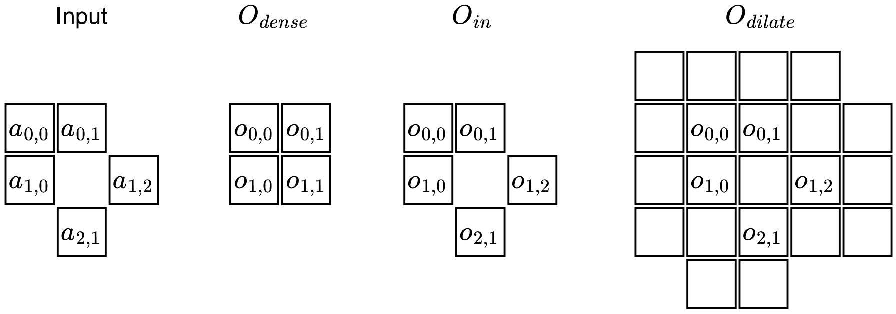


*Figure 3.5: Different output variants of sparse convolutions*

convolution. To utilize the parallel computation of GPUs, a *kernel mapping function* that specifies the operand addresses or indices for each atomic operation of convolutions needs to be defined. As shown in Figure 3.4a, the kernel, input, and the output matrix are first stretched to an array. Then the kernel parameters are multiplied with the input elements, where the multiplication results are finally added to the corresponding output addresses. In this case, the kernel mapping function is represented with the lines that connect the data point of the kernel, the input, and the output element addresses. Each type of line represents one convolution operation for one output element. If the input is sparse as shown in Figure 3.4b, the atomic operations are significantly fewer than for the dense convolution.

Because of the flexibility of defining the kernel mapping functions, the output shape can also be flexibly adjusted as illustrated in Figure 3.5. The example in Figure 3.4b results in an output $O_{dense}$ of shape 2 × 2. To keep the sparsity of the input data, the kernel mapping functions can be defined to only calculate the convolution result for the output locations that are aligned with the input locations, resulting in $O_{in}$. This output format is regarded as the standard sparse

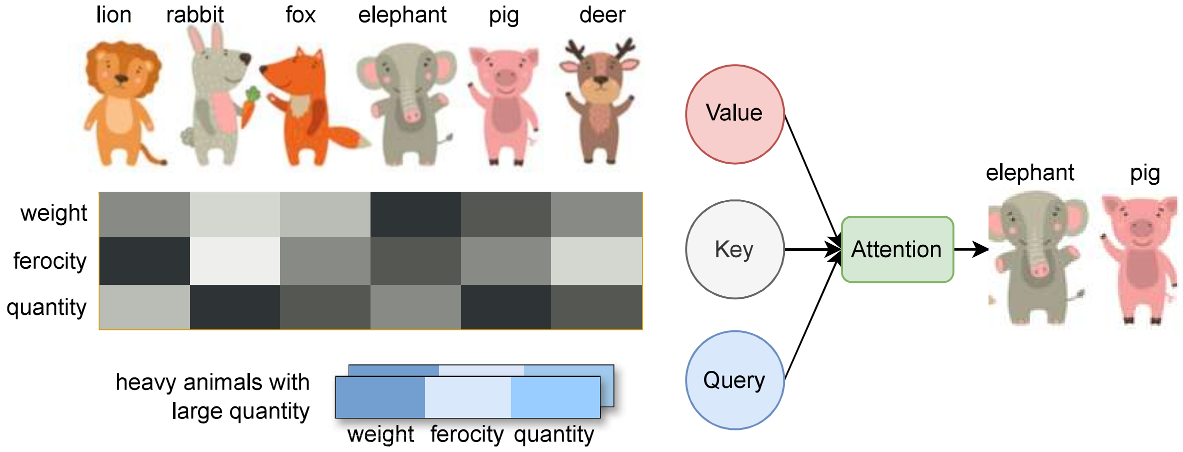


*Figure 3.6: Attention mechanism for retrieving information from an animal database*[1]

convolution. However, if the data is highly sparse, an isolated location is hard to learn from the features of distant locations in the input as the kernel window can never cover other input elements. To this end, *Coordinate-Expandable sparse Convolution (CEC)* is introduced to expand the input tensor coordinates and obtain the dilated output tensor $O_{dilate}$. All three output formats can be utilized alternatively to balance between the computational complexity and the learning ability of the models.

### 3.1.4 Attention mechanism and transformers

Attention is one of the most important cognitive mechanisms of humans for efficient perception. This mechanism tends to selectively process the most important information and ignore the other information. In machine learning, attention can be used to efficiently allocate the computational resources and only concentrate on learning the relevant features that are beneficial for generating the target results. It has been widely used in the machine learning tasks that require processing heavy data in grid-topology, such as computer vision (Itti et al., 1998; Mnih et al., 2014), or long sequential data such as machine translation (Bahdanau et al., 2015; Sutskever et al., 2014) and speech recognition (Chorowski et al., 2014). The attention mechanism can significantly reduce the computational overhead of these tasks. Besides, incorporating the positional encoding that encodes the spatial and temporal information into the learned deep features, the attention mechanism can process sequential data in parallel instead of processing them recurrently in order to build the temporal context. This significantly increases the data processing efficiency. Therefore, this thesis uses the attention mechanism to process the sequential point-cloud data.

Theoretically, the general attention mechanism in machine learning makes use of three components: keys $K$, values $V$, and queries $Q$. For understanding, the interplay among these three components can be compared to a search process in a text corpus or database, as the example shown in Figure 3.6. The two queries give the instructive information *heavy animals with large quantity* for the search process and interact with the keys, which can be regarded as a compact summary of the features of values, to find the importance for each value so that the weighted sum of the values is retrieved as the final attention result. As the example shows, the given two queries indicate that values (animal features) with larger weight and quantity in keys are of greater importance. As a result, the elephant and pig are retrieved from the database because the weight is the

---

[1] Database: `www.kaggle.com/datasets/iamsouravbanerjee/animal-image-dataset-90-different-animals`

most important factor in the query (darkest blue) and the quantity is the second most important factor.

This instructive information of queries can be encoded into a mathematical representation – a vector. In the example, two such queries with three features are given for searching two items that fulfill this instruction. In general, the tensor shape of queries can be written as $L \times d_k$, where $L$ is the number or length of queries and $d_k$ is the number of query features, also called feature embedding dimension. The keys $K$ and values $V$ have the tensor shape of $S \times d_k$ and $S \times d_v$, respectively. In the example, $L = 2$, $S = 6$ and $d_k = d_v = 3$. Mathematically, the attention can be expressed as

$$Attention(Q,K,V) = W \cdot V = softmax(Q \cdot K^T) \cdot V \tag{3.4}$$

where the weight $W$ for the values $V$ is calculated by performing a softmax function over the last dimension of the dot product between $Q$ and the transpose of $K$. This attention realization is called *Dot-Product Attention*. According to Vaswani et al. (2017), this type of attention tends to generate dot products of large magnitude, pushing the softmax function to generate extremely small gradients during training, hence deteriorating the model performance. To this end, the *Scaled Dot-Product Attention* is introduced by scaling the dot product with a factor $1/\sqrt{d_{qk}}$. The new attention function is then updated to

$$Attention(Q,K,V) = W \cdot V = softmax(\frac{Q \cdot K^T}{\sqrt{d_{qk}}}) \cdot V \tag{3.5}$$

When $L \neq S$, the attention is also called *Cross Attention*. However, in some cases, the same input might be respectively used as $K$,$V$ and $Q$ for the purpose of learning the sequence representation by exchanging information and learning the relationships between the elements in the same sequence. This is called *Self-Attention*. For simple implementation, attention normally uses the same embedding dimension $d = d_k = d_v$ for all inputs.

To improve the learning ability of the attention layers, Vaswani et al. (2017) proposed to project the $Q$,$K$,$V$ with linear layers into $h$ different representation subspaces and learn more diverse features from these spaces. Each attention head learns from one subspace, all $h$ attention heads are performed in parallel. This version of *Mult-Head Attention (MHA)* is formulated as

$$MultiHeadAttn(Q,K,V) = Concat(head_1, \ldots ,head_h)W^O \tag{3.6}$$

$$head_i = Attention(QW_i^Q,KW_i^K,VW_i^V) \tag{3.7}$$

where the learnable parameters $W_i^Q \in \mathbb{R}^{d \times d_k}$, $W_i^K \in \mathbb{R}^{d \times d_k}$, $W_i^V \in \mathbb{R}^{d \times d_v}$ project the input features into different subspaces of reduced dimensions $d_k = d_v = d/h$. Each head has its own projection parameters. The attended feature $head_i$ has the shape of $S \times d_v$. Features from all $h$ heads are concatenated along the embedding dimension and projected with $W^O \in \mathbb{R}^{d \times d}$ to obtain the final attention result.

Based on MHA, Vaswani et al. (2017) designed a transduction model, called *Transformer*, for learning the representations of sequential data in parallel, significantly reducing the training time. The overall architecture of the transformer is demonstrated in Figure 3.7. It is mainly composed of an encoder and a decoder, each of which contains $N$ identical layers. Each layer in the encoder contains a MHA and a fully connected feed-forward sub-layer, around which a residual connection (He et al., 2016) with the layer normalization (Ba et al., 2016) is employed. The decoder has a similar structure; however, with one more MHA sub-layer inserted. The bottom masked MHA layer ensures that each position in the sequence only attends to its preceding positions. The output of this layer is then updated by attending to the output of the encoder stack in the following

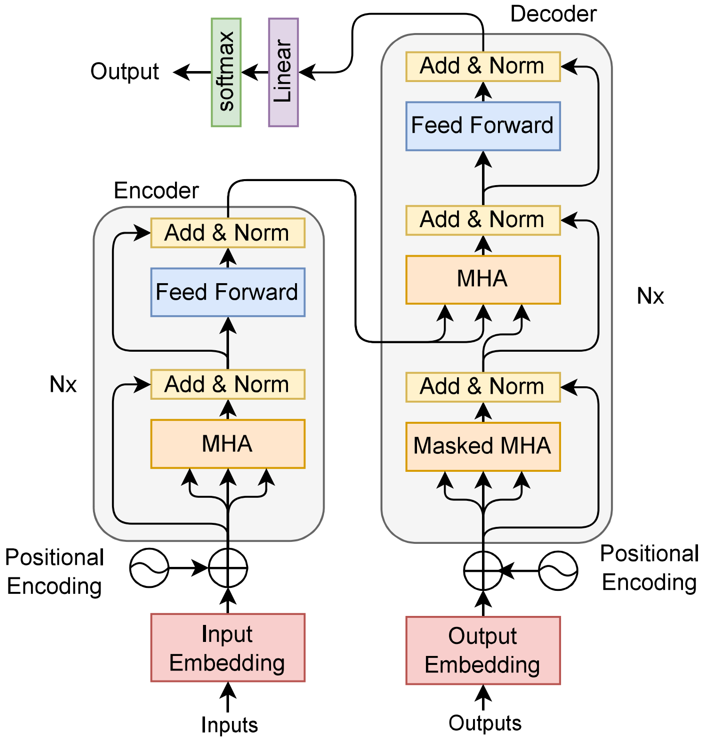


*Figure 3.7: The overall architecture of transformer*

MHA layer. The output of the feed-forward sub-layer in the last decoder layer is finally fed to a linear layer followed by a softmax function to obtain the output probabilities. At the bottom of the illustrated encoder and decoder (Figure 3.7), the input and output tokens are converted to vectors of dimension $d$. These embedded vectors, also called *Embeddings*, are then added with a positional encoding to make them aware of the token locations in the sequence while performing the attentions.

The positional encoding encodes the location information into the same dimension $d$ as the embeddings. It plays an important role in the transformer for identifying the spatial and temporal correlations between different tokens. The most commonly used position encoding is the *sin-cos position encoding* as described with Equation (3.8). Inspired by the binary representation of numbers (Kazemnejad et al., 2023), it forms the encoding vector $\overrightarrow{PE_t} \in \mathbb{R}^d$ as a geometric progression from $2\pi$ to $T \cdot 2\pi$ on the wavelengths (Vaswani et al., 2017), where $T$ is the encoding temperature which is set to 10000 by default, $i \in \{0, \ldots, d-1\}$ is the dimension index and $t \in [0, 2\pi]$ is the position of the token in the sequence. The frequency $\omega_k$ decreases with the encoding dimension.

$$\overrightarrow{PE_t^i} = f(t)^i := \begin{cases} sin(\omega_k \cdot t), \text{ if } i = 2k \\ cos(\omega_k \cdot t), \text{ if } i = 2k+1 \end{cases} \tag{3.8}$$

$$\omega_k = \frac{1}{T^{2k/d}} \tag{3.9}$$

## 3.2 Components of Collective Perception

### 3.2.1 Backbone neural networks

In this work, three backbone networks are used to extract the point cloud features. *VoxelNet* (Zhou and Tuzel, 2018) is an extension of a 2D CNN to process 3D data. To reduce computational demands, *PointPillar* (Lang et al., 2019) proposes to directly encode point clouds into 2D feature maps using linear layers and then only use 2D convolutions for feature extraction. While VoxelNet uses dense 3D convolutions to obtain better 3D features and *PointPillar* uses 2D convolution to be efficient, the fully sparse convolutional network, *MinkUnet* can be regarded as a compromise between feature extraction performance and computational efficiency. For all these networks, point clouds should first be encoded into voxels or pillars as input for convolutions. In the following, the voxel encoding process will be explained first. The details of the selected backbone networks are then discussed.

**Voxel encoding**

In order to process the point cloud data with convolutional layers, the unordered points in the point clouds should be first discretized into voxels, which are the elements of 3D grids. To formulate this voxelization process, the following assumptions are made:

- The input point cloud ($PCD$) is represented with a 2D tensor of shape $N_{pcd} \times C_{pcd}$, where $N_{pcd}$ is the number of points and $C_{pcd}$ is the number of point features, including the xyz-coordinates $p_x$, $p_y$, $p_z$, the intensity $p_\iota$ and the timestamp $p_\tau$ if given.
- The observable 3D space within the ranges $[X_{min}, X_{max}]$, $[Y_{min}, Y_{max}]$ and $[Z_{min}, Z_{max}]$ respectively for the $x, y, z$ axes, is rasterized with the voxel size of $vs_x \times vs_y \times vs_z$ into a 3D grid of shape $X \times Y \times Z$.
- Only the nonempty voxels $\mathbf{v}$ that contain $1 \leq m \leq M$ points are retained for further processing. The total number of nonempty voxels is notated as $N_v$. Each point $v^i$ in $\mathbf{v}$ has $C_{in}$ input features that are derived from the original PCD features.

In this work, three voxel feature encoding (VFE) methods are used, namely *mean VFE*, *linear VFE* with one linear layer and *MLP VFE* with three fully connected layers. The mean VFE takes the mean features of all the points in the voxel as the final features for this voxel. It is the default VFE for the *SpConv* backbone network. Linear VFE is the default method for the *PointPillar* and the *VoxelNet* backbone network and MLP VFE is for *MinkUnet*. Both linear and MLP VFEs compose the feature vector $f_v^i$ for the point $v^i$ by concatenating features of the absolute coordinates $p_x^i, p_y^i, p_z^i$, the point intensity $p_\iota^i$, the relative coordinate within the voxel $\tilde{p}_x^i, \tilde{p}_y^i, \tilde{p}_z^i$, the center coordinates of the voxel $v_x^i, v_y^i, v_z^i$. This representation is formulated in Equation (3.10) and Equation (3.11), where $|\mathbf{v}|$ is the cardinality of the set $\mathbf{v}$ that contains the points of a voxel.

$$f_v^i = [p_x^i, p_y^i, p_z^i, p_\iota^i, \tilde{p}_x^i, \tilde{p}_y^i, \tilde{p}_z^i, v_x^i, v_y^i, v_z^i], \quad i \in \mathbf{v} \tag{3.10}$$

$$\tilde{p}_*^i = p_*^i - \frac{\sum_{j \in \mathbf{v}} p_*^j}{|\mathbf{v}|}, \quad * \in x, y, z \tag{3.11}$$

The linear VFE is realized by projecting the point features $f_v^i$ into the goal voxel embedding dimension $C_v$ via a linear layer ($\mathbb{R}^{1 \times C_{in}} \rightarrow \mathbb{R}^{1 \times C_v}$) and taking the maximum value among all points in $\mathbf{v}$ for each dimension of $C_v$ ($\mathbb{R}^{m \times C_v} \rightarrow \mathbb{R}^{1 \times C_v}$, $m = |\mathbf{v}|$). The MLP VFE, instead, uses a MLP layer for the projection and the final voxel feature $\mathbf{f_v}$ is the mean of the projected point features that belong to the voxel.

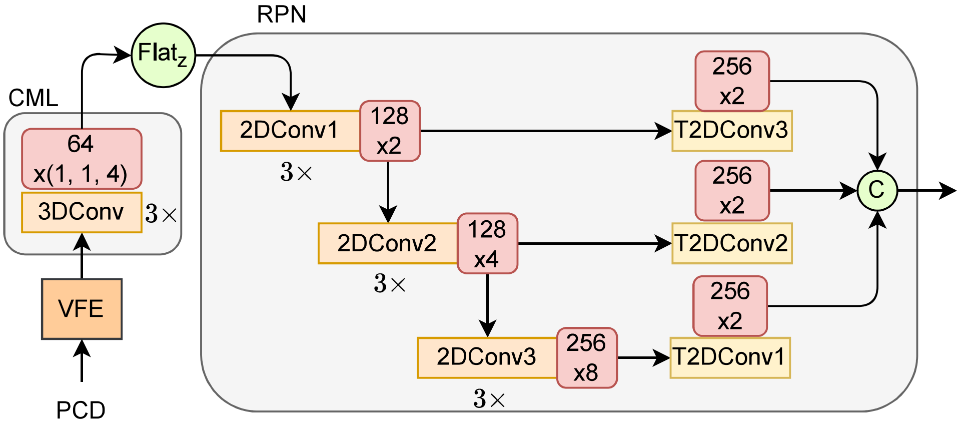


*Figure 3.8: The architecture of VoxelNet*

$$\mathbf{f_v} = max(linear(f_v^i)) \tag{3.12}$$

$$\mathbf{f_v} = mean(mlp(f_v^i)) \tag{3.13}$$

### VoxelNet

VoxelNet was first proposed by Zhou and Tuzel (2018) for 3D object detection on point cloud data. As illustrated in Figure 3.8, this network mainly consists of three modules: the VFE, the Convolutional Middle Layers (CML) and the Region Proposal Network (RPN). The VFE encodes the original point-cloud data into voxel embeddings by a linear layer. The VoxelNet implemented in this work uses the linear encoding following Equation (3.12). The encoded voxel features are the input for the CML module, which contains three 3D convolutional layers to further encode the point-cloud features in the 3D space. The first and the last layer have the strides of (1,1, 2), respectively, on the x, y, and z axes, and the second layer has the strides (1,1,1). Therefore, the output feature $F_{\text{cml}} \in \mathbb{R}^{X \times Y \times Z \times C}$ is compressed along the Z axis and has strides of ×(1,1, 4). To obtain 2D feature maps for the RPN, the feature $F_{\text{cml}}$ is then flattened along the Z-axis to shape $X \times Y \times ZC$. The RPN is composed of three down-sampling 2D convolutional layers with stride two. Therefore, feature maps of three different resolutions are obtained. They are two, four, and eight times down-sampled, respectively. Over each output of these three layers, a transposed 2D convolutional layer is employed to up-sample the features back to stride two (two times down-sampled) and then concatenated to merge the features learned from different resolutions and receptive fields.

### PointPillar

Instead of voxels, *PointPillar* (Lang et al., 2019) uses the pillar feature network (PFN) to encode input point clouds into a grid of pillars, as shown in Figure 3.9. PFN can be regarded as a special case of VFE, similar to VoxelNet. However, the voxel size on the z-axis is set to $vs_z = Z_{max} - Z_{min}$. This results in the encoded feature of shape $X \times Y \times 1 \times C$. Omitting the third dimension, the pillar feature map $F_{\text{pillar}} \in \mathbb{R}^{X \times Y \times C}$ is obtained. Over this feature map, the PointPillar network can directly perform 2D convolutions to extract the point cloud features which is much more efficient than the 3D convolution. This feature extraction model has the same structure as the RPN of VoxelNet; however, with different numbers of convolution layers in each convolution block and different numbers of encoding channels. The details are shown in Figure 3.10. The RPN employed

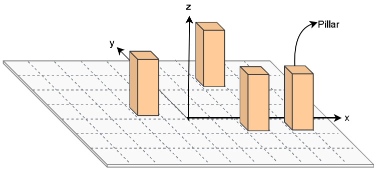


*Figure 3.9: The encoded pillar vectors of PointPillar network*

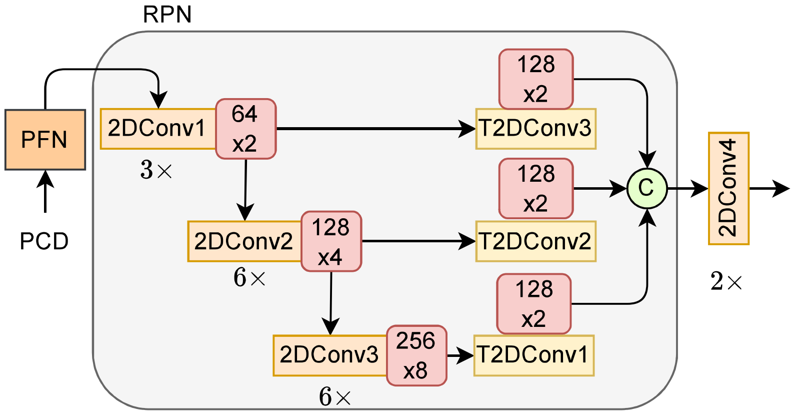


*Figure 3.10: The overall architecture of PointPillar network*

in PointPillar uses 4, 6, 6 convolution layers and 64, 128, 256 channels for the three 2D convolution blocks, respectively. In addition, the output dimension of the transposed 2D convolution blocks is also reduced to 128 channels, leading to a concatenated output of 384 channels, which is then projected to the output feature dimension of 256 channels by the final convolution block *2DConv4*.

### SpConv

*SpConv* network is a fully sparse 3D convolutional network that is composed of a stack of sparse 3D convolutional layers, as shown Figure 3.11. In this network, two types of 3D sparse convolution are used, namely *SpConv3D* and *SubMConv3D*[1]. *SpConv3D* is the sparse version of the *Conv3d* in PyTorch[2]. Therefore, it dilates the input coordinate and generates the output in the format of $O_{dilate}$ as shown in Figure 3.5. Aligning to the dense *Conv3d*, a spatial shape is given for each *SpConv3D* to constrain the output coordinates to the given shape. Differently, *SubMConv3D* takes $O_{in}$ as its output coordinates which is the same as the input. This is used to keep the sparsity of the intermediate sparse tensors so as to keep the efficiency of sparse convolutions.

This network first voxelizes the input point clouds and encodes each voxel with *mean VFE* as introduced in Section 3.2.1. Then the voxel features are further processed by a series of sparse convolutions, each convolution layer is notated by *Layer Name (output dimension, convolution stride)*. For example, the first convolutional layer is a *SubMConv3D* layer which projects the input voxel features into the output dimension of 16 without down-sampling (stride = 1). Then three convolution blocks with down-sampling of the voxels in each block are attached. As shown on the right side of Figure 3.11, each block contains three convolution layers, namely one *SpConv3D* followed by two *SubMConv3D* layers. Only the first layer uses a convolution stride of two to down-sample the voxels two times. In other words, the number of the voxels is down-sampled, but the size of the voxels is doubled. After three convolution blocks, the voxel features are down-sampled eight times. The output dimension of each block is 32, 64, and 64, respectively. The last layer of this network is a *SpConv3D* layer with the output dimension of *out_dim* (normally set to 128), the stride two along the z-axis, and the stride one along the x- and y-axis.

### MinkUnet

Unet (Ronneberger et al., 2015) was initially proposed for processing medical images. It merges features of different resolutions at each convolution stage to enhance the feature extraction and

---

[1] Implementation in GitHub. `https://github.com/traveller59/spconv`

[2] PyTorch implementation of dense 3D convolution.. `https://pytorch.org/docs/stable/generated/torch.nn.Conv3d.html`

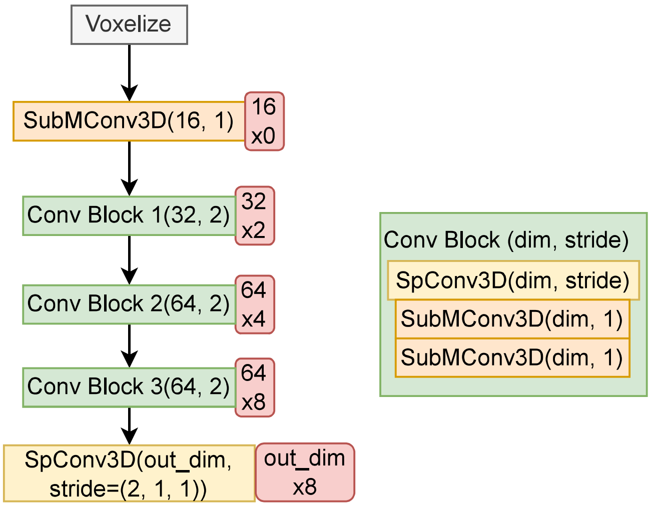


*Figure 3.11: The architecture of SpConv 3D backbone network*

representation ability of the network. Because of this advantage, it is widely applied or extended to other research fields, including point-cloud data processing. As shown in Figure 3.12, the *MinkUnet* (Choy et al., 2019) with the Unet-like structure is selected as a backbone network to extract the features of the point cloud in this work. As the name indicates, MinkUnet uses the fully sparse 3D convolutions implemented in the MinkowskiEngine library[1], which is very memory efficient.

The input point cloud is first voxelized and encoded with the MLP-based VFE into the shape of $N_v \times C_v$, where $C_v = 32$. The encoded voxels are then fed to the four convolutional blocks, *Conv1* contains only one layer to digest the input, *Conv2* to *Conv4* consist of three convolutional layers, in which the first layer down-samples the sparse tensors to lower resolution. In each down-sampling step, the resolution is halved, and the new generated voxel size is doubled. Therefore, the convolution stride at the down-sampling step $n_{ds}$ is $s = 2^n_{ds}$ and $P*$ represents the output features at different strides $s$. For example, $P1$ means the decode features in stride $s = 1$. Similarly, $\hat{P}_*$ represents the concatenated features of $P*$ with the corresponding shortcut features. In the up-sampling layers, the transposed convolutional layers have a structure similar to the counterparts in the down-sampling layers. The features from the shortcuts of the convolutional layers are all concatenated with the features from the transposed convolutional layers. All sparse convolutional layers in the MinkUnet are batch normalized and activated with Leaky ReLU. In the end of the network, the voxel features $\mathbf{f_v}$ and the encoded point features $\tilde{f}_v^i$ are concatenated to devoxelize the stride-one voxels $P_1$ and obtain features $P_0$ for each point.

### 3.2.2 Feature fusion

This thesis focuses on fusing the learned deep features as it can flexibly adjust the feature content to balance between the collective perception performance and the communication bandwidth requirement. However, the deep features for CPM can be in any format. According to the methods used in this thesis, the format can be categorized into *Map Feature Sharing* (MFS) and *Keypoint Feature Sharing* (KFS). With the MFS strategy, the IAs share the learned full deep feature maps with each other. This map can be the output of any intermediate layer of the network. To reduce the CPM size, KFS selects the most important keypoint features in the Region of Interest (RoI). These keypoints can be either 3D points or 2D points in the feature map.

---

[1] Minkowski Engine. `https://github.com/NVIDIA/MinkowskiEngine`

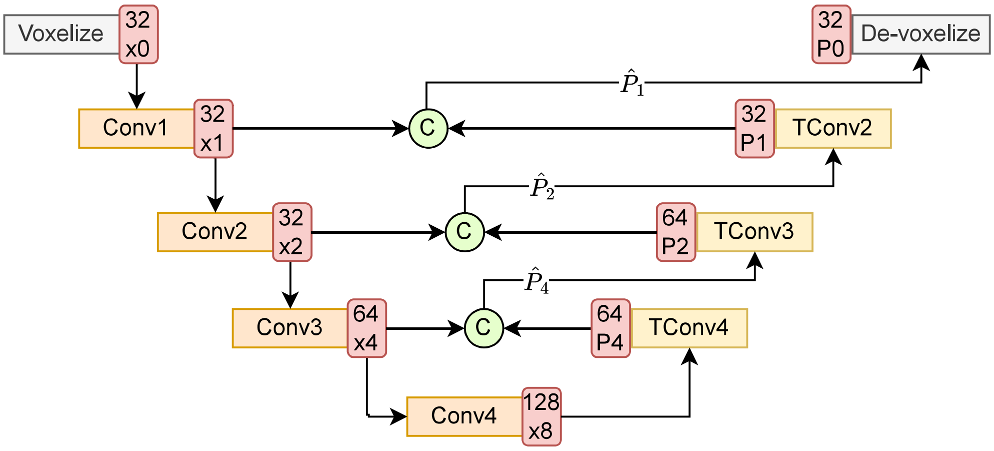


*Figure 3.12: The architecture of MinkUnet*

In addition to the CPM format, the feature fusion method can also be classified into two categories: *non-learnable* fusion and *learnable* fusion. The non-learnable fusion method indicates that no parameter in this fusion process is required to be learned during the training process. Typically, these methods employ operations like the *maximum*, *addition* and *concatenation*. In contrast, a learnable fusion method is a fusion module with learnable parameters. In the following, the non-learnable *Maxout fusion* and the learnable *Attention fusion* are introduced as they are used in the baseline or comparative models in this thesis.

### Maxout fusion

Maxout fusion was used in *Fcooper* (Chen et al., 2019a). As the name indicates, it takes the maximum value from the feature maps of different IAs. Assume the feature maps are $\mathbf{F} \in \mathbb{R}^{N\times W\times H\times C}$, where $N$ is the number of IAs, $W$,$H$,$C$ is, respectively, the width, height, and the number of channels of the feature map of each IA, the Maxout operation is performed over the first dimension $N$. The output feature $F \in \mathbb{R}^{W\times H\times C}$ is then regarded as the fused feature of the ego-IA.

### Naive Fusion

Naive Fusion involves mixing all the BEV feature points from all IAs. It takes the averaged feature at each BEV position as the final fused feature of multiple IAs. Similarly to Maxout fusion, this method does not contain any learnable parameters. It is simple; however, it is not learnable.

### Attention fusion

The attention mechanism is useful for selecting the important information; therefore, it is also employed by Xu et al. (2022c) to merge the feature maps of different IAs. Similarly to Maxout fusion, attention fusion also operates over the first dimension that corresponds to the number of IAs; however, it uses the attention mechanism. It uses a self-attention module to attend to the features across the IAs, as shown in Figure 3.13. Mathematically, it takes the same feature tensor $\mathbf{F} \in \mathbb{R}^{(W\cdot H)\times N\times C}$ as the queries, keys, and values for the attention module, where $W \cdot H$ can be regarded as the batch size, $N$ the sequence length, and $C$ the embedding dimension. The attended feature is reshaped into $\mathbf{F}_{\text{attn}} \in \mathbb{R}^{N\times W\times H\times C}$ and only the feature map $F^{ego}_{\text{attn}} \in \mathbb{R}^{W\times H\times C}$ of the ego-IA is selected as the final fused features.

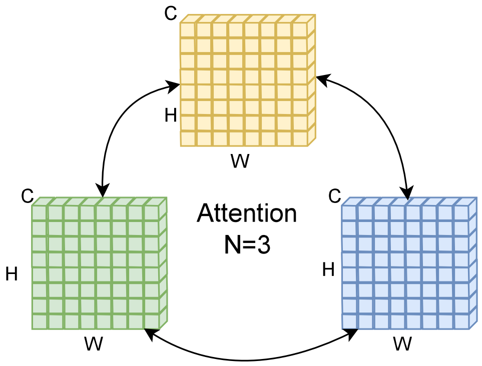


*Figure 3.13: Attention fusion. Green, blue and yellow feature maps are from $N = 3$ different IAs.*

### 3.2.3 Task heads

The task heads are the modules injected into the neural network that aim to achieve some learning tasks, such as object detection and semantic segmentation heads. These task heads may contain one or several sub-heads, which can be categorized into classification and regression tasks. Classification tasks aim to generate scores for classifying points or objects to the predefined semantic labels. Scores can also be used to rank points or regions of interest (RoIs) to select the most important information for further processing. Regression tasks learn to predict numerical values, such as the pose and size of an object in the driving scenario.

For autonomous driving, the objects in the driving scenarios should be identified and located so that the driving system can safely plan driving decisions to avoid collisions with these objects. This involves the object detection task. In addition, the vehicle should be aware of its own position relative to the driving environment. By generating semantic maps in real time, the IAs can not only be aware of their location relative to the current environment, but also be able to know their global location by registering the current semantic map into the stored global map. The task of generating such semantic maps is called BEV semantic segmentation as the maps are 2D in bird's eye view.

**Object detection**

Object detection involves the process of identifying and localizing objects within a given scene, using captured 2D images or 3D point clouds. The focus of this work is the 3D object detection process on point cloud data, as in the example shown in Figure 3.14. To identify predefined target objects, such as vehicles, the object detection head should look over all possible locations of the observed areas and generate probability scores for each location to draw a conclusion if an object exists at each of these locations. This process is achieved by a classification sub-head. However, the driving scene is a continuous 2D space in bird's eye view, resulting in infinite locations that require classification scores. To this end, the classification heads operate only on the discretized feature maps and generate one score for each point on the feature maps. Based on these feature map points, a regression sub-head is required to predict the accurate *bounding box* (bounding box) represented by Equation (3.14), where $x$,$y$,$z$ are the coordinates of the center point along each axis in the 3D space, and $l$,$w$,$h$,$r$ are the length, width, height, and orientation of the bounding box, respectively.

$$\mathcal{B} = [x,y,z,l,w,h,r] \tag{3.14}$$

For numerical stability during the training process, the regression targets of bounding boxes are always encoded. There are two commonly used encoding methods, *anchor-based encoding* (AEnc)

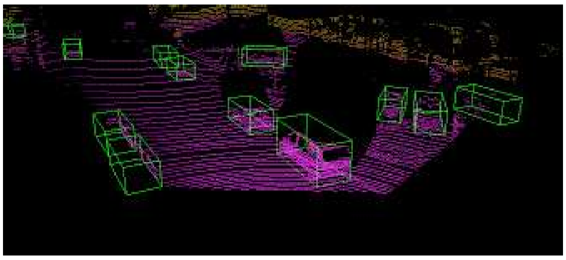


*Figure 3.14: 3D object detection*

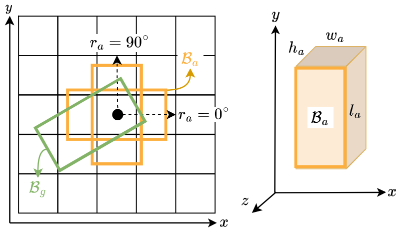


*Figure 3.15: Anchors for regression target encoding*

and *center-based encoding* (CEnc). Assume that the ground truth bounding box is $\mathcal{B}_g$ (Equation (3.15), green box in Figure 3.15), then AEnc encodes the targets $\mathcal{B}_t$ (Equation (3.16)) based on the anchor box $\mathcal{B}_a$ (Equation (3.17), orange box in Figure 3.15) and $\mathcal{B}_g$ using Equation (3.19) to Equation (3.22). As in the example shown in Figure 3.15, two anchors with orientation angles of 0 and 90 degrees are generated in each cell of the grid of the feature map for the AEnc. Note that this configuration is commonly used in many previous works (Zheng et al., 2021; Yuan et al., 2022; Xu et al., 2022c), although an arbitrary number of anchors for each cell could be used. Instead, CEnc (Equation (3.23) to Equation (3.25)) encodes the target based on $\mathcal{B}_g$ and the center $\mathcal{C}_a$ (Equation (3.18)) of the features map points. The center point is shown in Figure 3.15 as a black point in the center of the grid. The final encoded targets $\mathcal{B}_t$ are the values that the regression head needs to predict.

$$\mathcal{B}_g = [x_g, y_g, z_g, l_g, w_g, h_g, r_g] \tag{3.15}$$

$$\mathcal{B}_t = [x_t, y_t, z_t, l_t, w_t, h_t, r_t] \tag{3.16}$$

$$\mathcal{B}_a = [x_a, y_a, z_a, l_a, w_a, h_a, r_a] \tag{3.17}$$

$$\mathcal{C}_a = [x_a, y_a, z_a] \tag{3.18}$$

$$d_{xy} = \sqrt{l_a^2 + w_a^2} \tag{3.19}$$

$$x_t = \frac{x_g - x_a}{d_{xy}}, \quad y_t = \frac{y_g - y_a}{d_{xy}}, \quad z_t = \frac{z_g - z_a}{h_a} \tag{3.20}$$

$$l_t = log(\frac{l_g}{l_a}), \quad w_t = log(\frac{w_g}{w_a}), \quad h_t = log(\frac{h_g}{h_a}) \tag{3.21}$$

$$r_t = r_g - r_a \tag{3.22}$$

$$x_t = x_g - x_a, \quad y_t = y_g - y_a, \quad z_t = z_g \tag{3.23}$$

$$l_t = log(l_g), \quad w_t = log(w_g), \quad h_t = log(h_g) \tag{3.24}$$

$$r_t = [\sin r_g, \cos r_g] \tag{3.25}$$

To address this confusion in selecting the bounding box encoding methods, a comparative study is conducted in Section 8.2.6 with AEnc, CEnc, and a different number of anchors instead of the conventional configuration of two.

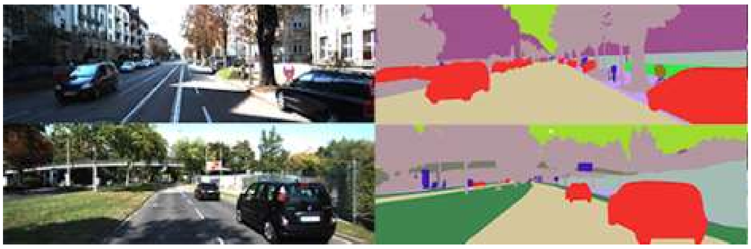


*Figure 3.16: Semantic segmentation of 2D images (Geiger et al., 2012)*

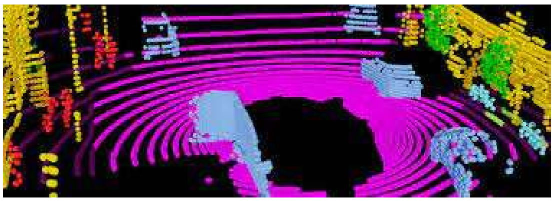


*Figure 3.17: Semantic segmentation of 3D points*

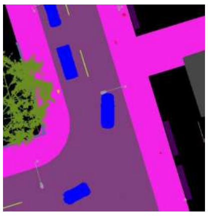


*Figure 3.18: Semantic segmentation of a BEV map*

### BEV semantic segmentation

Semantic segmentation is the task of assigning a class label to each point that can be the pixel on a 2D image (Figure 3.16), the point in a point cloud (Figure 3.17), or the point in a BEV map (Figure 3.18). The focus of this work is BEV map semantic segmentation (BEV SemSeg). On the one hand, the map generated by BEV SemSeg can be used for short-range navigation. On the other hand, the road geometry generated by BEV map SemSeg is beneficial for localizing the vehicles accurately.

## 3.2.4 Loss functions

To train the model with the above-mentioned task heads, loss functions are required to back-propagate the gradients and update the learnable parameters in the neural networks. For the classification head, the *cross entropy loss* $\mathcal{L}_{\text{ce}}$ is commonly used. However, training the model with cross-entropy loss on a dataset with an unbalanced amount of samples for different classes tends to make the network learn more features from the majority class; the minority classes tend to be overlooked. In this thesis, both object detection and BEV semantic segmentation for the class vehicles have more negative (background) samples than positive (foreground) ones. To overcome this bias in learning, *focal loss* $\mathcal{L}_{\text{fcl}}$ is introduced. In some safety-critical applications, such as autonomous driving, quantifying the confidence of the classification result is beneficial. For this purpose, *evidential loss* $\mathcal{L}_{\text{edl}}$ is used to train the BEV semantic segmentation model. For the regression head, the *smooth l1 loss* $\mathcal{L}_1$ is employed.

### Cross entropy loss

The concept of *Entropy* in information theory was introduced by Shannon and Elwood (1948) to calculate the average number of digits required to encode information or messages. It quantifies the average level of information, also interpreted as "surprise" or "uncertainty" of a random variable $Y$ which is distributed according to $p : \mathcal{Y} \rightarrow [0,1]$. For discrete events, the Shannon entropy is defined as

$$H(Y) = -\mathbb{E}[\log p(Y)] = -\sum_k p(y_k) \log p(y_k) = \sum_k p(y_k) \log \frac{1}{p(y_k)} \tag{3.26}$$

where $p(Y)$ is the distribution of all possible events $y_k$, and $p(y_k)$ is the probability of the occurrence of the $k$-th event. For the classification problem, $y_k$ can be regarded as the event that one instance is classified as the $k$-th class out of all possible semantic classes. Based on the entropy, the difference between two probabilistic distributions $p : \mathcal{Y} \rightarrow [0,1]$ and $q : \mathcal{Y}^* \rightarrow [0,1]$ can be measured with *cross entropy* as defined with

$$H(Y,Y^*) = -\mathbb{E}_{p(Y)}[\log q(Y^*)] = -\sum_k p(y_k) \log q(y_k^*) \tag{3.27}$$

In machine learning, the *cross entropy loss* is derived from the *Kullback-Leibler Divergence* (KL-Divergence) as described in Equation (3.28) to Equation (3.30), also known as the relative entropy of $p$ with respect to $q$. The distribution $p$ can be regarded as the true underlying distribution, from which the dataset $\mathcal{D}$ is drawn. The distribution $q(y_k|x_i,\boldsymbol{\theta})$ is then the approximated distribution of $p$ learned from the samples in dataset $\mathcal{D}$, where $x_i$ is the $i$-th sample in $\mathcal{D}$ and $\boldsymbol{\theta}$ is the parameters of the model $\mathcal{M}$.

$$\begin{aligned} D_{KL}(p\|q) &= H(p,q) - H(p) && (3.28)\\ &= -\sum_{ik} p(y_k|x_i) \log q(y_k|x_i,\boldsymbol{\theta}) + \sum_{ik} p(y_k|x_i) \log p(y_k|x_i) && (3.29)\\ &= \sum_{ik} p(y_k|x_i) \log \frac{p(y_k|x_i)}{q(y_k|x_i,\boldsymbol{\theta})} && (3.30) \end{aligned}$$

Training the machine learning model refers to the process of finding the distribution $q$ that infinitely approaches the true distribution $p$. This can be achieved by minimizing the KL-Divergence between $p$ and $q$. From Equation (3.28), one can observe that minimizing cross entropy $H(p,q)$ has the same effect as minimizing the KL-Divergence, because the entropy of $p$ is an unknown constant which does not depend on the model parameters $\boldsymbol{\theta}$. Therefore, the loss $\mathcal{L}_{\text{ce}}$ can be expressed with

$$\mathcal{L}_{\text{ce}} = -\sum_{ik} p(y_{ik}) \log q(y_{ik}) = -\sum_{ik} \delta(\hat{y}_{ik}) \cdot \log \hat{p}_{ik} \tag{3.31}$$

$$\delta(\hat{y}_{ik}) = \begin{cases} 1 & \text{if } \hat{y}_{ik} = y_{ik} \\ 0 & \text{otherwise} \end{cases} \tag{3.32}$$

where $i$ is the sample index, $\hat{y}_{ik} \in \{0,1\}$ is the predicted label, $y_{ik} \in \{0,1\}$ the ground truth label, and $\hat{p}_{ik} \in [0,1]$ the predicted probability for the $k$-th class of the $i$-th sample.

### Focal loss

It is very common that the training dataset contains unbalanced numbers of samples for different classes. In driving scenarios, the background samples are always in majority compared to the foreground objects. This unbalance tends to weaken the model on learning the features of foreground objects and generates a lot of false negative classification results. To tackle this problem, *focal loss* (Lin et al., 2020) introduces a modulating factor to reduce the weight of the loss value for well-classified majority examples and focus on misclassified hard examples. It regards each class as a binary classification between the fore- and background.

Starting from Equation (3.27), the *binary cross entropy loss* of each sample over each class can be simplified to

$$l_{\text{bce}}(\hat{p}_{ik},y_{ik}) = l_{\text{bce}}(p_t) = -\log(p_t) = \begin{cases} -\log(\hat{p}_{ik}) & \text{if } y_{ik} = 1 \\ -\log(1-\hat{p}_{ik}) & \text{otherwise} \end{cases} \tag{3.33}$$

$$p_t = \begin{cases} \hat{p}_{ik} & \text{if } y_{ik} = 1 \\ 1 - \hat{p}_{ik} & \text{otherwise} \end{cases} \tag{3.34}$$

The focal loss is then defined as

$$\mathcal{L}_{fcl} = \sum_{ik} \alpha_t (1 - p_t)^\gamma l_{\text{bce}}(p_t) = -\sum_{ik} \alpha_t (1 - p_t)^\gamma \log(p_t) \tag{3.35}$$

where $\gamma \in [0,\infty)$ is the *focusing* parameter, normally set to $\gamma = 2$ by default. The modulating factor is $(1-p_t)^\gamma$. As $p_t$ is approaching 1, this factor goes to 0 so that the well-classified samples are down-weighted. The parameter $\alpha_t$ is an additional factor used to balance the fore- and background class, normally set to $\alpha_t = 0.25$.

### Evidential loss

The conventional approach to converting the continuous model outputs, $\mathbf{z}$ (commonly referred to as logits, as they represent values prior to the exponential-based softmax function), into class probabilities involves applying the softmax function (Equation (3.36)). However, this method often exaggerates probabilities due to the exponential nature of the function, leading to unreliable uncertainty estimations. For instance, the softmax function can assign high confidence to incorrect classes for out-of-distribution samples, as the model has not encountered these during training. To address this limitation and enable the model to "know what it does not know," Sensoy et al. (2018) proposed leveraging the Dempster-Shafer Theory of Evidence (DST) to improve uncertainty estimation in classification tasks.

$$softmax(z_k) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.36}$$

DST generalizes the Bayesian theory with *subjective probability* (Dempster, 1968), also called *mass* or *degree of belief*. It assigns the masses to the elements in a power set, which is the set of all possible subsets of the states of a system. To quantify the belief masses and uncertainty with a well-defined theoretical framework, Jøsang (2016) formalizes the belief assignment of DST with Subjective Logic (SL) as a Dirichlet distribution because it models the second-order probabilities (the probability of probabilities) and uncertainty.

The Dirichlet distribution is parameterized with the concentrations $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_K]$, where $\alpha_k > 0$. Its probability density function is given by

$$Dir(\mathbf{p}|\boldsymbol{\alpha}) = \frac{1}{Beta(\boldsymbol{\alpha})} \prod_{k=1}^{K} p_k^{\alpha_k - 1} \tag{3.37}$$

Several examples of Dirichlet distributions over three classes are illustrated in Figure 3.19. When the probability mass is concentrated at one corner of the triangle, such as in the case of $\boldsymbol{\alpha} = (2,2,10)$, it signifies a high probability for the class located at that corner. Conversely, the example with $\boldsymbol{\alpha} = (1,3,3)$ indicates that the classes at both the top and bottom right corners of the triangle have the highest probabilities. When the concentration parameters for all classes are equal, the model shows no preference for any specific class. However, as demonstrated by the example $\boldsymbol{\alpha} = (0.9, 0.9,0.9)$, when concentration parameters are less than 1, an undesirable situation arises where all corners of the triangle exhibit high probabilities. To prevent this, $\boldsymbol{\alpha}$ is typically constrained with $\boldsymbol{\alpha} \geq 1$.

The total concentration, denoted as $S = \sum_{k=1}^{K} \alpha_k$, serves as a measure of the distribution's strength. A higher value of $S$ indicates the distribution is more concentrated as the example

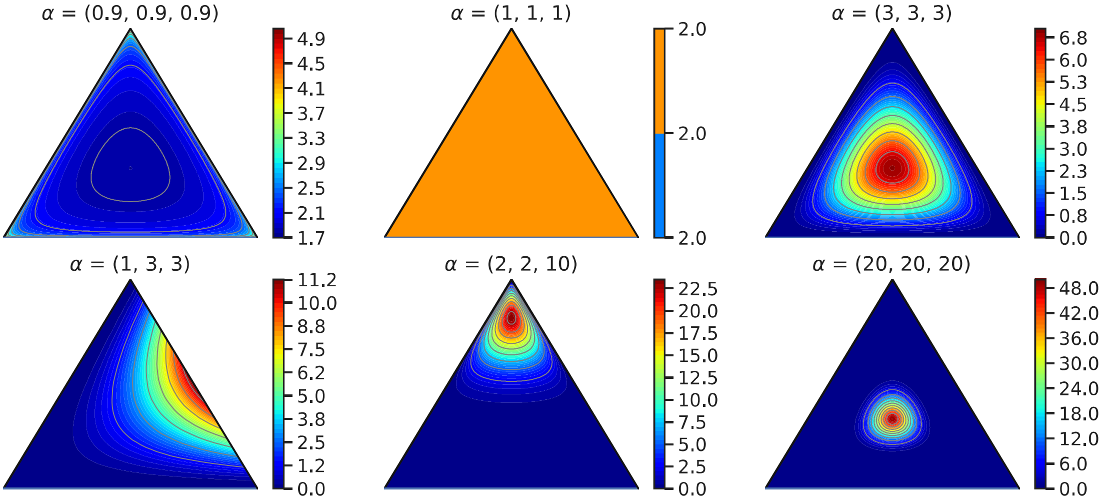


*Figure 3.19: Dirichlet distribution*

with $\boldsymbol{\alpha}$ = (20,20,20) and $K = 3$ shown in Figure 3.19. Intuitively, the belief masses $b_k$ can be related to the concentration parameters $\alpha_k$. By assigning the belief masses to an uniform Dirichlet distribution $Dir(\mathbf{p}|\alpha_i = 0, \cdots, \alpha_K = 0)$, the distribution becomes more concentrated. Sensoy et al. (2018) formulates this relationship with

$$b_k = \frac{e_k}{S} \tag{3.38}$$

$$\alpha_k = e_k + 1 \tag{3.39}$$

where $e_k \geq 0$ is termed as *evidence*. It ensures $\alpha_k \geq 1$ to avoid the invalid distribution status for the classification problem where the masses are pushed away to the extreme cases as shown in the case of $\boldsymbol{\alpha}$ = (0.9,0.9,0.9) in Figure 3.19. Based on this relationship, SL assumes that a frame of $K$ classes are mutually exclusive, each is assigned a belief mass $b_k$. In addition, an overall mass $u$ is defined to quantify the uncertainty. This results in $K + 1$ mass values that sum up to one as described in Equation (3.40), where $u, b_k \geq 0$ and $k = 1, \ldots, K$.

$$u + \sum_{k=1}^{K} b_k = 1 \tag{3.40}$$

From Equation (3.39) to Equation (3.40), one can derive that the uncertainty $u$ is the number of classes $K$ over the strength $S$ as shown in Equation (3.41).

$$\begin{aligned} u &= 1 - \sum_{k=1}^{K} b_k = 1 - \frac{\sum_{k=1}^{K} e_k}{S} \\ &= 1 - \frac{\sum_{k=1}^{K}(\alpha_k - 1)}{S} = \frac{S - \sum_{k=1}^{K} \alpha_k + K}{S} \\ &= \frac{K}{S} \end{aligned} \tag{3.41}$$

At the beginning of the learning process, the prior distribution of the classification problem is initialized as an uniform distribution $Dir(\mathbf{p}|\alpha_i = 0, \cdots, \alpha_K = 0)$ (Figure 3.19, $\boldsymbol{\alpha}$ = (1,1,1)), meaning no evidence is observed so that the distribution contains no information and is totally uncertain ($u = 1$) about "what class does an input sample belong to?". During the training process, the belief or opinion masses are gradually assigned to the distribution parameters $\boldsymbol{\alpha}$ to sculpt the Dirichlet distribution. As the training ends, the class assignment probabilities of the $i$-th sample $x_i$ are computed with Equation (3.42) to Equation (3.43), where $\boldsymbol{\alpha}_i$ is related to the output $f(x_i|\boldsymbol{\theta})$ of the neural network parameterized with $\boldsymbol{\theta}$. To generate evidence vectors that fulfill the constraint $e_k \geq 0$, Sensoy et al. (2018) proposed to replace the softmax activation at the last layer in a conventional deterministic neural network with the ReLU or the exponential activation.

$$\boldsymbol{\alpha}_i = e_i + 1 = f(x_i|\boldsymbol{\theta}) + 1 \tag{3.42}$$

$$\hat{\mathbf{p}}_i = \frac{\boldsymbol{\alpha}_i}{S} \tag{3.43}$$

$$\tag{3.44}$$

Assume $\mathbf{y}_i$ is the so-called one-hot ground truth label vector of length $K$, with only one element "1" that indicates the ground-truth class of the observation sample $x_i$, the other elements of $\mathbf{y}_i$ are all "0". Then, the *evidential loss* can be formulated as the sum of squares loss $\|\mathbf{y}_i - \mathbf{p}_i\|_2^2$ with respect to the class predictor (Equation (3.45)).

$$\mathcal{L}_{\text{edl},i} = \int \|\mathbf{y}_i - \mathbf{p}_i\|_2^2 \frac{1}{Beta(\boldsymbol{\alpha_i})} \prod_{k=1}^{K} p_{ik}^{\alpha_{ik}-1} d\mathbf{p}_i \tag{3.45}$$

$$= \sum_{k=1}^{K} \mathbb{E}[(y_{ik} - p_{ik})^2] \tag{3.46}$$

By factoring out the ground truth label $y_{ik}$, Equation (3.46) can be transformed into

$$\mathcal{L}_{\text{edl},i} = \sum_{k=1}^{K} (y_{ik} - \mathbb{E}[p_{ik}])^2 + Var(p_{ik}) \tag{3.47}$$

$$= \sum_{k=1}^{K} (y_{ik} - \frac{\alpha_{ik}}{S_i})^2 + \frac{\alpha_{ik}(S_i - \alpha_{ik})}{S_i^2(S_i + 1)} \tag{3.48}$$

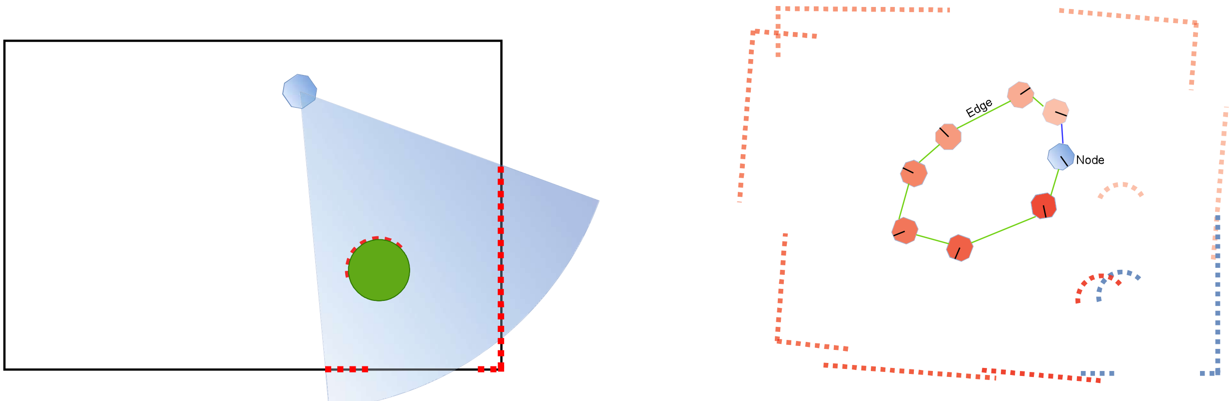
$$= \sum_{k=1}^{K} (y_{ik} - \hat{p}_{ik})^2 + \frac{\hat{p}_{ik}(1 - \hat{p}_{ik})}{(S_i + 1)} \tag{3.49}$$

To prevent the network from generating extremely large values during training, an additional term regularizes the concentration parameters to be closer to the uniform distribution. This term is formulated with the KL-divergence between the predicted distribution and the uniform distribution. The final evidential loss function can be represented by

$$\mathcal{L}_{\text{edl}} = \sum_{i=1}^{N} \mathcal{L}_{\text{edl},i} + \lambda_t \sum_{i=1}^{N} D_{KL}\Big(Dir(\mathbf{p}_i|\tilde{\boldsymbol{\alpha}}_i) \| Dir(\mathbf{p}_i|\langle 1, \ldots, 1 \rangle)\Big) \tag{3.50}$$

where $\lambda_t$ is the annealing coefficient that changes with the ratio between the epoch number $A_{\text{epoch}}$ and the maximum annealing step $A_{\max}$, reads as

$$\lambda_t = \min(1, A_{\text{epoch}}/A_{\max}) \tag{3.51}$$



*(a) Scanning scene with LiDAR (blue octagon) in the arena (black rectangle). The scanned points are in red.*

*(b) Scanning results captured at a sequence of scanning poses, stating with the blue node and ending with the pink node which is connected to the starting node with a blue edge.*

*Figure 3.20: An example scene for optimizing the LiDAR poses while scanning the arena.*

$\tilde{\boldsymbol{\alpha}}_i$ is the filtered version of $\boldsymbol{\alpha}_i$ to ensure that the KL-divergence only punishes the misleading predictions of $\boldsymbol{\alpha}_i$. It can be formulated as Equation (3.52), where $\odot$ is the element-wise multiplication.

$$\tilde{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_i \odot (1 - \mathbf{y}_i) + \mathbf{y}_i \tag{3.52}$$

### Smooth L1 loss

For the regression tasks, the commonly used loss is the $\mathcal{L}1$ loss, which is the least absolute deviation, and the $\mathcal{L}2$ loss, which is the least square errors. They are described with Equation (3.53) and Equation (3.54), where $\hat{y}$ is the real target value and $y$ is the predicted value.

$$\mathcal{L}1 = |\hat{y} - y| \tag{3.53}$$

$$\mathcal{L}2 = (\hat{y} - y)^2 \tag{3.54}$$

Generally, the $\mathcal{L}2$ loss function penalizes large errors more heavily than the $\mathcal{L}1$ loss and is preferred when the data is clean and the goal is to minimize the impact of small errors. In contrast, $\mathcal{L}1$ tends to be more robust to outliers in the data and prevents gradient exploding. Combining the advantages of both loss functions, *smooth $\mathcal{L}1$ loss* is introduced. As described in Equation (3.55), the smooth $\mathcal{L}1$ loss shifts between the $\mathcal{L}1$ and $\mathcal{L}2$ loss that are parameterized with $\beta$ depending on the error between the real and the predicted value.

$$\mathcal{L}1_{sm} = \begin{cases} 0.5 \cdot (\hat{y} - y)^2/\beta, & \text{if } |\hat{y} - y| < \beta \\ |\hat{y} - y| - 0.5 \cdot \beta, & \text{otherwise} \end{cases} \tag{3.55}$$

## 3.3 Pose Graph Optimization for Pose Alignment

Pose Graph Optimization (PGO) is an algorithm for improving the accuracy of estimated poses (positions and orientations) of a robot or a sensor as it moves through an environment. It is achieved by minimizing the errors in the poses by considering the relationships and constraints between them. PGO is commonly used in simultaneous localization and mapping (SLAM).

A simple scene, where a LiDAR sensor (blue octagon) is scanning an arena (black rectangle) with a landmark (green circle) in it, is shown in Figure 3.20a. A single measurement from the LiDAR is visualized with red points. As the LiDAR moves within the arena, it captures a sequence of measurements, as illustrated in Figure 3.20b. Each measurement is associated with a specific pose, which reflects the LiDAR's location and orientation at the time of measurement. Using these poses and their corresponding measurements, a graph can be constructed.

In this graph, each node represents a pose (*e.g.,* the location and orientation of the LiDAR during a measurement), while edges represent spatial constraints between the poses. For instance, subsequent measurements are expected to align spatially as closely as possible. The goal of PGO is to estimate all poses in the graph in a manner that best satisfies these constraints.

In the context of collective perception, where multiple intelligent agents (IAs) measure the same environment, the pose of each IA is represented as a node in the graph. However, point clouds captured by an IA's LiDAR often consist of tens of thousands of points. Directly using point clouds as measurement constraints for PGO presents two key challenges: (1) the high computational cost required for processing large-scale point clouds, and (2) the increased communication bandwidth needed to share these point clouds for collaborative PGO. Therefore, in this thesis, detected bounding boxes (bounding boxes) are shared instead, serving as compact and efficient measurements for PGO.

## 3.4 Evaluation Metrics

This thesis addresses the challenges of cooperative object detection and bird's-eye-view (BEV) semantic segmentation. To assess the performance of the proposed frameworks, Average Precision (AP) (Padilla et al., 2020) (detailed in Section 3.4.1) is employed to measure object detection accuracy, while Intersection over Union (IoU) (Section 3.4.2) is used to evaluate BEV semantic segmentation accuracy. Additionally, a calibration plot (Section 3.4.3) is applied to assess the quality of the estimated uncertainties in the BEV semantic segmentation results.

### 3.4.1 Average Precision (AP)

The commonly used metric to evaluate object detection is *Average Precision (AP)*, which is calculated by the integral of the area under the Precision-Recall (P-R) Curve as the green area shown in Figure 3.21. Mathematically, it can be expressed by Equation (3.56), where $r$ is the recall and $p(r)$ is the precision at the recall $r$. In this work, the AP following the implementation in Algorithms 1 and 2 is used as the unified metric for object detection.

$$AP = \int_{r=0}^{1} p(r)dr \tag{3.56}$$

In order to calculate precision and recall, Algorithm 1 takes the detected bounding boxes $\mathbf{B}_{\text{pred}}$ and their prediction scores (confidences) $\mathbf{S}_{\text{pred}}$, and the ground truth bounding boxes $\mathbf{B}_{\text{gt}}$ as input. The detected bounding boxes are sorted according to their scores in a descending order, namely $s_0 \leq s_1 \leq \cdots \leq s_{N_{\text{gt}}}$. The True Positive (TP) and False Positive (FP) flags of all detections are initialized to 0. For each detected bounding box $b_i$, its Intersection over Union (IoU) with each ground truth bounding box $b_j$ in the same scan frame is calculated with

$$iou_{ij} = \frac{I(b_i,b_j)}{U(b_i,b_j)} \tag{3.57}$$

where *Intersection* $I(b_i,b_j)$ is the overlapping area of $b_i$ and $b_j$ (Figure 3.22 left), and *Union* $U(b_i,$ $b_j)$ is the overall covered area of these two bounding boxes (Figure 3.22 right). Both *Intersection* and *Union* are calculated in BEV which is critical for the route planning in the 2D driving space. If the IoU of $b_i$ with any ground truth bounding box $b_j$ is larger than the threshold $thr_{iou}$, this ground truth bounding box is removed from the set $\mathbf{B}_{\text{gt}}$ to avoid repeated matching between the

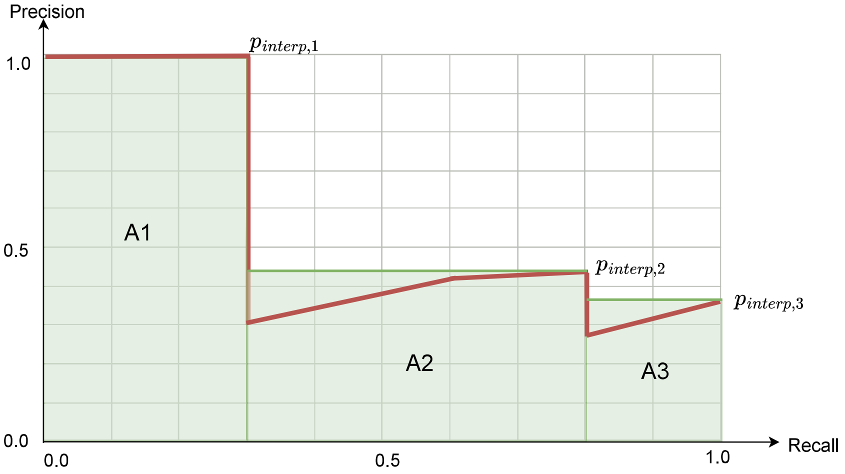


*Figure 3.21: AP: **A**rea **U**nder Recall-Precision **C**urve (AUC). The green areas (A1, A2, A3) are utilized to ensure that the AP metric is less sensitive to minor variations in the ranking of P-R pairs that define the P-R curve. The interpolated precision for these green regions is denoted as $p_{interp,*}$.*

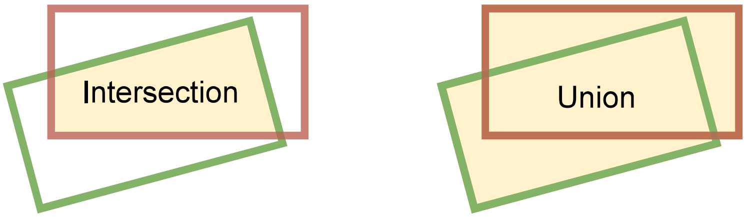


*Figure 3.22: Intersection and Union of two bounding boxes. The resulting intersection and union are illustrated with the yellow region.*

ground truth and the detections. Then the TP flag $tp_i$ of the detection $b_i$ is set to 1, otherwise the FP flag $fp_i$ is set to 1. The cumulative sum of elements of $TP$ is then calculated following line Line 14 in Algorithm 1. Then the recall of the $i$-th detection $r_i$ is the cumulative sum $\tilde{tp_i}$ over the total number of ground truth bounding boxes $N_{\text{gt}}$. The precision $p_i$ is then the ratio of $\tilde{tp_i}$ to $\tilde{tp_i} + \tilde{fp_i}$. The output of this algorithm is a Precision-Recall (PR) curve described by $Prec$ and $Rec$. Given the PR-curve, AP is computed via Algorithm 2, which iterates over the PR values from the end to the start of the PR curve and calculates the area under the PR-curve cumulatively with $p_{interp} \cdot (r_{i+1} - r_i)$, where $p_{interp}$ is the maximum precision aligned to the right as the green line shown in Figure 3.21.

### 3.4.2 Intersection over Union (IoU)

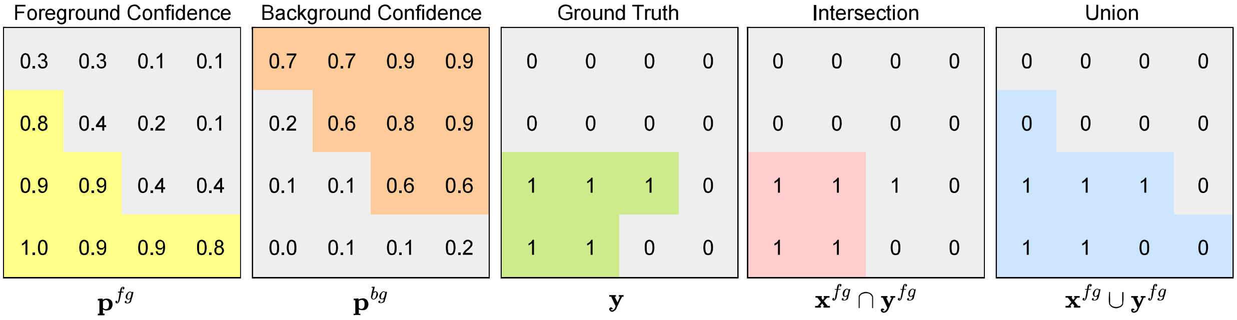
The Intersection over Union (IoU) is a commonly used metric for evaluating semantic segmentation performance. This thesis uses separated binary classification heads for each target semantic class because one pixel in the BEV map could belong to several semantic classes at the same time, *e.g.,* drivable road surface and vehicle. Specifically, one head is used for classifying the BEV semantic map into road surface (foreground) and non-road surface class (background), and one head for vehicle (foreground) and non-vehicle class (background). Therefore, the evaluation is also computed separately for each class with Algorithm 3. As illustrated in Figure 3.23, the predicted classification confidences $\mathbf{p} = (\mathbf{p}^{fg}, \mathbf{p}^{fg})$, respectively for the fore- (yellow pixels) and background (orange pixels), are generated for each BEV semantic map. The corresponding ground-truth labels are represented with the semantic map $y$, where the green pixels with label 1 are the foreground class and the gray pixels with label 0 are the background class. Based on the confidences $\mathbf{p}$, the final classification result, namely the pixels $\mathbf{x}^{fg}$ that are classified into foreground (yellow pixel in Figure 3.23), can be computed with Algorithm 3, line 1. Similarly, the ground-truth foreground pixels are represented with $\mathbf{y}^{fg}$ (Algorithm 3, line 2). The $IoU$ is then calculated with Algorithm 3, line 3, namely the number of intersection pixels (Figure 3.23, pink) divided by the number of union pixels (Figure 3.23, blue).

**Algorithm 1** Calculation of Precision and Recall

```
Ensure: $\mathbf{B}_{\text{pred}} = \{b_i | i \in 0,1,\dots,N_{\text{pred}}\}$, $\mathbf{S}_{\text{pred}} = \{s_i | i \in 0,1,\dots,N_{\text{pred}}\}$, $\mathbf{B}_{\text{gt}} = \{b_j | j \in 0,1,\dots,N_{\text{gt}}\}$, $TP = \{tp_i | tp_i = 0, i \in 0,1,\dots,N_{\text{pred}}\}$, $FP = \{fp_i | fp_i = 0, i \in 0,1,\dots,N_{\text{pred}}\}$, $thr_{iou}$ is the IoU threshold.
for i in range(0, $N_{\text{pred}}$) do
    $IoU = \varnothing$
    for each $b_j \subset \mathbf{B}_{\text{gt}}$ do
        $IoU = IoU \cup \{iou_{ij}(b_i, b_j)\}$
    end for
    if $max(IoU) > thr$ then
        $j = \text{argmax}(IoU)$
        $\mathbf{B}_{\text{gt}} = \mathbf{B}_{\text{gt}} - \{b_j\}$
        $tp_i = 1$
    else
        $fp_i = 1$
    end if
end for
$\tilde{TP} = \{\tilde{tp}_i | \sum_0^i tp_i, \quad i \in 0,1,\dots,N_{\text{pred}}\}$,    $\tilde{FP} = \{\tilde{fp}_i | \sum_0^i fp_i, \quad i \in 0,1,\dots,N_{\text{pred}}\}$
$Rec = \{r_i | \tilde{tp}_i / N_{\text{gt}}, \quad i \in 0,1,\dots,N_{\text{pred}}\}$
$Prec = \{p_i | \tilde{tp}_i / (\tilde{fp}_i + \tilde{tp}_i), \quad i \in 0,1,\dots,N_{\text{pred}}\}$
return $Rec$, $Prec$
```

**Algorithm 2** Calculation of AP

```
Ensure: $p_{interp} = 0$, $ap = 0$, $Prec = \{p_i | i \in 0,1,\dots,N_{\text{pred}}\}$, $Rec = \{r_i | i \in 0,1,\dots,N_{\text{pred}}\}$
for i in range($N_{\text{pred}} - 1$, 0) do
    if $p_{interp} < p_i$ then
        $p_{interp} = p_i$
    end if
    if $r_i < rec_{i+1}$ then
        $ap = ap + p_{interp} \cdot (r_{i+1} - r_i)$
    end if
end for
return $ap$
```



Foreground Confidence

| | | | |
|---|---|---|---|
| 0.3 | 0.3 | 0.1 | 0.1 |
| 0.8 | 0.4 | 0.2 | 0.1 |
| 0.9 | 0.9 | 0.4 | 0.4 |
| 1.0 | 0.9 | 0.9 | 0.8 |

$\mathbf{p}^{fg}$

Background Confidence

| | | | |
|---|---|---|---|
| 0.7 | 0.7 | 0.9 | 0.9 |
| 0.2 | 0.6 | 0.8 | 0.9 |
| 0.1 | 0.1 | 0.6 | 0.6 |
| 0.0 | 0.1 | 0.1 | 0.2 |

$\mathbf{p}^{bg}$

Ground Truth

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |

$\mathbf{y}$

Intersection

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |

$\mathbf{x}^{fg} \cap \mathbf{y}^{fg}$

Union

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |

$\mathbf{x}^{fg} \cup \mathbf{y}^{fg}$

*Figure 3.23: Intersection over Union (IoU) for BEV semantic segmentation.*

**Algorithm 3** Calculation of BEV Semseg IoU

```
Ensure: $\mathbf{p} = \{(p_i^{fg}, p_i^{bg}) | i \in 0,1,\dots,N\}$: predicted confidence map. $\mathbf{y} = \{(y_i | y_i \in \{0,1\}, i \in 0,1,\dots,N\}$: ground-truth labels. $N$: the number of pixels in the sample BEV map.
$\mathbf{x}^{fg} = \{x_i | p_i^{fg} > p_i^{bg}, i \in \{0,1,\dots,N\}\}$
$\mathbf{y}^{fg} = \{y_i | y_i = 1, i \in \{0,1,\dots,N\}\}$
$IoU = |\mathbf{x}^{fg} \cap \mathbf{y}^{fg}| / |\mathbf{x}^{fg} \cup \mathbf{y}^{fg}|$
return $IoU$
```

### 3.4.3 Calibration plot

The calibration plot is used to analyze the quality of the predictive uncertainty. It is the correlation plot of the classification accuracy versus the classification uncertainty. First, the uncertainty $u \in [0,1]$ is divided into ten bins, and each bin has an interval of 0.1. Subsequently, the average classification accuracy $ac$ of all the samples in each uncertainty interval is calculated. Examples of the calibration plot are shown in Figure 3.24. The blue bars are the average classification accuracy in each bin. A perfect calibration plot is shown by a diagonal line (calibrated line), indicating the highest negative correlation between classification accuracy and uncertainty, *i.e.,* high accuracy is associated with low uncertainty. The left sub-figure shows the desired calibration plot of a perfect model on uncertainty estimation. The middle sub-figure shows an example of an overconfident model, where the model performs less accurately than the desired accuracy at the given uncertainty intervals. In contrast, the right sub-figure shows the underconfident case where the model has higher accuracy than the calibrated line.

In addition to the calibration plot, *Calibration Error (CE)* is proposed to give a quantitative metric for evaluating the model's performance on estimating the uncertainty. It is the average of the absolute errors between the weighted classification accuracy of the model and the desired accuracy on the calibrated line. Mathematically, it can be calculated by Equation (3.58), where $u_i$ and $ac_i$ are the uncertainty and the average accuracy at the $i$-th bin, respectively.

$$CE = \frac{\sum_{i=1}^{10} |u_i - ac_i|}{10} \tag{3.58}$$

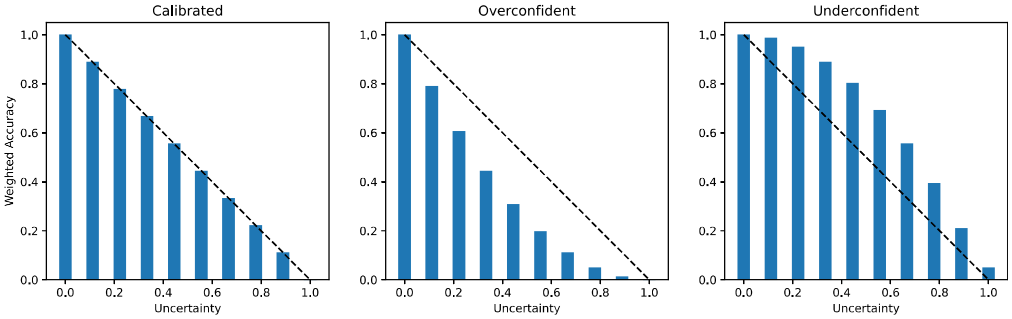


*Figure 3.24: Examples of calibration plot*

# 4 Related Work

Empowered by deep learning, the perception system of autonomous vehicles is able to extract complex features from the data, such as images and point clouds, that contains rich spatial and contextual information. These features are then used for specific perception tasks, such as object detection and semantic segmentation. These tasks aim to interpret the sensing data and represent the driving space in a way that is beneficial for decision-making of autonomous driving.

In the following section, previous work on object detection and semantic segmentation for autonomous driving is reviewed. Based on these studies, the properties and challenges of driving space representation are discussed. Then a review on ego-vehicle-based perception systems for object detection and semantic segmentation is introduced in Section 4.2. Recognizing the advantages of leveraging data from previous frames to enhance performance, the section also introduces temporal modeling techniques for autonomous driving. Finally, uncertainty estimation techniques are explored, as they are crucial for making safer driving decisions.

Most collective perception frameworks are implemented as extensions of ego-based perception frameworks. This extension involves the data fusion process from various IAs and addresses the challenges that arise during this process. The state-of-the-art collective perception models related to this aspect are discussed in Section 4.3.

## 4.1 Representation of Driving Spaces

Object detection (Jiao et al., 2019; Li et al., 2022a) is a typical way to interpret the dynamic driving space of an AV. Detected objects are typically characterized by 2D or 3D bounding boxes derived from camera or LiDAR data. However, this interpretation may be incomplete, as areas without detections or those obstructed by occlusions remain uninterpreted. These regions may include drivable areas, non-drivable areas, or zones occupied by undetected or unobserved objects. As a result, the AV may struggle to make reliable driving decisions based solely on object detection outcomes.

Semantic segmentation is another widely used method for interpreting the driving space. It classifies each measurement point—either pixels in images or points from LiDAR reflections—into specific semantic classes. To distinguish points within the same semantic class that belong to different object instances, this method is extended to panoptic segmentation (Kirillov et al., 2018), where each measurement point is assigned both a semantic label and an instance identity. Although semantic and panoptic segmentations provide a comprehensive and detailed view from the perspective of the ego vehicle in range view, they appear partial and sparse when converted to a BEV representation, which is typically used by the AV for planning driving maneuvers (Qiu et al., 2022).

Typically, the interpretation of the driving space is further extended to BEV semantic segmentation to mitigate the aforementioned limitations. The driving environment is represented as a BEV 2D image (Zhou and Krähenbühl, 2022; Xu et al., 2022a) and each pixel in the BEV map is marked with a semantic label, which gives a holistic overview of the driving surface for vehicle mapping and planning (Loukkal et al., 2021). In previous image-based works, BEV interpretations are also carried out as occupancy grid mapping (Lu et al., 2019), cross-view semantic segmentation (Pan et al., 2020a), or map-view semantic segmentation (Zhou and Krähenbühl, 2022). They transform image features from the image coordinates to an orthographic coordinate of the BEV map via either explicit geometric (Philion and Fidler, 2020) or implicit learned transformations (Zhou and Krähenbühl, 2022; Xu et al., 2022a; Li et al., 2022c). Compared to images, point cloud data with

3D information are more straightforward to generate such BEV maps by compressing information in the orthogonal direction in approaches such as PIXOR (Yang et al., 2018), PointPillars (Lang et al., 2019), and VoxelNet (Zhou and Tuzel, 2018).

This work also resorts to BEV maps for a holistic view of the 2D driving space, however with considering the observability. Because of the sparsity of distant measurements and occlusions, previous works (Philion and Fidler, 2020; Zhou and Krähenbühl, 2022; Xu et al., 2022a; Li et al., 2022c) that generate dense BEV maps are unreliable regarding the unobserved areas. For example, the occupied area of some occluded vehicles might be classified as a drivable area just because the categorical distribution learned from the historical data implies that the invisible points in the BEV map are more likely to be a drivable area than a vehicle. Therefore, in order to avoid unaccountable predictions on unobserved areas, this work proposes to only draw results from observed areas based on the geometric location of the measured points. In addition to the detection results of the dynamic objects, the AV is able to capture the real-time situation of the driving environment including the empty drivable area and other dynamic transportation participants, hence making safer driving decisions.

## 4.2 Ego-based Perception Systems

Ego-based perception systems process the data from the sensors mounted on the ego vehicle to achieve specific perception tasks. Object detection identifies and localizes the objects that are not registered in the stored navigation maps. Collision avoidance with these objects is essential for safe driving. The related works for object detection are discussed in Section 4.2.1. As another driving space interpretation, BEV semantic segmentation identifies the surrounding environment in a BEV map-view. The literature about this perception task is introduced in Section 4.2.2. To further improve the performance of the perception system, previous works also attempted to fuse the information from the previous frames and proved the benefit of temporal modeling of sequential data. These works are discussed in Section 4.2.3. At last, the works about uncertainty estimation of the perception results are discussed because knowing what the system does not know is also important for safe driving.

### 4.2.1 Object Detection

In general, object detection can be classified into one- and two-stage detectors. The one-stage detector uses one detection head to generate the final detection results from the learned deep features. For example, VoxelNet (Zhou and Tuzel, 2018) uses a 3D convolutional neural network to extract 3D deep features, compresses them along the height to obtain a 2D BEV feature map, and finally generates classification and regression results of the possible objects at each location in the feature map. PointPillar (Lang et al., 2019) and CIASSD (Zheng et al., 2021) use the same strategy; however, with different feature extraction networks. Differently, the two-stage detector PVRCNN (Shi et al., 2020) uses a Region-of-Interest (RoI) head to generate object proposals in the first stage. Based on these proposals, the model then summarizes more detailed information about the proposals to decide if these proposals are really an object and refines the regression parameters of these objects.

All the above-mentioned models use an anchor-based detection head to generate bounding boxes for the detected objects. This head assigns each location in the learned BEV deep feature map with one or multiple anchor bounding boxes of different orientations as the initial object assumptions and then performs classification and regression over these assumptions to generate the refined detection results. To avoid manually configuring the initial parameters for the anchor bounding boxes, CenterPoint (Yin et al., 2021) extends the 2D image-based object detection model CenterNet (Duan et al., 2019) to point cloud data and directly predicts the bounding boxes at each feature map location without taking anchor bounding boxes as references. Both anchor- and center-based detection heads may generate multiple bounding boxes for the same ground truth object. Therefore, they require an additional Non-Maximum Suppression (NMS) module to filter the detection result so that only one bounding box is kept for each detected object.

Omitting the non-parallelizable NMS module, Carion et al. (2020) built a 3D object detection model following DETR (Carion et al., 2020). This model initializes some random query points in the 3D driving space and projects these points into the image space to retrieve image features for the decoder of the transformer (Vaswani et al., 2017) to learn features for query points. Then a classification and regression head is used over these learned query features to obtain the final detection result. PETR (Liu et al., 2022a) uses a similar structure. Instead of projecting the query points to image-view, it encodes the image features with the embedded 3D positional information and lets the query points directly interact with the image features via a transformer (Vaswani et al., 2017) to learn the spatial and context features. Although the DETR-like models are anchor-free and NMS-free, they converge very slowly. One reason is that the learning target of the query points is assigned with the Hungarian algorithm, which could lead to unstable assignment and large distances between the query points and the target objects. To this end, this work uses the learning-efficient one-stage and center-based detection head.

### 4.2.2 BEV Semantic Segmentation

BEV semantic segmentation is a combination of the concept of the grid occupancy map (Elfes, 2013) and semantic segmentation (Shelhamer et al., 2014). It generates a dense BEV grid map with each grid cell assigned a semantic label. Different perception sensors, including cameras and LiDARs, can be used for generating such maps. VED (Lu et al., 2019) employs a Variational Auto-Encoder (Kingma and Welling, 2014) (VAE) to encode monocular images into latent space and then decode the latent features into a BEV map containing the semantic labels. Instead, Philion and Fidler (2020) "lift" the features of multi-view images to the 3D driving space by sampling discrete frustum points based on the predicted depth distributions for each pixel and then project these sampled points along with their learned features into the 3D space using the camera intrinsic and extrinsic parameters. The projected features are assigned to the closest cell in the BEV grid, and then fused by max pooling to obtain the BEV feature map for the following CNN operations so as to generate the final BEV semantic segmentation result. CVT (Zhou and Krähenbühl, 2022) simplified the projection process from image-view to BEV map-view using cross-view attention. It initializes a learnable BEV map-view grid with its BEV location embedded in this grid and lets these map-view points interact with the image features which are made camera-aware by adding the embedded features of the casting rays of each pixel. BEVFormer (Li et al., 2022c) uses a similar network structure; however, it extends it to sequential data by aggregating the BEV features from the previous frame with a self-attention module. Compared to camera-based BEV semantic segmentation, point cloud data is more straightforward to be projected into map-view and more efficient because of its sparsity. However, LiDAR-based BEV semantic segmentation is rarely researched. This work will concentrate on the LiDAR-based models and only take the camera-based models as comparative baselines.

### 4.2.3 Temporal modeling of sequential data

In early works, LSTM (Hochreiter and Schmidhuber, 1997) was widely used for processing sequential data. However, transformer (Vaswani et al., 2017) has rapidly taken its leading place because of the significant reduction in training and referencing (generating results for new unseen samples with the trained model) time by processing the sequential data in parallel. Based on transformer, DETR (Carion et al., 2020) first introduced the query-based object detection for 2D images. Since it is anchor-free and needs no post-processing, it has been widely used and was extended to 3D object detection (Carion et al., 2020; Liu et al., 2022a,b). More importantly, the object queries provide great convenience to efficiently interact with features of any modality. For instance, by integrating the 3D geometry information into the image memory features, PETR (Liu et al., 2022a) built a simple yet effective framework to let 3D queries directly interact with the 2D features in image space and obtained superior 3D object detection performance. By propagating the learned object queries to the next frames and modeling the temporal context for the current frame, QueryProp (He et al., 2022) improved the efficiency and accuracy for video object detection. With only images as input, StreamPETR (Wang et al., 2023) achieved on-par 3D object detection performance to the LiDAR-track benchmark on the NuScenes (Caesar et al., 2020) dataset. Furthermore, the concept of object query propagation has been applied to object tracking tasks. MOTR (Zeng et al., 2022) and TrackFormer (Meinhardt et al., 2022) use object queries to achieve temporal modeling and associate the objects between frames in an end-to-end manner, which is simple yet effective for tracking. Inspired by these works, this thesis models the temporal information of observations from different IAs with the selected potential object queries and shares the learned queries for the time-aligned cooperative object detection (TA-COOD, Chapter 8) to save communication bandwidth.

### 4.2.4 Uncertainty estimation

Uncertainties arise when dealing with inaccurate data and imperfect algorithms, making their estimation essential, particularly in safety-critical applications. The uncertainty associated with a DNN's output is referred to as predictive uncertainty (Gawlikowski et al., 2023). This uncertainty is typically characterized by modeling epistemic uncertainty, which captures systematic uncertainty within the model, and (or) aleatoric uncertainty, which accounts for random noise in observations (Kendall and Gal, 2017). Additionally, approaches such as the Prior Network (Malinin and Gales, 2018) quantify predictive uncertainty by modeling distributional uncertainty, which is caused by mismatches between the training data distribution and new inference data distribution.

To estimate the epistemic uncertainty, Bayesian Neural Networks (BNNs) (MacKay, 1992; Neal, 1995) provide a natural interpretation of the uncertainty by directly inferring distributions over the network parameters. However, applying BNNs to DNNs is challenging because calculating the posterior over millions of parameters is intractable. To address this, approximation methods have been developed, such as Monte-Carlo (MC) Dropout by Gal (2016) and Deep Ensemble by Lakshminarayanan et al. (2017). Gal (2016) demonstrates that training a dropout-based neural network can be analogous to optimizing the posterior distribution of the network's output. However, multiple forward passes with dropout enabled are required to infer uncertainty, making the process inefficient and time-consuming. Consequently, this method is not considered in this work. Similarly, Deep Ensemble involves training multiple models to approximate the distribution of network parameters and also requires several forward passes over each trained model. Due to these inefficiencies, it is also not adopted in this work.

To capture the aleatoric uncertainty, Direct Modeling is widely used, *e.g.,* by Feng et al. (2019); Meyer et al. (2019); Miller et al. (2019); Pan et al. (2020b); Feng et al. (2020). Compared to MC Dropout and Deep Ensemble, Direct Modeling assumes a probability distribution over the network outputs and directly predicts the parameters for the assumed distribution. Therefore, uncertainty is obtained over a single forward run and is more efficient. For classification problems, the conventional deterministic DNNs apply the Softmax function over the output logits to model the categorical distribution as a multinomial distribution. However, the Softmax outputs are often overconfident and poorly calibrated (Sensoy et al., 2018; Vasudevan et al., 2019).

Instead, converting the output logits into positive numbers via, *e.g.,* ReLU activation to parameterize a Dirichlet distribution quantifies class probabilities and uncertainties better. For example, the Prior Network (Malinin and Gales, 2018) captures the predictive uncertainty by explicitly modeling the distributional uncertainty and minimizing the expected Kullback-Leibler (KL) divergence between the predictions over certain (in-distribution) data and a sharp Dirichlet and between the predictions over uncertain (out-of-distribution) data and a flat Dirichlet. However, additional out-of-distribution samples are needed to train such a network to differentiate in- and out-of-distribution samples. In complex visual problems like object detection and semantic segmentation, obtaining enough samples to cover the infinite out-of-distribution space is prohibitive.

Differently, the Evidential Neural Network (Sensoy et al., 2018) treats the network output as beliefs following the Evidence and Dempster-Shafer theory (Dempster, 1968) and then derives the parameters for the Dirichlet distribution to model the epistemic uncertainty. Compared to BNNs, this method quantifies the uncertainty of a classification by the collection of evidence leading to the prediction result, meaning that the epistemic uncertainty of the classification can be easily quantified by the amount of evidence. Instead of minimizing the discrepancy of the predictive distributions with pre-defined ground truth distributions, Evidential Neural Network formulates the loss as the expected value of the basic loss, *e.g.,* cross-entropy, for the Dirichlet distribution. Therefore, no additional data or ground truth distributions are needed. Hence, this work applies this method for modeling the categorical distributions of the points in a 2D driving space and focuses on epistemic uncertainty (a lack of knowledge in the neural network-based model) to give a probabilistically explainable output of the perception system in IAs, and to use this estimated uncertainty in the collective perception step for distilling the most important information shared among the IAs.

## 4.3 Collective Perception

In recent years, cooperative perception with sharing information among IAs has proven to be beneficial for improving the perception accuracy and road safety (Chen et al., 2019a; Xu et al., 2022c; Yuan et al., 2022; Wang et al., 2020a; Yu et al., 2022; Xu et al., 2022a). By exchanging perceived objects, Günther et al. (2016) proved that sharing information among IAs can significantly increase the object detection accuracy compared to the ego-based perception. Instead of sharing detected objects, works by Chen et al. (2019b); Marvasti et al. (2020a); Xu et al. (2022c); Wang et al. (2020a) compared the performance of sharing and fusing information from different processing stages, including raw, semi-processed and fully-processed data fusion, all show that semi-processed data fusion has the best potential to achieve the best performance with limited and controllable communication resource consumption. However, semi-processed data can be in any format and processing stages. For instance, F-Cooper (Chen et al., 2019a) who experimented with both voxel feature fusion and spatial deep feature map fusion, found that voxel feature fusion has better performance because it keeps more information details. Instead, Xu et al. (2022c) encode the point clouds into point-pillars (Lang et al., 2019) with fully-connected layers, and then uses dense convolutions to generate feature maps. These feature maps from different IAs are then fused with an attention module instead of max-out used in F-Cooper. These learnable encoding and fusion modules ensure that the most important information is selected and shared to achieve better performance. To further increase the perception accuracy while spatial misalignment between the feature maps of IAs exists, SCOPE (Yang et al., 2023) introduced the pyramid LSTM (Lei et al., 2022) to reason about temporal and spatial information from sequential point clouds. While these works all share BEV feature maps for fusion, Hu et al. (2022) utilizes the learned criterion confidence to select the most relevant information and reduce the data size for sharing. Similarly, FPVRCNN (Yuan et al., 2022) adopts the sparsity of point clouds and learns the most important object keypoints for sharing to further reduce the bandwidth requirement. They all achieve good performances on the OPV2V (Xu et al., 2022c) dataset with significantly less data sharing.

### 4.3.1 Model efficiency

Except FPVRCNN, all above mentioned collective perception models encode point clouds into BEV feature maps and process them with dense convolutions or transformers which are very computationally demanding. Moreover, modeling the temporal context with sequential data like SCOPE (Yang et al., 2023) might be beneficial to obtain robust temporal features. However, the computational demand is also increasing dramatically as the data from several IAs in several frames should be processed. Most of the previous works (Xu et al., 2022c; Wang et al., 2020a; Yin et al., 2024; Xu et al., 2022b; He et al., 2023) use dense convolutions or transformers to process BEV feature maps, which consume a lot of GPU memory and it is hard to extend the model to sequential data on limited computational resources. In contrast, the potential of fully sparse network structures has rarely been researched. In this work, a fully sparse framework is proposed for collective perception.

### 4.3.2 Sensor asynchrony

Previous works for collective perception have sought to minimize bandwidth usage by compressing the learned deep BEV features (Wang et al., 2020a; Chen et al., 2019a; Xu et al., 2022c) or selecting deep keypoint features (Yuan et al., 2022), rectify localization errors through learning-based techniques (Wang et al., 2020a; Xu et al., 2022b) or analytic algorithms (Yuan et al., 2022; Yuan and Sester, 2022), and synchronize communication delays with the attention mechanism (Xu et al., 2023) or explicitly predicting the future features before data fusion (Yu et al., 2023). Nonetheless, none have considered the asynchronized sensor ticking time. More specifically, previous benchmarks (Xu et al., 2022c; Yu et al., 2022; Xu et al., 2023; Yu et al., 2023) assume that the sensors have synchronized global ticking time. In reality, these sensors might have asynchronous ticking time in each aligned frame, leading to an inhomogeneous observation time of each object in the scenario and large spatial displacement during data fusion, especially when the CAVs have high speeds. Therefore, this work explores the sensor asynchrony problem and designs a temporal model to mitigate its influence on collective perception.

### 4.3.3 Datasets for collective perception

It is highly expensive and complicated to obtain training data for cooperative object detection. It requires several vehicles and sensors observing the same scene. This scale leads to highly complex calibration and post-processing to generate accurate ground truth meta information (*e.g.,* sensor poses) and annotations. Therefore, the initial attempts for collective perception either use simplified driving scenarios (Chen et al., 2019b,a) or synthetic data (Xu et al., 2022c; Yuan and Sester, 2021; Li et al., 2022b). For instance, Chen et al. (2019a) conducted cooperative perception experiments on a dataset captured from a static parking lot with static sensors. This is the simplest set up without considering any dynamics of sensors or objects in the scenario. Via simulation, Xu et al. (2022c) generated a large-scale dataset for collective perception and built a cooperative object detection benchmark; however, without considering sensor asynchrony.

Recently, the real datasets DAIR-V2X (Yu et al., 2022) and V2V4Real (Xu et al., 2023) were made accessible. Both are only configured with two agents, DAIR-V2X with one CAV and one connected infrastructure agent, V2V4Real with two CAVs. Because of the dynamics and asynchrony of sensors, there might be spatial misalignment between dynamic objects observed by the two agents. To generate unified ground truth bounding boxes for cooperative object perception, both DAIR-V2X (Yu et al., 2022) and V2V4Real (Xu et al., 2023) take the ego-vehicle's annotation as the ground truth if the annotations from two agents spatially overlap. Only at the blind spot of the ego-vehicle, the annotations of the cooperative agent are taken. In this way, the final generated bounding box might contain errors and does not reflect the true location of the encapsulated object at the given timestamp. This might also negatively influence the temporal predictability of sequential models. To better preserve the detailed temporal information and its relation to the accurate location of the objects in the scenario, this work proposes to generate global time-aligned ground truth bounding boxes for the time-aligned object detection task.

# 5 Datasets

This chapter introduces the datasets used for the BEV semantic segmentation and object detection task. Both simulated and real datasets are utilized to evaluate the proposed frameworks. The BEV semantic segmentation framework (Chapter 7) is evaluated on the simulated dataset *OPV2V* and the real dataset *V2Vreal*. The Time-Aligned Cooperative Object Detection (TA-COOD) framework is assessed on the simulated dataset *OPV2V*, the real dataset *DairV2X*, and their newly generated variants, *OPV2Vt* and *DairV2Xt*. These variants are enhanced with more fine-grained, point-wise timestamps for each point in the point clouds, improving temporal precision.

## 5.1 Simulation Datasets

### 5.1.1 OPV2V

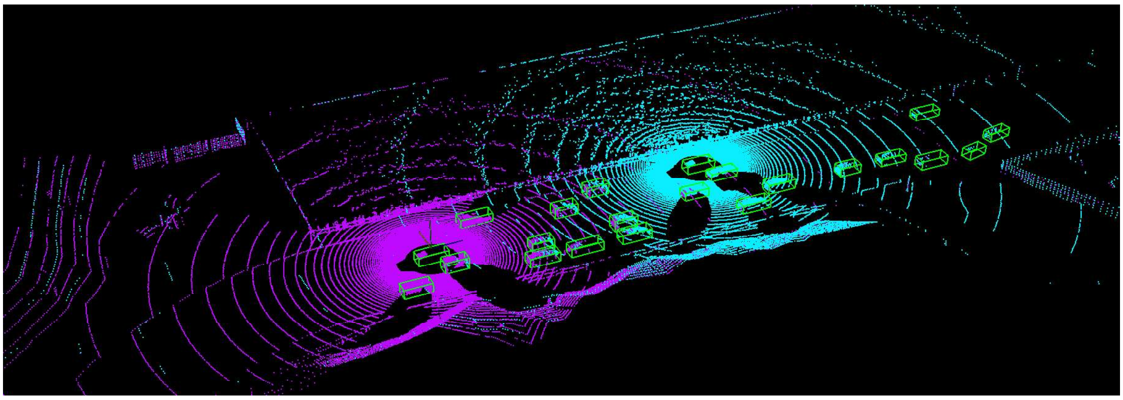
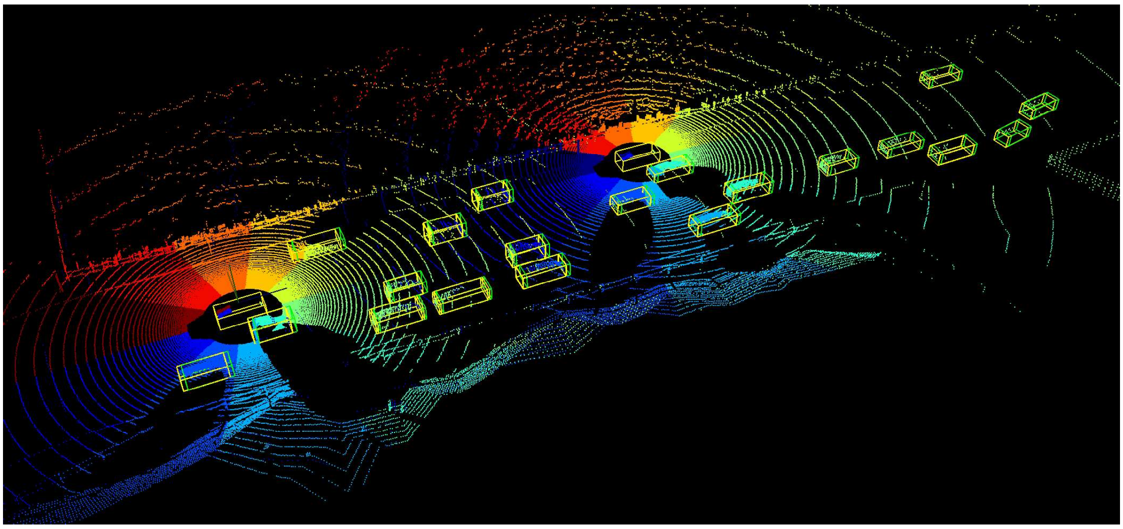


*Figure 5.1: OPV2V dataset: cyan and purple points are the LiDAR data of the ego and cooperative vehicle, repectivly. Green boxes are the ground-truth annotations of the vehicular participants in the scenario.*

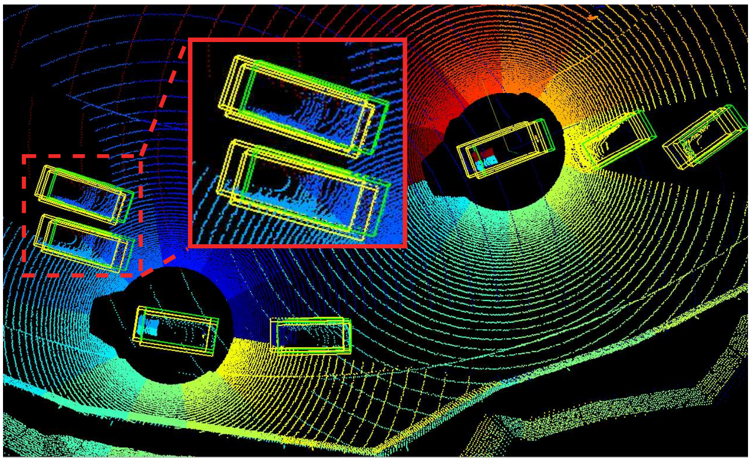
*OPV2V* dataset (Xu et al., 2022c) is a synthetic dataset generated by the simulator CARLA (Dosovitskiy et al., 2017). It serves as a prominent benchmark specifically curated for collective perception tasks. It comprises 44 scenes and 6765 training frames, along with 16 scenes and 2170 frames designated for testing purposes. These scenes encompass diverse driving scenarios, spanning urban environments, rural areas, and highways across nine simulated cities. It contains image and point cloud data as well as the ground truth bounding boxes for object detection. Besides, it also provides the semantic labels for the BEV semantic segmentation task. In Figure 5.1, the point cloud data from a sample frame with two vehicles, simulated using a 64-beam LiDAR, is illustrated. The purple points are the LiDAR observations of the ego vehicle, and the cyan points are the scans of the cooperative vehicle. Green boxes are the ground-truth bounding boxes of the vehicular objects that should be detected. Adhering to the official evaluation configuration of *OPV2V*, the parameters are configured as follows: the communication range $R$ is set to 70 meters, the maximum number of cooperative vehicles $\mathbf{C}_{\text{nbr}}$ is set to 7, and the detection range is $[-140{,}140]m$, $[-40{,}40]m$, and $[-3.0{,}1.0]m$ along the x-, y-, and z-axes, respectively.

### 5.1.2 OPV2Vt

The *OPV2V* dataset is simulated frame by frame without accounting for the asynchronous timing of sensor ticks within each frame, leading to unrealistic data points that are assumed to be observed



*(a) OPV2Vt*



*(b) OPV2Vt zoom-in view*

*Figure 5.2: OPV2Vt dataset: The points scanned in one frame are observed at different timestamps (blue to red scaled colors). The objects observed by two sensors are captured at different times, resulting in unaligned ground truth bounding boxes (yellow boxes). The bounding boxes at the last time point of this frame are taken as the final ground truth for the time-aligned cooperative object detection (green boxes).*

at the same time. In contrast, real mechanical LiDAR sensors scan the environment by rotating mirrors to alter the laser beams' direction. This process captures measurement points sequentially in continuous time, leading to a rolling shutter effect when the sensor is mounted on a moving vehicle. For a given target object, this effect can cause significant time offsets (0 to 0.1s) between measurements taken by the ego vehicle and those taken by a cooperative vehicle. Such time offsets may result in considerable displacement (more than $1m$) of the measured target object, posing challenges in the data fusion process necessary for collective perception.

In the context of this thesis, a more realistic dataset with considering asynchronous sensors is generated by simulation with CARLA. Unlike the *OPV2V* dataset, this newly generated dataset incorporates more precise temporal information, and is thus referred to as *OPV2Vt*. To be coherent with the existing simulation dataset *OPV2V*, the new simulation is conducted by replaying the scenarios of *OPV2V*. First, each frame of *OPV2V* is interpolated to ten sub-frames by interpolating the poses of objects and sensors in each frame. This partitioning into ten sub-frames is suitable as it maintains a manageable simulation budget while ensuring diverse observation times for objects.

Mathematically, the interpolation is achieved with Equation (5.1) and Equation (5.2). For example, the step from the 0-th frame at $t_0$ to the 1-th frame at $t_1$ is interpolated into ten sub-frames with

$$s_i = i \cdot (s_{t_1} - s_{t_0})/10 \tag{5.1}$$

$$r_i = \begin{cases} mod(i \cdot (r_{t_1} - r_{t_0} - 2\pi)/10, \quad 2\pi), & \text{if } r_{t_1} - r_{t_0} > \pi \\ mod(i \cdot (r_{t_1} - r_{t_0} + 2\pi)/10, \quad 2\pi), & \text{if } r_{t_1} - r_{t_0} < \pi \\ mod(i \cdot (r_{t_1} - r_{t_0})/10, \quad 2\pi), & \text{otherwise} \end{cases} \tag{5.2}$$

where i $\in \{1,2,\ldots,10\}$, $s_{t_0}$ and $s_{t_1}$ can be any variable of the location $[x,y,z]$ at time $t_0$ and $t_1$, respectively. Similarly, $r_{t_0}$ and $r_{t_1}$ represent one of the orientation variables $[r_{roll},r_{pitch},r_{yaw}]$. The interpolated location and orientation elements at the $i$-th sub-frame are represented with $s_i$ and $r_i$. The operation $mod(a,b)$ is $a$ modulo $b$. The different condition cases in Equation (5.2) for orientation are to prevent the orientation from flipping to the wrong directions.

Based on the interpolated poses of objects and sensors, all newly-generated sub-frames are replayed in the CARLA simulator to generate the corresponding data and annotations. As a result, each sub-frame only generates one-tenth of a full scan, which is as each patch of single-colored points shown in Figure 5.2a. Each sub-frame patch data is measured at different times. The dark blue shows the earliest measurement and the dark red the latest. To generate full scans that have different sensor ticking times, a random sensor ticking time in the range of $0s \leq t_i \leq 0.05s$ is generated for each sensor. Accordingly, several sub-frames ($n = t_i/0.01$) of data at the beginning of each sensor measurement sequence are discarded. Then each ten consecutive sub-frames of data are concatenated to obtain the full scan data for each frame. The sub-frame timestamps are assigned to each corresponding point in the full scan point clouds. The bounding boxes at the last sub-frame, namely the scan end, are taken as the global time-aligned ground-truth bounding boxes for the time-aligned object detection. The detailed data generation process is described with Algorithm 4. For evaluation of object detection, this dataset uses the same detection range as *OPV2V* for evaluation.

## 5.2 Real Datasets

### 5.2.1 V2Vreal

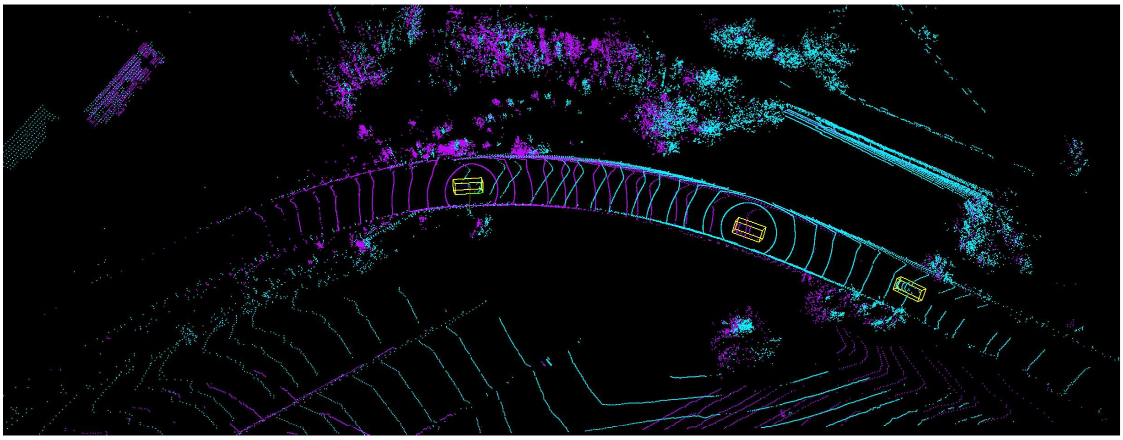


*Figure 5.3: V2Vreal dataset: LiDAR data with 3D bounding box annotations*

The *V2V4Real* (Xu et al., 2023) dataset is collected in Columbus, Ohio, USA, by two CAVs, each equipped with a Velodyne VLP-32 LiDAR sensor, two mono cameras (front and rear), and

GPS/IMU integration systems. The dataset covers a driving area of 410 km, including 347 km of highway road and 63 km of city road. From these driving miles, 67 representative scenarios are selected, each 10-20 seconds long and sampled at the frequency of 10Hz. In total, it contains about 10K annotated LiDAR frames. An example frame of LiDAR data with one ego vehicle (measurement points in purple) and one cooperative vehicle (measurement points in cyan) is shown in Figure 5.3. The ground-truth bounding boxes are shown with yellow boxes. Compared to *OPV2V*, the point clouds captured in this dataset are sparser; distant objects can hardly be observed. Therefore, the detection range of this dataset for the driving direction, $x$-axis, is reduced to $[-102.4,102.4]\,m$. For the $y$-coordinate, it keeps the range $[-38.4,38.4]\,m$. Along the z-direction, the range $[-5,3]\,m$ is taken because this dataset contains rugged roads on hills with uneven surfaces.

### 5.2.2 DairV2X

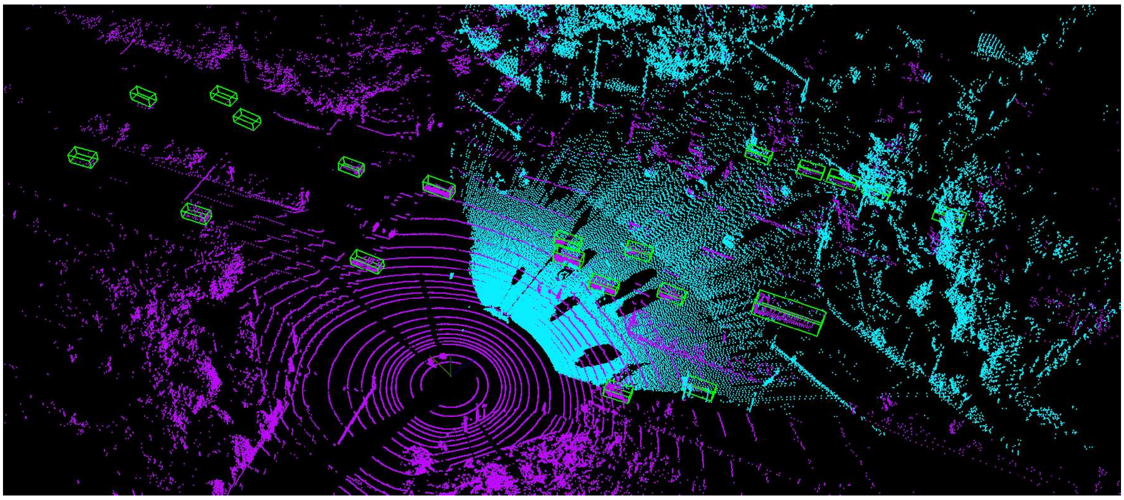


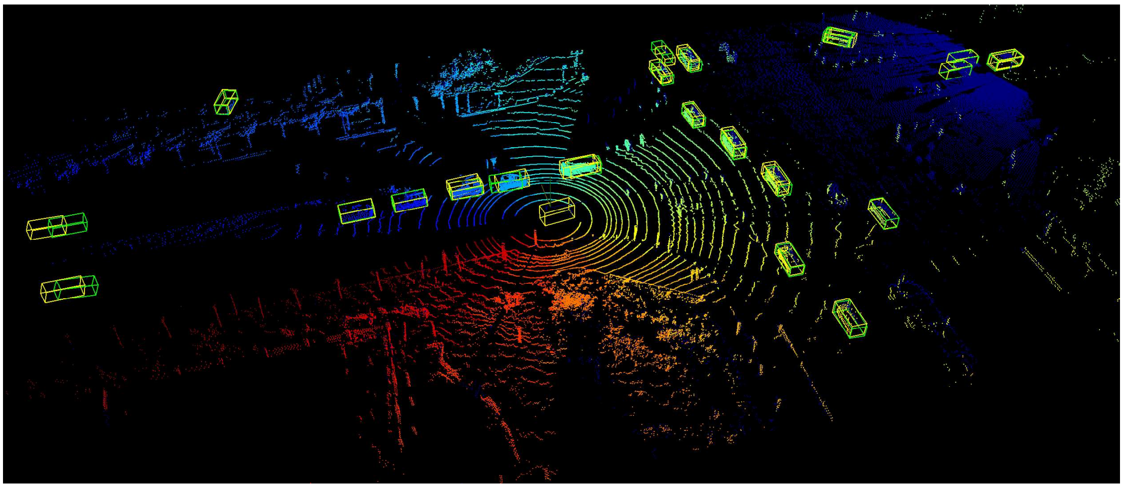
*Figure 5.4: DairV2X dataset: LiDAR data with 3D bounding box annotations (green). Magenta points: point cloud from vehicle. Cyan points: point cloud from infrastructure.*

*DairV2X* (Yu et al., 2022) is a real dataset captured with two IAs, one CAV and one connected intelligent infrastructure (CI) at 28 intersections. The dataset comprises approximately 11.4K frames of both LiDAR and camera data, in which 4.8K frames are used for training and 6.6K frames are used for evaluation of the object detection task. An example frame of LiDAR data is demonstrated in Figure 5.4. The detection range of this dataset is set to $[-100,100]m$, $[-40,40]m$, and $[-3.0,1.0]m$ along the x-, y-, and z-axes, respectively.

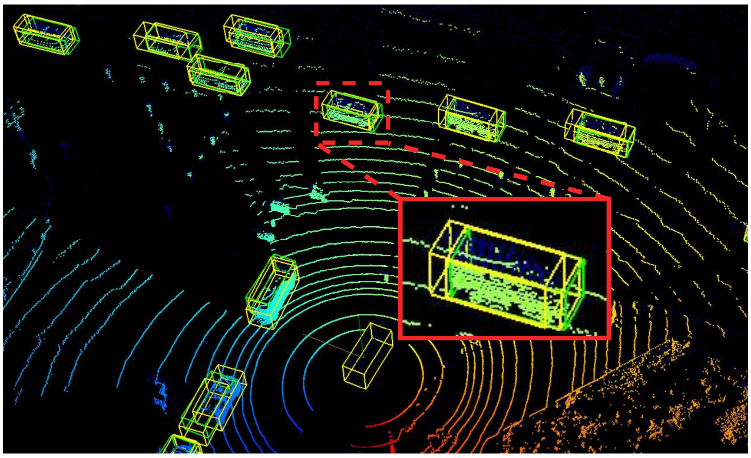
### 5.2.3 DairV2Xt

*DairV2Xt* is adapted from the dataset *DairV2X* (Yu et al., 2022). Since the *DairV2X* dataset has kept the timestamps for each point in the point cloud data, this thesis extends this dataset for the task of time-aligned cooperative object detection; this extended dataset is called *DairV2Xt*. The extension is done through two steps: global registration and global bounding box interpolation for generating globally time-aligned ground-truth bounding boxes. The details of the meta data generation for *DairV2Xt* are demonstrated in Algorithm 5.

To utilize the sequential data for modeling the temporal context, the spatial data alignment between consecutive frames should be accurate. This is not satisfied in the original *DairV2X* dataset. Therefore, this work first refines this dataset by sequentially registering each point cloud scan of the CAV into a unified global reference coordinate $O_{ref}$ (Algorithm 5, line 12), which is



*(a) One frame point cloud data of DairV2Xt*



*(b) DairV2Xt zoom-in view*

*Figure 5.5: DairV2Xt dataset: The points scanned in one frame are observed at different timestamps (blue to red scaled colors). The objects observed by two sensors are captured at different times, resulting in unaligned ground truth bounding boxes (yellow boxes). The bounding boxes at the last time point of this frame are taken as the final ground truth for the time-aligned cooperative object detection (green boxes).*

the CAV pose at the frame where the CAV is closest to CI. Based on the registration, a global merged and down-sampled point cloud is obtained. Then, the CI point clouds are registered to obtain the relative location to $O_{ref}$ (Algorithm 5, line 14-19). With the corrected sensor poses in each frame, the annotated objects are transformed from the local sensor coordinate to the globally aligned coordinate $O_{ref}$. This process ensures that the objects are aligned frame-by-frame with their respective trajectories in the global coordinate system (Algorithm 5, line 20-32). By interpolation over the trajectories, the ground-truth bounding boxes that are aligned to the scan end time point $t_{aligned}$ in each frame are generated (Algorithm 5, line 33-37). An example frame of the extended dataset *DairV2Xt* is shown in Figure 5.5b. The yellow bounding boxes are the local ground-truth bounding boxes that are aligned to the local scan data before interpolation. The green bounding boxes are the globally time-aligned bounding boxes after the interpolation. The detection range of this dataset at the y- and z-axis is the same as *DairV2X*.

**Algorithm 4** OPV2Vt Generation

```
Input: OPV2V meta info $\mathcal{F}$
$\Delta t$                                                ▷ Time interval between two adjacent frames
$\mathcal{F} = \{(\mathbf{v}_i,\mathbf{s}_i,t_i)|i = \{1,2,\ldots,N_f\}\}$,    ▷ Frame data $f_i$ at all timestamps $t_i$
$\mathbf{v} = \{v_i|i = \{1,2,\ldots,N_v\}\}$ ,            ▷ Vehicle info in the i-th frame
$\mathbf{s} = \{s_i|i = \{1,2,\ldots,N_s\}\}$ ,            ▷ Sensor info in the i-th frame
$v = (id,x,y,z,h,w,l,\theta)$                             ▷ bounding box parameters of the j-th vehicle
$s = (id,x,y,z,\theta)$                                   ▷ Sensor parameters of the j-th vehicle
Output: Generated dataset $\mathcal{D}$
Initialize: $\mathcal{G} = \varnothing$, $\mathcal{D} = \varnothing$, $\delta t = \Delta t/10$

for $i \leftarrow 2$ to $N_f$ do                            ▷ Loop over all frames
    for $k \leftarrow 1$ to 10 do                         ▷ Interpolate adjacent two frames into 10 subframes
        $\mathbf{v} = \mathbf{s} = \varnothing$
        for $j \leftarrow 1$ to $N_v$ do                  ▷ Interpolate over the parameters of each vehicle
            if $v_{i-1,j}$ exists then
                $\mathbf{v} \leftarrow \mathbf{v} + \text{Interpolate}(v_{i-1,j},v_{ij},k)$
            end if
        end for
        for $j \leftarrow 1$ to $N_s$ do                  ▷ Interpolate over the parameters of each sensor
            if $s_{i-1,j}$ exists then
                $\mathbf{s} \leftarrow \mathbf{s} + \text{Interpolate}(s_{i-1,j},s_{ij},k)$
            end if
        end for
        $t = t_i + k \cdot \delta t$                      ▷ New timestamp for current subframe
        $\mathcal{G} \leftarrow \mathcal{G} + (\mathbf{v},\mathbf{s},t)$    ▷ Save data for current subframe
    end for
end for
$\mathcal{P} = \varnothing$
for $(\mathbf{v},\mathbf{s},t)$ in $\mathcal{G}$ do          ▷ Simulation replay over all new subframes
    $\mathbf{p} \leftarrow \text{CarlaPlay}(\mathbf{v},\mathbf{s},t)$    ▷ Generate 1/10 lidar scans for each vehicle in $g$
    $\mathcal{P} \leftarrow \mathcal{P} + (\mathbf{p},t)$              ▷ Save scanned point clouds
end for
$\mathcal{P} \leftarrow \text{Sort}_t(\mathcal{P})$                    ▷ Sort data in ascending order of $t$
$t_{\max} = \max(t|(\mathbf{p},t) \in \mathcal{P})$
for $j \leftarrow 1$ to $N_s$ do                            ▷ Compose full scans with random sensor ticking times
    $\tau = \text{RandomInteger}(1,5) \cdot \delta t$         ▷ Generate random sensor ticking time
    while $\tau < t_{\max}$ do
        $P \leftarrow \{p_j|(p_j,s_j) \in \mathbf{p},(\mathbf{p},t) \in \mathcal{P},\tau \le t < \tau + \Delta t\}$    ▷ Full scan
        $s = s_j$ where $(p_j,s_j) \in \mathbf{p},(\mathbf{p},t) \in \mathcal{P},t = \tau$                            ▷ Sensor pose
        $G \leftarrow \mathbf{v}$ where $(\mathbf{v},\mathbf{s},\tau + \Delta t) \in \mathcal{G}$                      ▷ Ground-truth bounding boxes
        $\mathcal{D} \leftarrow \mathcal{D} + (P,G,s,\tau)$
        $\tau \leftarrow \tau + \Delta t$
    end while
end for
```

**Algorithm 5** DairV2Xt Generation

```
Input: DairV2X dataset $\mathcal{F}$
$\Delta t$    ▷ Time interval between two adjacent frames
$\mathcal{F} = \{(P_i,Q_i,\mathbf{v}_i^P,\mathbf{v}_i^Q,s_i^P,s_i^Q,t_i) | i \in \{1,2,\ldots,N_f\}\}$,    ▷ Frame data at all times $t_i$
$P = \{x_i,y_i,z_i,t_i | i \in \{1,2,\ldots,N_P\}\}$,    ▷ CAV Point cloud in each frame
$Q = \{x_i,y_i,z_i,t_i | i \in \{1,2,\ldots,N_Q\}\}$,    ▷ Infrastructure Point cloud in each frame
$\mathbf{v}^* = \{v_i | i = \{1,2,\ldots,N_v\}\}$ ,    ▷ bounding boxes in point cloud $*$
$s^* = (x,y,z,\theta)$    ▷ Sensor parameters for point cloud $*$
$v = (x,y,z,h,w,l,\theta)$    ▷ bounding box parameters
Output: Generated dataset $\mathcal{D}$
Initialize: $\mathcal{D} = \varnothing$

$idx \leftarrow \operatorname{argmin}_i dist(s_i^P,s_i^Q)$    ▷ Index of min. norm-2 distance of sensor positions
$O_{\text{ref}} \leftarrow P_{idx}$    ▷ Global reference frame
for $i \leftarrow idx$ to $N_f$ do    ▷ Register $P$ for $i > idx$
    $(O_{\text{ref}},s_i^P) \leftarrow$ Register$(O_{\text{ref}},P_i)$
end for
for $i \leftarrow idx$ to 1 do    ▷ Register $P$ for $i < idx$
    $(O_{\text{ref}},s_i^P) \leftarrow$ Register$(O_{\text{ref}},P_i)$
end for
for $i \leftarrow 1$ to $N_f$ do
    $(\cdot,s_i^Q) \leftarrow$ Register$(O_{\text{ref}},Q_i)$    ▷ Register $Q$
    $\mathbf{v}_i^P \leftarrow$ Transform$(\mathbf{v}_i^P,s_i^P)$    ▷ Transform bounding boxes of $P$ to aligned frame $O_{\text{ref}}$
    $\mathbf{v}_i^Q \leftarrow$ Transform$(\mathbf{v}_i^Q,s_i^Q)$    ▷ Transform bounding boxes of $Q$ to aligned frame $O_{\text{ref}}$
    for $(\mathbf{v},R_i)$ in $\{(\mathbf{v}_i^P,P_i),(\mathbf{v}_i^Q,Q_i)\}$ do
        $\mathbf{u} \leftarrow \varnothing$
        for $v$ in $\mathbf{v}$ do    ▷ Timestamping each bounding box
            $t \leftarrow$ RetriveTimestamp$(v,R_i)$
            $\mathbf{u} \leftarrow \mathbf{u} + (t,v)$
        end for
        $\mathcal{T} \leftarrow$ Track$(\mathcal{T},\mathbf{u})$    ▷ Track bounding boxes
    end for
end for
for $i \leftarrow 1$ to $N_f$ do
    $t \leftarrow \max_t(\{t | (x,y,z,t) \in P_i\})$    ▷ Align global timestamp to maximum $t$ in $P$
    $\mathbf{v} \leftarrow$ Interpolate$(\mathcal{T},t)$    ▷ Get bounding boxes at time $t$
    $\mathcal{D} \leftarrow \mathcal{D} + (P_i,Q_i,s_i^P,s_i^Q,\mathbf{v})$    ▷ Save data for frame $i$
end for
```

# 6 Framework Design

Collective perception requires processing data from multiple IAs. Although it can be implemented in various ways, certain processing units often share common features. Therefore, conceptualizing these units as modules with distinct functionalities is beneficial for simplifying the explanation of methodologies employed in this thesis. From a programming perspective, such modularization enhances model development and training efficiency, particularly for computationally demanding collective perception tasks that require processing large volumes of data from multiple IAs. This section introduces a unified framework, *CoSense3D*, for collective perception, encompassing both object detection and semantic segmentation tasks. To this end, a task formalization of the collective perception framework is first described in Section 6.1. The structural details of the framework are then outlined in Section 6.2. The deep learning modules within this framework form the core focus of this thesis. They are integral components used repeatedly in the models discussed in later chapters. Therefore, these modules are comprehensively detailed in Section 6.3. Based on the proposed framework, the last section compares object detection experiments that utilize various gradient calculation schedules for different IAs to identify the most efficient model training approach that might be used for the models introduced in the later chapters.

## 6.1 Formalization of Collective Perception

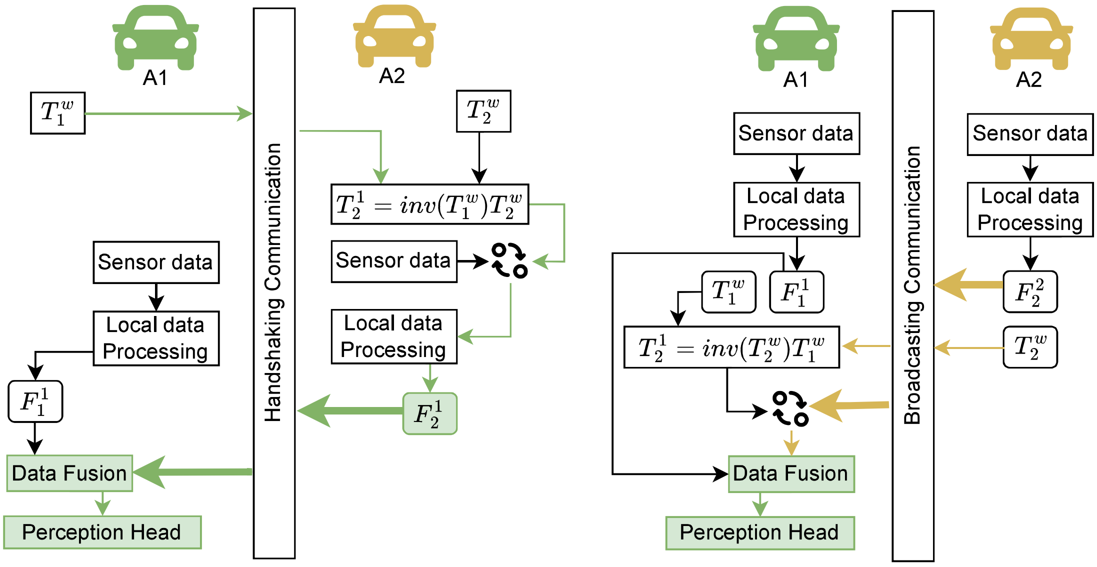


*Figure 6.1: Collective perception formalization in two communication modes. Left: handshaking communication. Right: broadcasting communication.*

In a collective perception scene, as exemplified in Figure 1.1 of Section 1.1, consider $N+1$ IAs, denoted as $\mathbf{A} = \{A_1, \ldots, A_N\}$, capable of sharing information with each other. Each IA is equipped with a LiDAR sensor. Instead of analyzing the collective perception results from the perspective of all IAs, this thesis focuses on the ego vehicle, $A_1$. This requires only the data fusion process of the ego vehicle, hence reduces the computational requirement and simplifies the experiment settings. Specifically, one vehicle is designated as the ego vehicle for each sequence in the dataset, while all other IAs are treated as cooperative agents that share CPMs with the ego vehicle. These

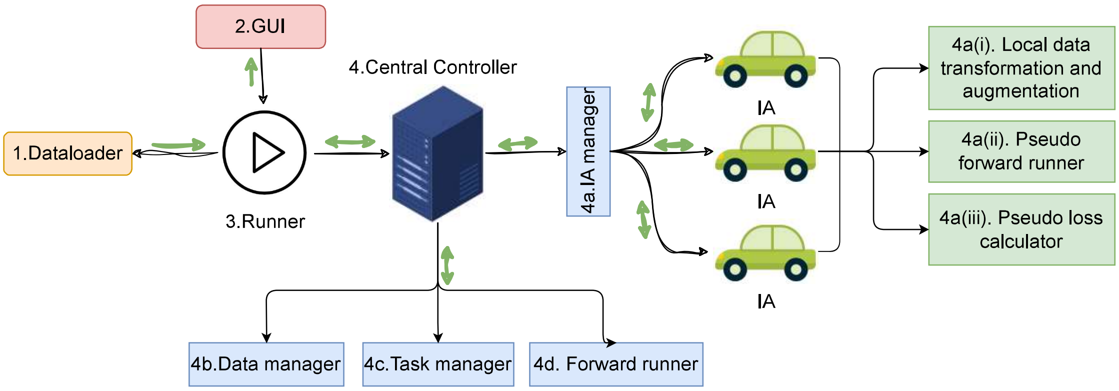


*Figure 6.2: CoSense3D: Agent-based training framework. Black errors indicate the instruction passing direction, green arrows indicate the data passing direction.*

cooperative agents are restricted to those within the communication range $R$ of $A_1$, represented as $\mathbf{A}_{\text{nbr}} = \{A_j | d(A_i,A_j) < R\}$ where $d$ is the distance between two agents.

The IAs are communicating with each other either with *handshaking communication* or *broadcasting communication* as illustrated in Figure 6.1. In *handshaking communication*, the ego vehicle first sends its pose $T_1^w$, the transformation matrix from ego to the world coordinate system, to the cooperative IAs. By receiving the request CPM, the cooperative IAs compute the transformation matrix $T_2^1$ from the cooperative to the ego coordinate system. Utilizing this matrix, the cooperative sensory data is projected to the ego coordinate system and subsequently processed to extract deep features denoted as $F_2^1$. These features, encapsulated as the response CPM, are then transmitted to the ego vehicle (thick green arrows). The ego vehicle then fuses all response data and generates the final collective perception result with the perception head, *e.g.,* object detection head or semantic segmentation head as described in Section 3.2.3. In *broadcasting communication*, both the ego and the cooperative sensory data are first processed locally to get the deep features $F_1^1$ and $F_2^2$, respectively, in their own coordinate systems. Then the cooperative vehicle broadcasts the feature $F_2^2$ together with its pose $T_2^w$ to the ego vehicle so that a transformation $T_2^1$ is computed to transform the received cooperative feature into the ego coordinate system. At last, the data from the ego vehicle and the cooperative vehicles are fused for the final perception head.

## 6.2 CoSense3D: a Efficient Framework for Collective Perception

The *CoSense3D* framework is outlined in Figure 6.2, comprising four main modules: *1. Dataloader*, *2. GUI*, *3. Runner*, and *4. Central Controller*. The Central Controller serves as the core of the framework, containing four sub-modules (colored in blue):

- *4a. IA Manager*: features three key functionalities: *4a(i). Local Data Transformation and Augmentation*, *4a(ii). Pseudo Forward Runner*, and *4a(iii). Pseudo Loss Calculator*. This module is responsible for prototyping the properties of each IA. For example, the ego and cooperative IAs can be configured with distinct data transformations, processing strategies, and loss calculation requirements.
- *4b. Data Manager* facilitates the distribution and aggregation of data among IAs.
- *4c. Task Manager* consolidates task specifications from the IAs.
- *4d. Forward Runner* executes the forward propagation of the deep learning model based on the provided specifications.

In the diagram, black arrows represent instruction flow, while green arrows denote data flow. The framework can operate with or without visualization in the GUI. Additional information about the APIs offered by this development tool is available on the project's main page[1]. In the following, the details of these modules are described.

### 1.Dataloader

The framework standardized the data loading API for collective perception with a predefined dictionary format to store the meta information in JSON files. With this API, a new dataset can be easily converted to the CoSense3D format without rewriting the PyTorch Dataloader and copying the large media files, such as point clouds and images, to a new folder structure. Only the meta information such as scenarios, frames, timestamps, parameters of sensors, and the annotations are parsed and saved to the CoSense3D format in JSON files. This standardized Dataloader is able to load images, point cloud data, 2D annotations for images, 3D local annotations for perception without IA cooperation, and 3D global annotations for collective perception.

### 2.GUI

The Graphical User Interface demonstrated with the red box in Figure 6.2 is designed to visualize the training and test data and check the training and test outcomes by one click. This is helpful for loading new datasets and developing new models. Before training on a new dataset, it is necessary to check if the data is converted and loaded correctly. During and after training, visualizing the model output is also helpful to identify the drawbacks and problems of the model and then refine or modify the model accordingly.

As shown in Figure 6.2, the GUI sends commands to the runner to start, stop, or step the runner process. After each runner step, it updates the visualization modules, 3D GLViewer, ImgViewer, ImgAnno3DViewer, and OutputViewer. As shown in Figure 6.3a, GLViewer is an OpenGL-based visualizer for 3D data, annotations (green boxes) and predictions (red boxes). ImgViewer shows image data and the corresponding 2D bounding boxes. ImgAnno3DViewer is used to check if the transformations and augmentations of images and 3D annotations are correctly loaded and processed. Each row in ImgViewer and ImgAnno3DViewer shows the images of a single IA. After training the model, the OutputViewer can be used to visualize the test result. The OutputViewer can contain multiple Canvas, which can be customized by the user. Figure 6.4 is an example that shows the BEV segmentation (top) and object detection (bottom) result.

### 3.Runner

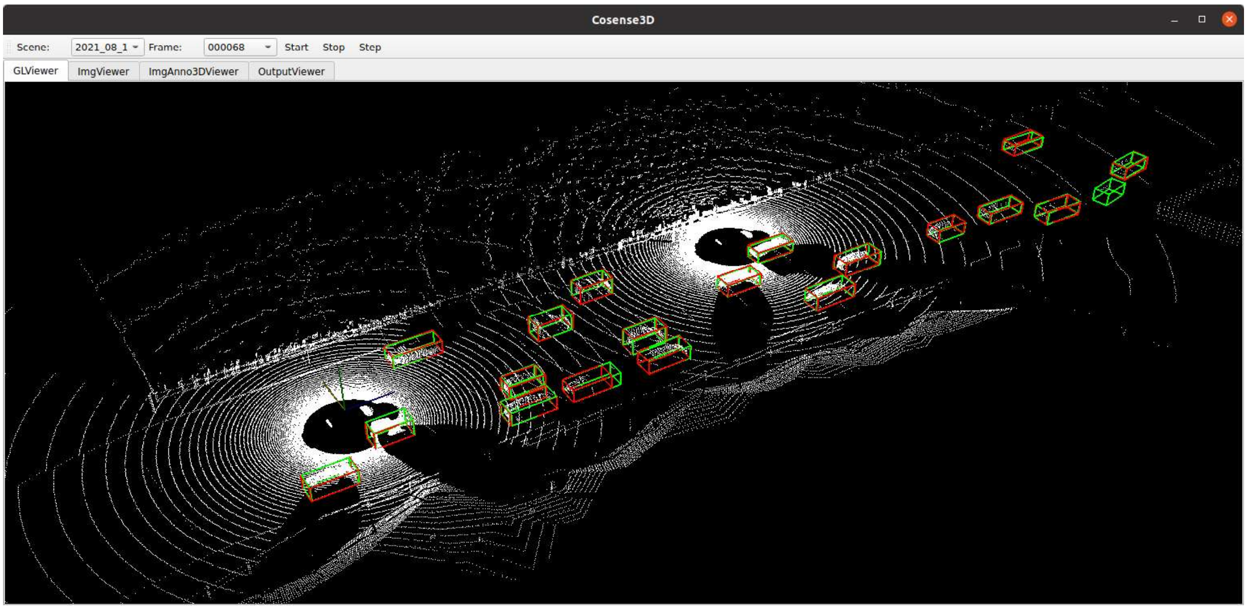
In this framework, three types of Runners are available: *TrainRunner*, *TestRunner*, and *VisRunner*. Users have the option to launch these Runners with or without a GUI. These Runners are specifically utilized for training, testing, and visualizing input data, respectively. They are responsible for dispatching data and orders to the Central Controller, which subsequently processes the orders in conjunction with the provided frame data. In return, the Runners read the processed data, *e.g.,* the detected object, for visualization in the GUI.

### 4.Central Controller

As illustrated in Figure 6.2, the Central Controller is the core module of the *CoSense3D* framework. It streamlines the data processing pipeline by delegating data loading, processing, sharing, and fusion tasks among three dedicated manager modules: the *IA Manager*, *Data Manager*, and *Task Manager*. This modular approach allows users to focus solely on defining the data processing strategies for each IA. For example, the ego and cooperative IAs can be configured with different

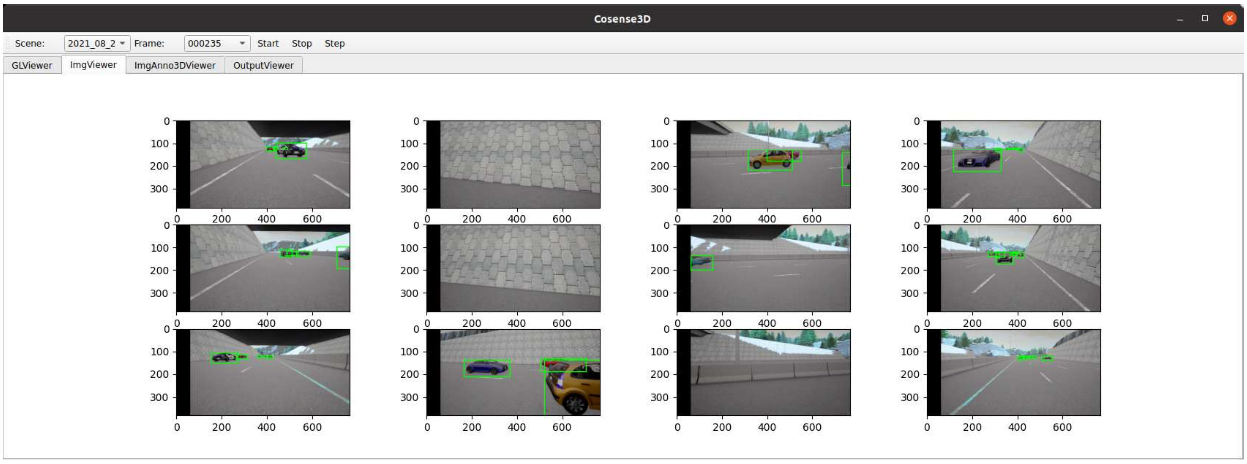
---

[1] `https://github.com/YuanYunshuang/CoSense3D`



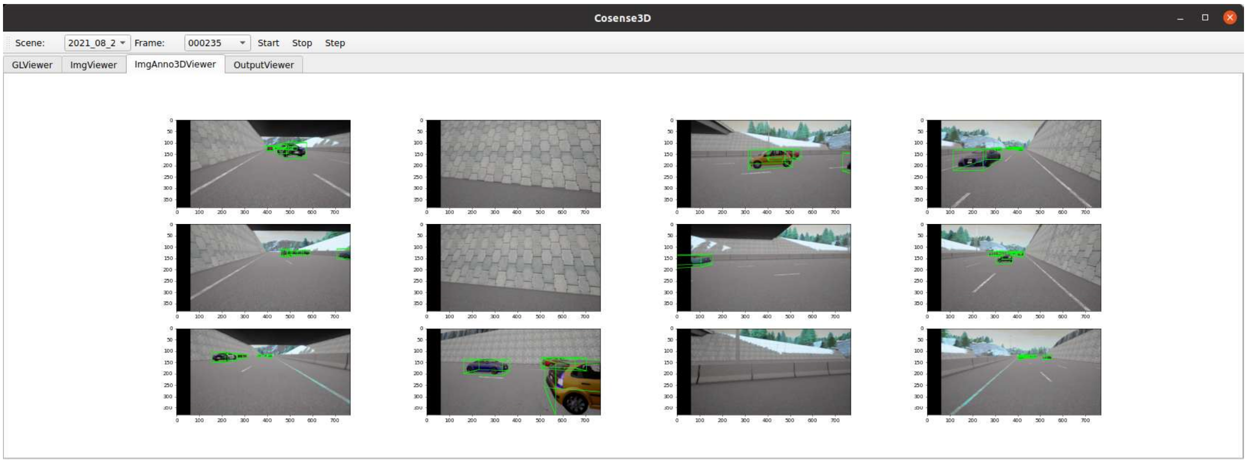
Scene: 2021_08_1 Frame: 000068

(a) GLViewer



Scene: 2021_08_2 Frame: 000235

(b) ImgViewer



Scene: 2021_08_2 Frame: 000235

(c) ImgAnno3DViewer

Figure 6.3: CoSense3D GUI.

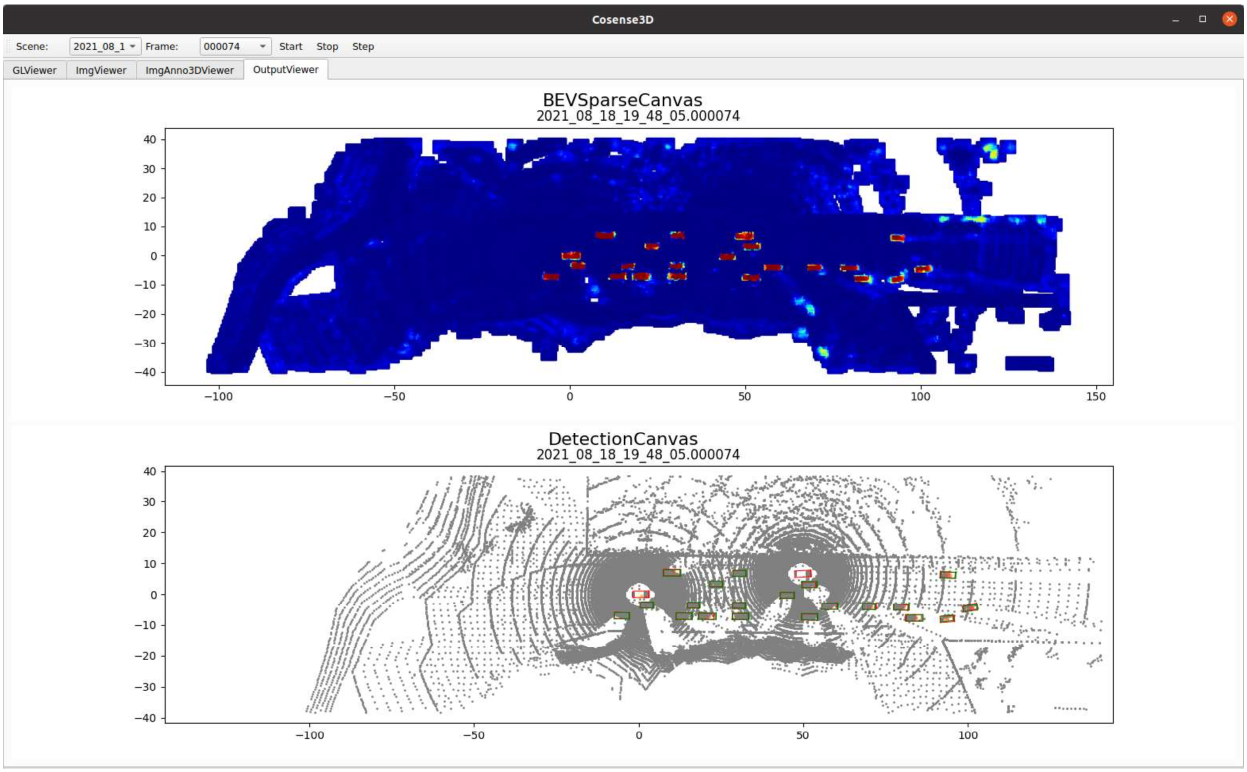


*Figure 6.4: CoSense3D GUI OutputViwer.*

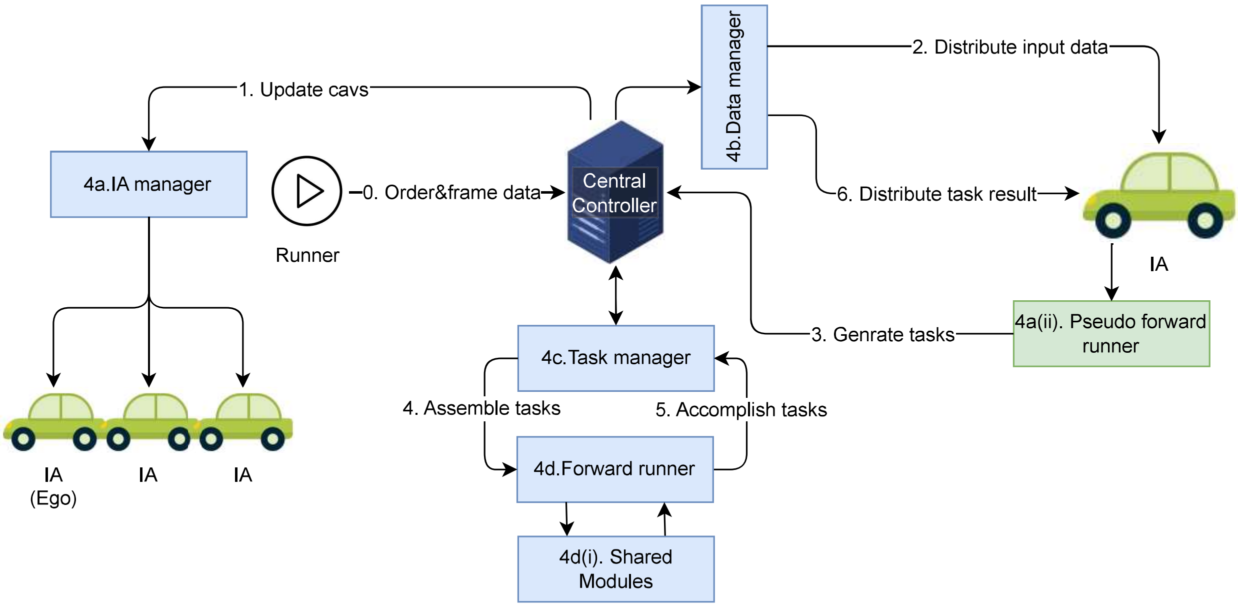


*Figure 6.5: Workflow of CoSense3D Central Controller.*

coordinate transformation strategies or set to calculate gradients selectively for specific modules during training. This process of defining the functional features of IAs is referred to as *prototyping.* Once the agents are prototyped, the Central Controller orchestrates data processing, utilizing the automated capabilities of the *Data Manager* and *Task Manager* to ensure efficient handling and integration of information.

More specifically, the Central Controller communicates with the order-dispatcher (*Runner*) and the IAs through its *IA Manager*. The *Data Manager* handles data gathering and distribution between the Central Controller and the IAs. Similarly, the *Task Manager* collects task descriptions (referred to as pseudo tasks) generated by the IAs, batches these tasks, and dispatches them to the *Forward Runner*, which houses all shared deep learning modules. To facilitate customized workflows for collective perception, the framework provides a standardized IA prototyping API. This API enables users to define tailored workflows, including the data augmentations, IA coordinate transformations, CPM sharing strategies, neural network modules as well as the forwarding order and gradient computation strategies of these modules.

Based on the predefined IA prototypes, the Central Controller operates in a standardized pipeline, as illustrated in Figure 6.5.

- Step 0: The Central Controller receives an order and frame data from the Runner.
- Step 1: The *IA Manager* updates the IAs using the meta information in the frame data and the predefined IA prototypes.
- Step 2: The *Data Manager* distributes the input data for the current frame to the updated IAs.
- Step 3: Upon receiving the input data, the IAs preprocess the data, generate tasks, and return them to the Central Controller.

Gradient computation for back-propagation is memory-intensive, as it requires storing intermediate results. To optimize efficiency, gradient calculations can be selectively omitted for certain parts of the data without deteriorating training results.

- Steps 4 and 5: The *Task Manager* consolidates tasks from all IAs and organizes them into two processing blocks for the *Forward Runner*: one block includes tasks requiring gradient computation, and the other excludes them to conserve GPU memory (for details on enabling or disabling gradient computation, see Section 6.4.1).

Finally, the *Forward Runner* processes these tasks in parallel, and the resulting outputs are distributed back to the respective IAs.

## 6.3 Standardized Modules of CoSense3D

### 6.3.1 Overview

The main data processing modules, including the deep learning modules, are defined in the Forward Runner. As illustrated in Figure 6.5, the Forward Runner executes forward running instructions which are summarized from the instructions of each IA. These instructions are sequentially executed over the shared neural network modules. The execution pipeline of these forwarding instructions is visualized in Figure 6.6. The data flow with gradient calculation is illustrated with a yellow background and solid lines, and the flow without gradients is with a white background and dashed lines. The point cloud data from different IAs are first augmented by the *Aug.* module which normally contains a *free space augmentation* and a *geometric augmentation* which are introduced in Section 6.3.2. The augmented point clouds are then encoded by a backbone network to 3D

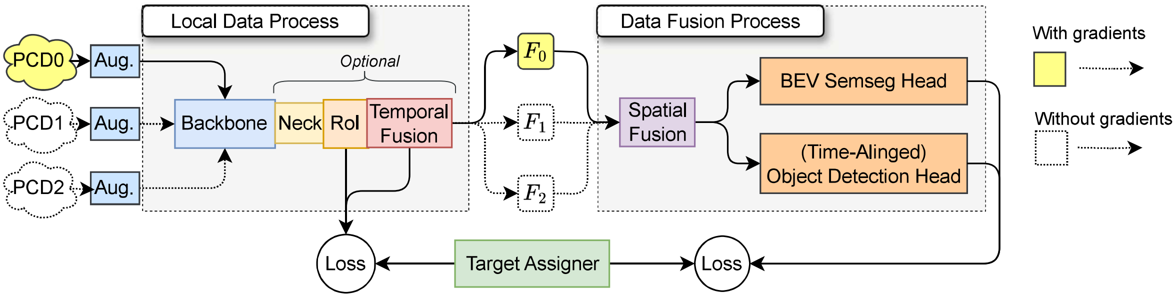


*Figure 6.6: Overview of the pipeline for shared modules*

spatial features of different resolutions. After the backbone network, some optional modules can be attached. In this work, three optional modules, *Neck*, *RoI* and *Temporal Fusion*, are used to construct the proposed models. Details about these modules are discussed in Section 6.3.3. The outcome of the backbone and/or the optional modules are the final CPM features for sharing with the ego-vehicle, in which these features are then fused by the *Spatial Fusion* module. According to the requirements, the fused features can then be used for the final collective perception task, including the *BEV Semantic segmentation* (BEV Semseg, Chapter 7) and *Time-Aligned Cooperative Object Detection* (TA-COOD, Chapter 8). The *Target Assigner* is the module for generating the ground truth learning target for the corresponding heads.

### 6.3.2 Data Augmentation

In deep learning, data augmentation is a technique used in data preprocessing to increase the diversity of data available for training models without actually collecting new data. For point-cloud data, *geometric augmentation*, such as rotation, scaling, translation, and flipping, are often used. In this thesis, a new technique for point-cloud preprocessing is introduced by augmenting the point clouds with free space points that lie on the LiDAR scanning rays. Since it augments the point clouds with more points, it is referred to as free space augmentation. In the following, the details of these two augmentation techniques are introduced.

#### Free space augmentation (FSA)

In a point cloud, the free space – traversed space by the ray that is not occupied by any obstacles – is often neglected. However, this information is also a result of the measurement and is critical for identifying the occupancy and visibility of the driving space. Therefore, the point cloud data can be augmented by sampling points from the LiDAR ray paths. This augmentation is called *free space augmentation* and the sampled points are called *free space points*, notated as $f_{si}$, where $i \in \mathbb{N}$. As exemplified in Figure 6.7, a LiDAR (orange cylinder) casts a ray (red line) that hits the surface of the ground at point $fs_0$ and only records this reflected point into the point cloud. Over the ray path, free space points $f_{si}$ are sampled in the limited distance $d_{fs}$ from the hit point $f_{s0}$ with a large step $s_{fs}$. In order to constrain the computational overhead, only the driving-critical region of a limited height, *i.e.,* $z \leq h_{fs}$, over the ground (blue area) is employed for sampling. Finally, these points are down-sampled again by voxel down-sampling with a given voxel size of $v_{fs}$ to obtain evenly and sparsely distributed free space points. These augmented free space points are then added to the original point cloud by setting their intensity value $\mathfrak{i} = -1$ as the indicator.

#### Geometric augmentation

Geometric augmentation involves applying geometric transformations to input data. In this work, point clouds are augmented using random rotations along the $z$-axis and random flips along the

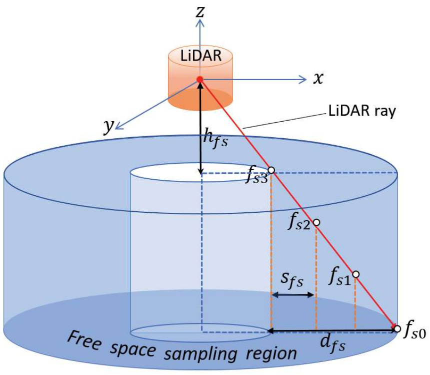


*Figure 6.7: Sampling free space points. The red point is the origin of the LiDAR coordinate system, $z$-axis indicates the vertical direction in the driving space, $x$-axis indicates the horizontal direction. $f_{s0}$ is the intersection point of the LiDAR ray (red dashed line) and the ground. $f_{s1}$,$f_{s2}$, and $f_{s3}$ are the sampled free space points belonging to the ray path.*

$x$- and $y$-axes. These transformations enable the training process to encounter targets in a wider range of orientations, enhancing model robustness. Additionally, the point clouds are randomly rescaled to vary the sizes of target objects, further diversifying the training data. The rotation angle and scaling factor are denoted as $r_{\mathrm{aug}}$ and $s_{\mathrm{aug}}$, respectively.

### 6.3.3 Optional modules

#### Neck

In general, certain frameworks may require the transformation of 3D backbone features into a format compatible with subsequent modules. The *Neck* module facilitates this transformation by acting as an intermediary between the Backbone and the Region of Interest (RoI) module, serving as a neck connecting these two components. In this work, three types of Neck modules are used, namely *Height Compression (HC)*, *Dilation Conv (DConv)* and *Map shrinker (Mshrink)*. The height Compression module contains several layers of convolutions with stride one over the $x$- and $y$-axis and stride larger than one over the $z$-axis so the features along the vertical $z$-axis can be compressed to one channel. The outcome of this module is a 2D feature map. The convolution layers employed in this module can be either dense or sparse, depending on the format of the input features. Dilation Conv is used to expand the 2D coordinate coverage of the input sparse features. It normally contains several expandable sparse convolution layers. The Map shrinker is used to shrink the 2D feature maps to the required size, and it contains several 2D dense convolution layers.

#### RoI head

To increase the efficiency of the whole model, RoI heads can be used during the *Local Data Process* to select data in the interesting regions for further processing. It contains one or several sub-heads, which can be a classification head ($\mathcal{H}_{\mathrm{roi}}^{\mathrm{cls}}$) that generates a ranking score for the input feature points or a detection head ($\mathcal{H}_{\mathrm{roi}}^{\mathrm{det}}$) that selects the object proposals for the processing in the next steps. These sub-heads can either be learned by back-propagating the loss or by the gradients back-propagated from the modules in the next steps.

#### Temporal Fusion

In case the input point clouds are sequential data, the *Temporal Fusion* module aims to fuse the information from previous frames so that the fused features also contain the temporal context. In this work, this module is designed before the data sharing process because the sequential data is normally very heavy in size. Fusing the temporal information locally at the IA can significantly reduce the CPM size. Inspired by StreamPETR (Wang et al., 2023), this thesis introduces a Temporal Fusion module for the object detection task, which will be discussed in Chapter 8.

### 6.3.4 Spatial Fusion

The *Spatial Fusion* module contains three steps: pose alignment, feature alignment, and feature fusion. Pose alignment aims to correct the relative pose errors between the ego and the cooperative agents. Then, the features from different IAs can be spatially aligned using the corrected relative poses. Once the features are aligned to the correct coordinate system, they can be fused using a specific fusion method, such as Maxout or Attention as introduced in Section 3.2.2. In this work, sparse fusion methods are introduced in Chapter 7 and Chapter 8.

### 6.3.5 Object detection head

The object detection head contains a classification and a regression sub-head. Given a feature point $q = \{x,y,F\}$, where $x,y$ are the coordinates of the point and $F \in \mathbb{R}^{d_{\text{feat}}}$ is the feature vector that describes this point, the classification head uses linear layers to map the input features dimension $d_{\text{feat}}$ to the output dimension $d_{\text{cls}}$, which equals the number of classes. Similarly, the regression head projects the input features into the dimension $d_{\text{reg}}$, which corresponds to the bounding box encoding dimension. This dimension represents the number of target parameters necessary for describing the bounding box. The activation in the last linear layers is either omitted to generate regression values without range constraints or retained to generate values in the preferred range. For example, employing the sigmoid function, the output range is then $[0,1]$.

For the input 2D feature map $F \in \mathbb{R}^{W \times H \times C}$ that contains feature points that are arranged in grid format, the object detection head performs its classification and regression over each location in the 2D grid for $N_{\text{anchor}}$ times. $N_{\text{anchor}}$ is the number of the anchors used for each location. Directly regressing the parameters of the ground-truth bounding boxes might lead to unstable training process of DNNs because these parameters might contain very large values. Therefore, the regression targets are normally normalized and encoded based on some reference values. One kind of reference values is from the predefined anchor bounding boxes as described in eqs. (3.19) to (3.22) to encode the target values. Therefore, the output dimension of the regression head is $d_{\text{reg}} = 7$. The above described detection head is notated as *Dense Anchor-Based detection* head, and *denseAdet* or $\mathcal{H}_{\text{dAdet}}$ in short.

For the sparse input features, this work proposes a refined center-based object detection head that tends to optimize the orientation of the detected bounding boxes. It is notated as *Sparse Center-Based detection* head, and *sparseCdet* or $\mathcal{H}_{\text{sCdet}}$ in short. As described in Section 3.2.3, the original version of the center-based object detection head encodes the object orientation into $[\sin r, \cos r]$. On the one hand, this encoding ensures that the target values of the orientation are in the range of $[-1,1]$, benefiting the stability of the learning process. On the other hand, it encodes the orientation angles into a continuous 2D space and avoids the sign flipping problem of the orientation where $r = -\pi$ and $r = \pi$ represent the same direction. However, the encoding using Equation (3.25) tends to generate worse orientation results than the direct distance encoding with Equation (3.22), because the latter one is linear and easier to learn. By combining the advantages of both encodings, this work proposes a novel encoding method called *CompassRose* encoding as it performs regressions with respect to the four orientation angles $\mathbf{r}_a = [0,0.5\pi,\pi,1.5\pi]$ which correspond to the four directions in the compass rose as shown in Figure 6.8.

The left part of Figure 6.8 shows a cosine wave with the $0\pi$ angle marked with a vertical dashed line. In the conventional way, the center-based detection head encodes this angle with sine and cosine. Taken the cosine part as an example, the unique anchor angle at $r_a = 0\pi$ needs to go

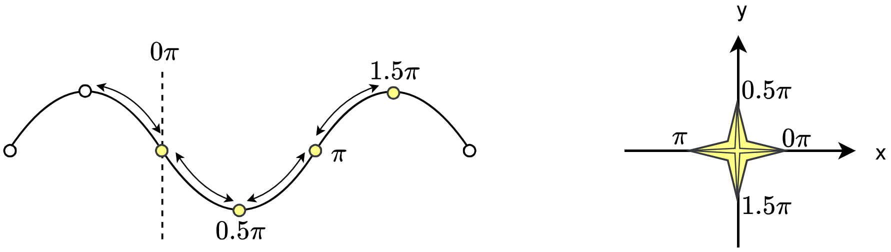


*Figure 6.8: Compass rose encoding for orientation of bounding box*

downhill and uphill to reach a target angle at $r_g = 0.8\pi$. This non-monotonic feature introduces difficulty for the regression tasks. To tackle this problem, CompassRose encoding introduces the compass rose angles so that each target angle $r_g$ can be regressed from the nearest compass angle. The pathway to the goal angle is monotonic. As shown in Figure 6.8, these compass rose angles are marked with small circles, and the yellow ones are in one period of the cosine wave. For the sine encoding part, the same strategy is used. The final angle encoding result is described with Equation (6.1) which is a vector with eight target elements which are the sine and cosine distance to the four encoded compass rose angles. To select the best regression result, the scores for the four angles are also generated by the model. They are activated by the softmax function so that the scores lie in the range [0,1]. The target scores are calculated with Equation (6.2) which is one minus the normalized distance of the compass angle to the ground truth angle. The operation arccos, instead of $\mathbf{r}_a - r_g$, is used to control the angle difference to lie in the range [1,$\pi$]. Based on these scores, the regressed angle with respect to the greatest score is selected as the final result. For the location and the dimension of the bounding boxes, Equations (3.23) and (3.24) are used.

$$\mathcal{B}_{\text{dir}} = [\cos r_g - \cos \mathbf{r}_a, \sin r_g - \sin \mathbf{r}_a] \tag{6.1}$$

$$\mathcal{B}_{\text{scr}} = [s_0, s_{0.5\pi}, s_\pi, s_{1.5\pi}] = \mathbf{1} - \arccos\left(\cos \mathbf{r}_a \cdot \cos r_g + \sin \mathbf{r}_a \cdot \sin r_g\right)/\pi \tag{6.2}$$

### 6.3.6 BEV Semseg head

Similarly to the classification sub-head of object detection, the BEV Semseg head also uses linear layers to project the input features of dimension $d_{\text{feat}}$ to the output dimension of $d_{\text{cls}}$. However, the learning target is different. Object detection targets are generated with respect to either the IoU between the anchors and the ground truth bounding boxes or to the ground truth bounding box centers. The BEV Semseg head learns according to the ground truth semantic map, which contains the ground truth semantic labels for each input feature point. In this work, three classes are learned, namely road, vehicle, and background.

Conventionally, the classification scores are generated by applying the softmax function to the logits generated by the classification head. They train this head with the focal loss. This type of classification head is named as *BEV Semseg Focal Head* and mathematically notated as $\mathcal{H}_{\text{bev}}^{(\text{fcl})}$. In addition to the classification scores, this work proposes to estimate the uncertainty of the generated scores with the evidential loss introduced in Section 3.2.4. The BEV classification head with this configuration is notated as $\mathcal{H}_{\text{bev}}^{(\text{edl})}$. The $\mathcal{H}_{\text{bev}}^{(\text{fcl})}$ and the $\mathcal{H}_{\text{bev}}^{(\text{edl})}$ head both generate classification results with respect to the discrete feature points. These points are with fixed resolution once the network is trained. Moreover, in the distant area of the ego-vehicle's view, the points are very sparse which leads to poor coverage over the object and the road surfaces. To this end, this work utilizes a Gaussian distribution to construct the BEV semantic maps, from which one can retrieve semantic labels for any point in the continuous driving space. This head, notated as $\mathcal{H}_{\text{bev}}^{(\text{gev})}$, is one of the core contributions of this work. More details are discussed in Chapter 7.

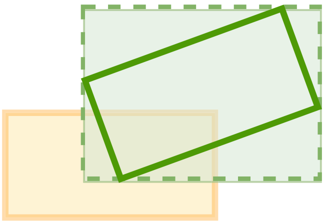


*Figure 6.9: Anchor assignment to fore- and background class. GT box in green, anchor box in yellow.*

### 6.3.7 Target assigner

Target assigner is a module that generates learning targets for the task heads, including the RoI classification head $\mathcal{H}_{\text{roi}}^{\text{cls}}$, the RoI detection head $\mathcal{H}_{\text{roi}}^{\text{det}}$, the dense anchor-based object detection head $\mathcal{H}_{\text{dAdet}}$, the sparse center-based object detection head $\mathcal{H}_{\text{sCdet}}$, as well as different versions of BEV Semseg head, $\mathcal{H}_{\text{bev}}^{(\text{fcl})}$, $\mathcal{H}_{\text{bev}}^{(\text{edl})}$ and $\mathcal{H}_{\text{bev}}^{(\text{gev})}$. These heads may share the same target assignment strategy. In the following, three types of assigners used in this work are explained.

#### Bounding Box Anchor Assigner

Given the predefined anchors at each grid location of the feature maps, the classification head determines if an anchor at a specific location is a foreground anchor that is aligned to one ground-truth bounding box, while the regression head aims to generate regression values that transform the anchor bounding box into the final detection bounding box that is well aligned to the ground-truth bounding box. To train the classification head, the Assigner should assign each anchor a label that defines whether it is fore- or background. This is achieved by calculating IoUs between these anchors and ground-truth bounding boxes.

As shown in Figure 6.9, the ground-truth bounding box is illustrated in green with solid lines, and the orange bounding box is an anchor. The IoU between these two bounding boxes would be very small when the ground-truth bounding box has a very different orientation than the anchor. To remove this sensitivity of IoUs to the orientation angles, the IoUs between the anchor and the axis-parallel ground-truth bounding box (green dashed box) are used for the fore- and background label assignment. Based on the calculated IoUs, two IoU thresholds $thr_{\text{pos}}$ and $thr_{\text{neg}}$ are used to select positive/foreground and negative/background samples. The assignments are expressed with

$$label = \begin{cases} 1, & \text{if } IoU > thr_{\text{pos}} \\ 0, & \text{if } IoU < thr_{\text{neg}} \\ -1, & \text{otherwise} \end{cases} \tag{6.3}$$

where 1 indicates the positive class, 0 the negative class and −1 the samples that are ignored during training. For the regression head, the anchor assigner generates the regression targets with eqs. (3.19) to (3.22) for each positive anchor box. During training, the loss is only calculated over the positive samples.

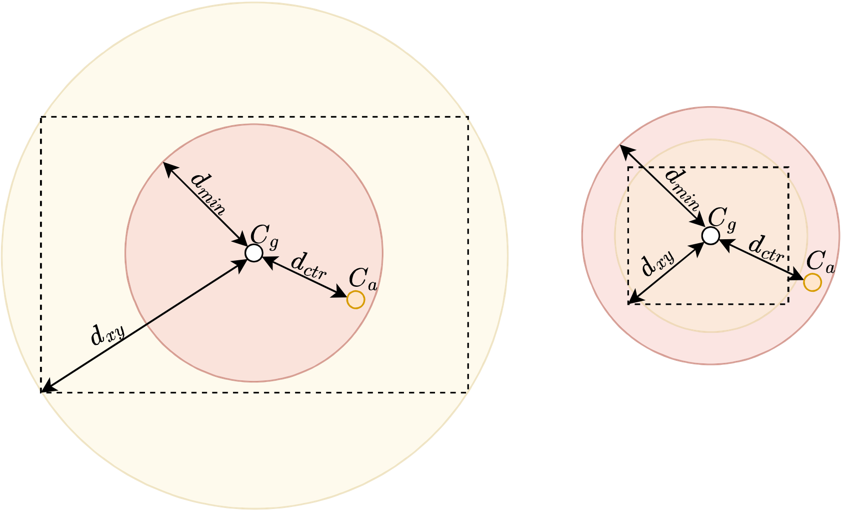


*Figure 6.10: Bounding box Center Assigner. Box with dished lines represents the ground-truth bounding box, white point is the center of this box and the orange points that lies in the red circle is assigned as positive (foreground) object sample.*

### Bounding Box Center Assigner

The Center Assigner generates classification labels with respect to the center of the feature points. Therefore, it is anchor-free. As illustrated in Figure 6.10, the classification labels are generated based on the distance between the center of the feature point (orange point) and the center (white point) of the ground truth bounding box (dashed box). Points in the red area are assigned positive labels, points outside the red circle are assigned negative labels. Mathematically, it is described by

$$label = \begin{cases} 1, & \text{if } d_{\text{ctr}}(C_a,C_g) < max(d_{\text{xy}},d_{\text{min}}) \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

where $d_{\text{ctr}}$ is the Euclidean distance between the feature point $C_a = [x_a,y_a]$ and the center point $C_g = [x_g,y_g]$ of the ground truth bounding box, $d_{\text{xy}}$ is half of the diagonal length of the ground truth bounding box and $d_{\text{min}}$ is a constant value that ensures the feature points $C_a$ in range $d_{\text{min}}$ of the center $C_g$ are assigned as positive samples. As shown on the right of Figure 6.10, there might be no anchor points within the range $d_{\text{xy}}$ when the bounding box size is very small. In this case, $d_{\text{min}}$ ensures that at least one anchor point $C_a$ is assigned to this ground-truth bounding box. The regression targets of this assigner are generated following Equations (3.23), (3.24), (6.1) and (6.2).

### BEV Map Assigner

The BEV map Assigner generates target labels for the BEV Semseg heads. Conventionally, these labels are retrieved from a rasterized ground truth label map which has the same resolution as the model output. Each output point takes the label at the corresponding location in the map as its ground truth target. In this method, all output points are taken as learning samples. Since this method assigns the target labels according to the discrete ground truth label map, it is hence called *discrete BEV Assigner*. It is used for the $\mathcal{H}_{\text{bev}}^{(\text{fcl})}$ and the $\mathcal{H}_{\text{bev}}^{(\text{edl})}$ head.

Differently, the $\mathcal{H}_{\text{bev}}^{(\text{gev})}$ head uses a random sampling strategy to generate learning targets that are scattered in the continuous driving space, therefore called *continuous BEV Assigner*. Given an input point cloud with $N_{\text{pcd}}$ points, $N_{\text{tgt}} = n_{\text{tgt}} \cdot N_{\text{pcd}}$ 2D random sample points are generated. More specifically, each point in the point cloud is regarded as a Gaussian distribution center, $n_{\text{tgt}}$ samples with the offset $x,y \sim \mathcal{N}(0,\sigma_{\text{tgt}})$ to the Gaussian center are generated. In this way, a pre-

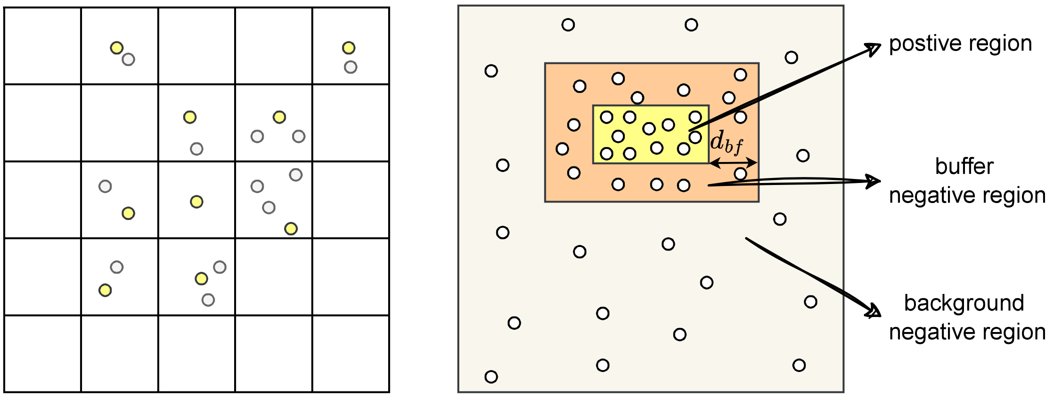


*Figure 6.11: Random sampling of continuous BEV Assigner. Left: global cell-based down-sampling, right: buffer-based sampling around vehicle (yellow).*

defined number $n_{\text{tgt}}$ of target points are generated from the center points to control the density. Note that the generated target points only cover the observed areas as they are random points generated by adding noise (random shifts) to the observation points in the original point cloud.

To prevent memory overflow and excessive computational time during training, the randomly sampled target points are further down-sampled. Due to varying spatial distributions, distinct down-sampling strategies are applied for large objects (*e.g.,* roads) and small objects (*e.g.,* vehicles). For large objects, down-sampling is performed by selecting one point per discretized cell, as shown on the left of Figure 6.11. The yellow points represent the final sampled points, while gray points are discarded. For small objects, which have significantly fewer foreground pixel samples, applying the same cell-based down-sampling in all areas may remove too many critical foreground points. To address this, all points within the bounding box and its buffer area (yellow and orange regions in Figure 6.11) are retained, as shown in the right image of Figure 6.11. Only points outside the buffer area are down-sampled using the cell-based method. The buffer area, with an extent of $d_{\text{bf}}$, enhances the model's ability to learn details along the bounding box edges. Consequently, the sampled points for training are sparse outside the buffer area to save GPU memory, while being dense inside the buffer area to enable the model to achieve fine-grained classification at bounding box boundaries.

## 6.4 Efficiency of CoSense3D

### 6.4.1 Efficient data processing and training

Developing deep neural networks for collective perception demands substantial computational power. This requirement involves loading and processing a significantly larger volume of data compared to conventional single-agent-based perception frameworks. To mitigate this challenge and optimize training efficiency while minimizing GPU memory usage, employing more efficient neural networks and training methods can be instrumental. To this end, the *CoSense3D* framework is specially designed to optimize the model development efficiency of collective perception from two perspectives.

Firstly, existing frameworks for autonomous driving perception such as *mmdetection3d* (MMLab, 2020) and *OpenCOOD* (Xu et al., 2022c) may encounter a CPU bottleneck and slow down the training process during preprocessing, data transformation, and augmentation on a lower-end CPU, particularly evident in collective perception scenarios. In such cases, a single frame of data may contain dozens of images and point clouds, potentially reaching several hundred if the sequential data in the temporal dimension is considered. To alleviate this bottleneck without necessitating hardware upgrades, CoSense3D involves migrating the transformation and augmentation processes to the GPU.

Secondly, computing gradients for all incoming data significantly burdens GPU memory, notably impacting the development of collective perception pipelines, especially with larger models like transformer-based architectures and resource-intensive image processing backbones. To increase development efficiency, CoSense3D provides an API for easy definition of data processing pipelines of IAs, enables flexible control over gradient computation for each IA, optimizing GPU memory usage and reducing the training time. This agent-oriented framework defines the behavior of each agent separately, hence termed as *agent-based framework* and the training processing under this framework is called *agent-based learning.*

### 6.4.2 Experimental analysis of training efficiency

Before the experiments for the collective perception tasks discussed in Chapters 7 and 8, it is beneficial to analyze the trade-offs between the model performance and the training efficiency so that more efficient training strategies can be found to accelerate the model developing processes. To this end, the efficacy of agent-based training on reducing the GPU memory usage and training time is evaluated by two groups of experiments, conducted with the state-of-the-art models on the object detection benchmark of the OPV2V dataset. In group one, all models are trained with all gradients enabled. Group two only enables a limited number $\mathbf{N}_{grad}$ of IAs for gradient calculation. As the example with three IAs shown in Figure 6.6, only $\mathbf{N}_{grad} = 1$ IA has gradients enabled during training. The features of the point clouds from IA 1 and 2 are generated by the shared backbone network with gradient calculation disabled and sent to the IA 0, where all features are then fused and fed to the task head.

#### Model selection

For the above mentioned two groups of experiments, the following four representative state-of-the-art models with different backbones and fusion methods for cooperative object detection are selected to explore how these methods influence the detection accuracy and the training efficiency with reduced gradient calculation:

**AttnFusion** (Xu et al., 2022c) stands out as one of the top-performing models within the OPV2V benchmark. It uses Voxelnet Zhou and Tuzel (2018) as the backbone network to encode point cloud data. This backbone contains several layers of dense 3D convolutions, which are highly GPU memory demanding. The 3D voxel features from the backbone are concatenated at the z-axis to obtain the final BEV feature map with shape $(H,W,C)$ and shared to the ego vehicle. To fuse these feature maps, as implied by its name, AttnFusion incorporates an attention fusion (ref. Section 3.2.2) to consolidate the BEV feature maps from all the neighboring IAs. At last, it uses an anchor-based detection head to generate the bounding box classification and regression result.

**F-Cooper** (Chen et al., 2019a) is the earliest model applied to feature-level cooperative perception. It uses the PointPillar backbone to encode the point clouds into 2D BEV feature maps, and then fuses these maps by a Maxout fusion (ref. Section 3.2.2). Afterwards, an anchor-based detection head is used to predict the objects in the scenario.

**FPVRCNN** (Yuan et al., 2022) is a two-stage detector that requires small communication bandwidth with adequate performance within the OPV2V benchmark. It first extracts the point cloud features with the SpConv backbone network and then uses a spatial-semantic feature aggregation (SSFA) module proposed in CIASSD (Zheng et al., 2021) to enhance the BEV feature encoding for the first-stage detection. Based on these detections, FPVRCNN samples keypoints within the detected bounding boxes and aggregates deep features of different resolutions for each keypoint using the Voxel Set Abstraction (VSA) Module of PVRCNN (Shi et al., 2020); the keypoint features

| Model | $\mathbf{N}_{grad}$ | AP@0.7 | AP@0.5 | AP$^{-}$@0.7 | AP$^{-}$@0.5 |
|---|---|---|---|---|---|
| F-Cooper | all | **82.2** | **89.9** | -13.5 | -6.4 |
| | 2 IAs (1 ego + 1 coop.) | 68.7 | 83.5 | | |
| FPVRCNN | all | 84.0 | 87.3 | +**0.9** | +**0.5** |
| | 1 IA (ego) | **84.9** | **87.8** | | |
| SparseDet | all IAs | **84.1** | **91.1** | -4.6 | -2.0 |
| | 1 IA (ego) | 79.5 | 89.1 | | |
| AttnFusion | all IAs | **87.6** | 92.3 | -0.5 | +**0.3** |
| | 1 IA (ego) | 87.1 | **92.6** | | |

*Table 6.1: Average precision of object detection on OPV2V benchmark with different gradient configurations. AP@iou: average precision at IoU threshold iou. AP$^{-}$@iou: the AP drop of reduced gradient computation in comparison to the full gradient computation.*

and the proposal bounding boxes are then shared and fused in the second stage to obtain refined bounding boxes.

**SparseDet** is a fully sparse and highly efficient model, proposed in this thesis. It uses the MinkUnet backbone to encode point cloud features. To enhance the connectivity between the sparse voxels during the convolutions, it additionally augments the original point clouds with free space points (ref. Section 6.3.2, FSA). In addition, several layers of coordinate-expandable sparse convolutions (CEC, Section 3.1, sparse convolution) are utilized after the MinkUnet to compress the features along the vertical axis and dilate the sparse coordinates. In this way, the features for the final detection can have a better coverage over the centers of visible objects. SparseDet fuses the sparse features from different IAs using naive fusion as introduced in Section 3.2.2. Different from the networks mentioned above, this model uses the center-based detection head for object detection.

### Experiment settings

All models are trained on a single Nvidia RTX3090Ti GPU and an Intel Core i7-8700 CPU for 50 epochs. Batch size is set to 2. All models use Adam optimizer with a starting learning rate of 0.002 and weight decay of $10^{-4}$. The learning rates reduce at epochs 15 and 30 with the multiplication factor of 0.1. During training, the input data are augmented with rotation along the z-axis with a random angle in the range of $r_{\text{aug}} = [-90,90]^{\circ}$, random flipping along the $x$- and $y$-axis, and scaling with a ratio in the range of $s_{\text{aug}} = [0.95,1.05]$. Additionally, the ground truth bounding boxes with fewer than two points are removed during the training process.

### Results and evaluation

The AP values of object detection of the selected models are listed in Table 6.1. In addition to AP at the IoU threshold of 0.5 and 0.7, AP$^{-}$ in the last two columns of Table 6.1 means the performance difference of reduced gradient computation in comparison to the full gradient computation. $\mathbf{N}_{grad}$ is the number of IAs that require gradient calculation during training. $\mathbf{N}_{grad}$ = all means all IAs require gradients. Note that the gradient calculation is always enabled for the ego vehicle as shown in Figure 6.6 ($PCD0$). Mathematically, $N_{grad} \leq 1$, one ego IA and $N_{grad} - 1$ cooperative IAs are enabled for gradient calculation.

Different from the other three models, $\mathbf{N}_{grad}$ of F-Cooper is set to 2 in the second group of experiments. This adjustment was made as $\mathbf{N}_{grad} = 1$ in this case exhibited notably poor performance. In general, F-Cooper performs worse than the other models and is very sensitive to the number of gradient-enabled IAs. This is caused by the Maxout operation, which takes out the maximum values across different IA features, including the gradient-disabled features. Therefore, this operation discards some of the BEV features coming from a gradient-enabled IA and keeps the non-gradient features. In this way, the downstream modules after Maxout are not able to back-propagate all gradients back to the upstream modules that encode point clouds into BEV features.

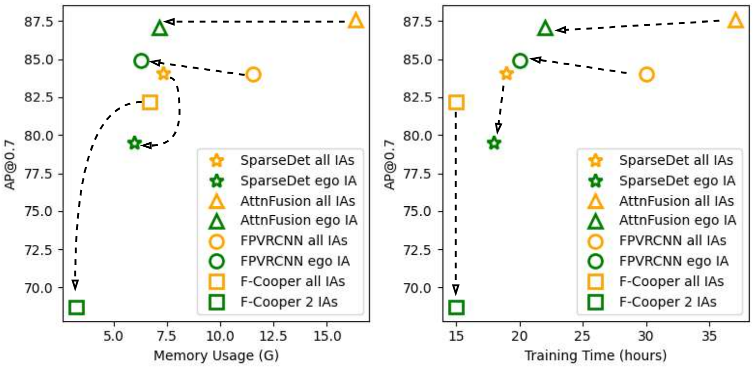


*Figure 6.12: Relation between object detection performance and GPU memory usage (left) as well as training time (right) with different gradient configurations.*

In comparison, SparseDet has significantly smaller performance drops than F-Cooper when using ego-agent-based training. However, its performance is worse than FPVRCNN and AttnFusion. This is because it uses naive fusion (ref. Section 3.2.2) to merge the sparse BEV features from all IAs, which is similar to Maxout that requires no gradient calculation.

Differently, both FPVRCNN and AttnFusion use gradient-enabled modules for fusing the features coming from all IAs. FPVRCNN merges all keypoint features with a RoI grid pooling layer while AttnFusion fuses BEV maps from different IAs by an attention module that attends the features along all IAs. The performance drops of these two models with only the gradient calculation for the ego agent are not noticeable. In some cases, the performance is even enhanced. For example, the APs at the IoU threshold of 0.5 for FPVRCNN and AttnFusion with ego-agent-based training both surpass that with all-agent-based training by a small margin.

From the experimental result shown in Table 6.1, one can conclude that agent-based training may deteriorate the detection performance if the model uses a non-learnable module (Section 3.2.2) that partially drops the upstream features from gradient-enabled IAs. However, if a more effective learnable fusion module is used to merge all upstream features without dropping gradient-enabled features, a more efficient agent-based training strategy (Section 6.4.1) can be used to accelerate the training and the model development process without a noticeable performance drop.

Quantitatively, the detection AP versus the memory usage and training time is illustrated in Figure 6.12. Yellow markers show the results of training with all gradients enabled, and green markers the results of agent-based training. The results of different models are illustrated with different shapes of markers. It is readily evident that agent-based training can significantly reduce the GPU memory usage and the training time, especially for larger models. For example, AttnFusion consumes about 16.7 $G$ GPU memory with gradients enabled for all CAVs, while agent-based training only uses 7.3 $G$, and thus reduces about 56% of GPU memory usage.

The reduced load of gradient calculation directly impacts the training time. This can be identified by the similar patterns of the yellow and the green markers between the left and the right plot in figure 6.12. The reduction ratio in GPU memory usage closely mirrors that of the training time except for F-Cooper. The memory usage of F-Cooper is nearly halved, whereas the training time remains nearly unchanged. This result testifies again to the conclusion drawn from the failure case of F-Cooper, as previously discussed, which is caused by the Maxout operation. The memory usage is reduced because the gradients for non-learnable CAVs are not cached during the forward run. However, for all configurations of $\mathbf{N}_{grad}$, F-Cooper only back-propagates $m \leq h \cdot w \cdot d$ gradients to the upstream modules before Maxout, where $m$ is the number of features with gradients back-propagated and $h$, $w$, $d$ are the height, width, and depth of the BEV feature map of a single IA, respectively. Increasing $\mathbf{N}_{grad}$ for F-Cooper only makes $m$ closer to the maximum value $h \cdot w \cdot d$.

In contrast, gradients of the learnable fusion method such as attention fusion used by AttnFusion are fully propagated back to the learnable IAs. Mathematically, $m = \mathbf{N}_{grad} \cdot h \cdot w \cdot d$ gradients are back-propagated, where $m$ is linearly related to the number of gradient-enabled IAs. Therefore, decreasing $\mathbf{N}_{grad}$ of AttnFusion can linearly reduce the gradient calculation, hence reducing the training time. This effect can hardly be observed by F-Cooper because it only reduces a very small portion of gradients of a single BEV feature map. Similarly, SparseDet only reduces the training time by less than one hour because of the random sampling of the naive fusion module. In addition, the memory reduction of SparseDet by agent-based learning is noticeably less than that of the other three models because it is a fully-sparse model that is already very efficient with respect to the increasing number of IAs.

As a summary, *agent-based training* can significantly reduce the memory usage and training time without noticeable performance drops if an appropriate fusion method without dropping learnable features is used. In this case, the memory usage and training time of dense models is roughly positively proportional to the number of the learnable IA $\mathbf{N}_{grad}$. However, for sparse models, this relationship is non-linear and related to the actual size of the sparse tensors.

# 7 Gaussian-based Evidential BEV Semantic Segmentation (GevBEV)

BEV Semantic Segmentation provides a comprehensive BEV map representation of the 2D dynamic driving space. Previous studies generate the dense BEV semantic map without accounting for whether certain regions are directly observed. Due to occlusions and the sparsity of point cloud measurements, especially in distant areas, these predictions can be unreliable as they often involve extrapolation in unobserved regions. To address these challenges, this section introduces a novel framework that takes the occlusions and point cloud sparsity into consideration and leverages the uncertainty estimation to quantify the reliability of the predictions. Unlike previous methods based on discretized BEV maps, this framework employs evidential deep learning, utilizing Gaussian and Dirichlet distributions to generate BEV semantic segmentation maps in a continuous driving space. This framework is referred to as *GevBEV*. The Gaussian distribution enables parameter-sampling for the Dirichlet distribution in the continuous driving space. The Dirichlet distribution contributes to the uncertainty estimation. This methodology for modeling the *GevBEV* maps is defined in Section 7.1. The overall framework for this purpose is introduced in Section 7.2, including the backbone network for extracting the point cloud features, the fusion method, and the semantic segmentation head that predicts parameters for Gaussian distributions. Based on this framework, extensive experiments are conducted. The experimental details and evaluation metrics are given in Section 7.3 and Section 7.4, respectively. Finally, the experimental results are discussed in Section 7.5.

## 7.1 Definition of GevBEV

Given a point cloud, a neural network is used to extract the point cloud features in the format of a sparse feature map $F \in \mathbb{R}^{N_c \times d_\text{feat}}$ that contains $N_c$ sparse map grid center points $\mathbf{c} = \{\mathbf{c}_i = (x_i, y_i, f_i) | i \in \{1,2,\ldots,N_c\}\}$, each with a feature vector $f_i$ of dimension $d_\text{feat}$ and the coordinate $x_i,y_i$. Figure 7.1 illustrates an example feature map with $N_c = 3$ grid center points (pink cells). This thesis proposes Gaussian-based Evidential BEV Semantic Segmentation, *GevBEV* in short, that assigns each grid center point a Gaussian distribution (colored rings in Figure 7.1). The probability masses of each center point $\mathbf{c}_i$ represent the possibility that the surrounding space belongs to the same class as $\mathbf{c}_i$. To learn such distributions, two regression heads composed of linear layers are employed to predict the parameters of the Gaussian distribution, one head $\mathcal{H}_\text{evi}$ for generating the classification evidence of the center points and one $\mathcal{H}_{var}$ for the variance of the Gaussian distributions. More specifically, these two heads project the input feature dimension $d_\text{feat}$ into the dimension $d_\text{evi} = 2$ (fore- and background evidence) and $d_\text{var} = 4$ (variances along $x$- and $y$-axis for both fore- and background), respectively. Both heads are activated with ReLU to obtain positive values. Note that this configuration is designed for a binary classification problem, namely for the fore- and background classification. As illustrated on the right side of Figure 7.1, this thesis uses separated binary classifications for each of the two semantic classes: road surface and vehicle. More specifically, one binary classification with two sub-heads $(\mathcal{H}_\text{evi},\mathcal{H}_\text{var})$ for the road surface (foreground) and the non-road surface (background) classification, one for vehicle (foreground) and non-vehicle (background) classification. For simplicity, the definition of one binary classification between the fore- and background class is introduced in the following.

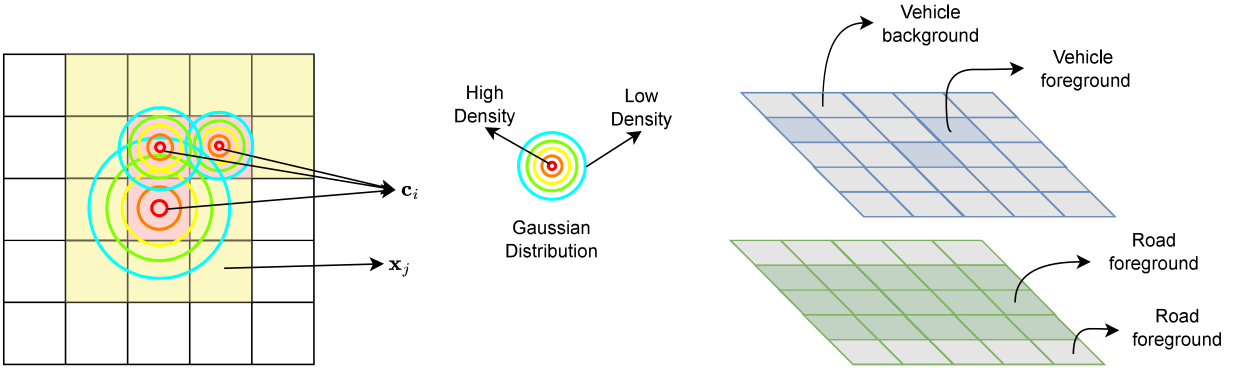


*Figure 7.1: Left: Gaussian-based Evidential BEV Semantic Segmentation. Red to cyan rings represent Gaussian density from high to low. Right: BEV maps of the binary classification for vehicle and road, respectively.*

The outputs of the heads $\mathcal{H}_{\text{evi}}$,$\mathcal{H}_{\text{var}}$ are noted as

$$\mathbf{o}_{\text{evi}} = [o_{\text{evi}}^{\text{fg}}, o_{\text{evi}}^{\text{bg}}], \tag{7.1}$$

$$\mathbf{o}_{\text{var}} = [o_{\sigma x}^{\text{fg}}, o_{\sigma y}^{\text{fg}}, o_{\sigma x}^{\text{bg}}, o_{\sigma y}^{\text{bg}}], \tag{7.2}$$

where fg indicates the foreground and bg the background. $\mathbf{o}_{\text{evi}}$ is regarded as the evidence of the grid center point to be foreground or background. $\mathbf{o}_{\text{var}}$ are the regressed variances of the point in $x$- and $y$-axis. To ensure that each point is contributing, a small initial variance $\sigma_0$ is added to the predictions, resulting in the variances $\sigma_{x,y}^2 = \mathbf{o}_{\text{var}} + \sigma_0^2$. For any new given target point $\mathbf{x}_j$ (yellow cells in Figure 7.1) in the neighborhood of the center point $\mathbf{c}_i$ in the BEV space, the probability density $\phi(\mathbf{x}_j)_i$ of this new point belonging to a specific class is drawn by Eq (7.5),

$$\Sigma_i = \begin{bmatrix} \sigma_x^2, 0 \\ 0, \sigma_y^2 \end{bmatrix}, \tag{7.3}$$

$$m_{ji} = (\mathbf{x}_j - \mathbf{c}_i)^T \Sigma_i^{-1} (\mathbf{x}_j - \mathbf{c}_i), \tag{7.4}$$

$$\phi(\mathbf{x}_j)_i = \frac{\exp\left(-0.5 \cdot m_{ji}\right)}{\sqrt{2\pi^d |\Sigma_i|}}, \tag{7.5}$$

where $\Sigma_i$ is the covariance of the center point $\mathbf{c}_i$ for foreground or background distribution, $m_{ji}$ is the squared Mahalanobis distance of point $\mathbf{x}_j$ to the center point $\mathbf{c}_i$, and $d = 2$ is the dimension of the distribution.

The overall Dirichlet evidence $e(\mathbf{x}_j)$ for point $\mathbf{x}_j$ is the sum of the normalized and weighted probability mass drawn from all the neighboring center points $\text{nbr}(j)$ that are in the maximum distribution range $\nu$. It reads as

$$\begin{aligned} e_k(\mathbf{x}_j) &= \sum_{i \in \text{nbr}(j)} \frac{\phi(\mathbf{x}_j)_i}{\phi(\mathbf{c}_i)} \cdot o_{\text{evi}}^k, \\ &= -\frac{1}{2} \sum_{i \in \text{nbr}(j)} m_{ji} \cdot o_{\text{evi}}^k, \end{aligned} \tag{7.6}$$

where $\phi(\mathbf{c}_i)$ is the probability density at the center point, and $k \in \{\text{fg},\text{bg}\}$. Hereafter, Equation (7.6) is derived as following

$$\begin{aligned}
\log(\phi(\mathbf{x}_j)_i) &= -\frac{1}{2}m_{ji} - \frac{1}{2}\log(2\pi^d|\Sigma|), \\
&= -\frac{1}{2}(d\log(2\pi) - \log|\Sigma|) - \frac{1}{2}m_{ji}, \\
\log(\phi(\mathbf{c}_i)) &= -\frac{1}{2}m_i - \frac{1}{2}\log(2\pi^d|\Sigma|), \\
&= -\frac{1}{2}(d\log(2\pi) - \log|\Sigma|), \\
\frac{\phi(\mathbf{x}_j)_i}{\phi(\mathbf{c}_i)} &= \exp(\log\frac{\phi(\mathbf{x}_j)_i}{\phi(\mathbf{c}_i)}), \\
&= \exp(\log\phi(\mathbf{x}_j)_i - \log\phi(\mathbf{c}_i)), \\
&= -\frac{1}{2}m_{ji},
\end{aligned} \tag{7.7}$$

where the squared Mahalanobis distance of the center point $\mathbf{c}_i$ to itself is $m_i = 0$. Following Sensoy et al. (2018), the expected probability $p_{j,k}$ – point $\mathbf{x}_j$ belonging to class k – and the uncertainty $u_j$ of this classification result are

$$\hat{p}_{j,k} = \frac{\alpha_{j,k}}{S_j} = \frac{e_k(\mathbf{x}_j) + 1}{\sum_{k\in\{\text{fg,bg}\}}(e_k(\mathbf{x}_j) + 1)}, \tag{7.8}$$

$$u_j = \frac{K}{S_j}, \tag{7.9}$$

where $\alpha_{j,k}$ is the concentration parameter of class $k$ for $k = 1,...,K$, and $S_j$ the strength of the Dirichlet distribution of point $\mathbf{x}_j$.

## 7.2 Architecture of the Framework GevBEV

### 7.2.1 Overview

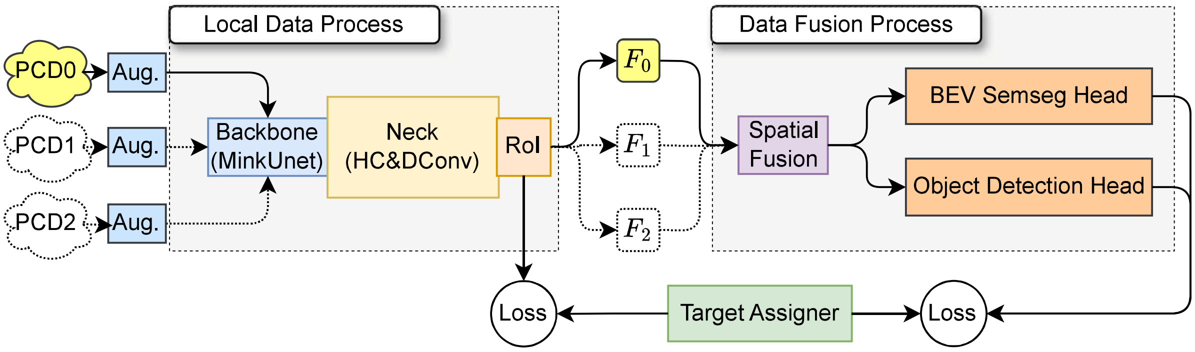


*Figure 7.2: Overall model architecture of BEV Semseg*

Adapting the framework introduced in the last chapter (Figure 6.6), the modules used in each processing stage of *GevBEV* are illustrated in Figure 7.2. The input point clouds are first augmented with free space points and geometric transformation (ref. Section 6.3.2). Then the *MinkUnet* backbone network (ref. Section 3.2.1) is utilized to extract the point cloud multi-resolution 3D spatial features, including $\tilde{P}^{sem}$ for BEV semantic segmentation and $\tilde{P}^{det}$ for object detection. These 3D

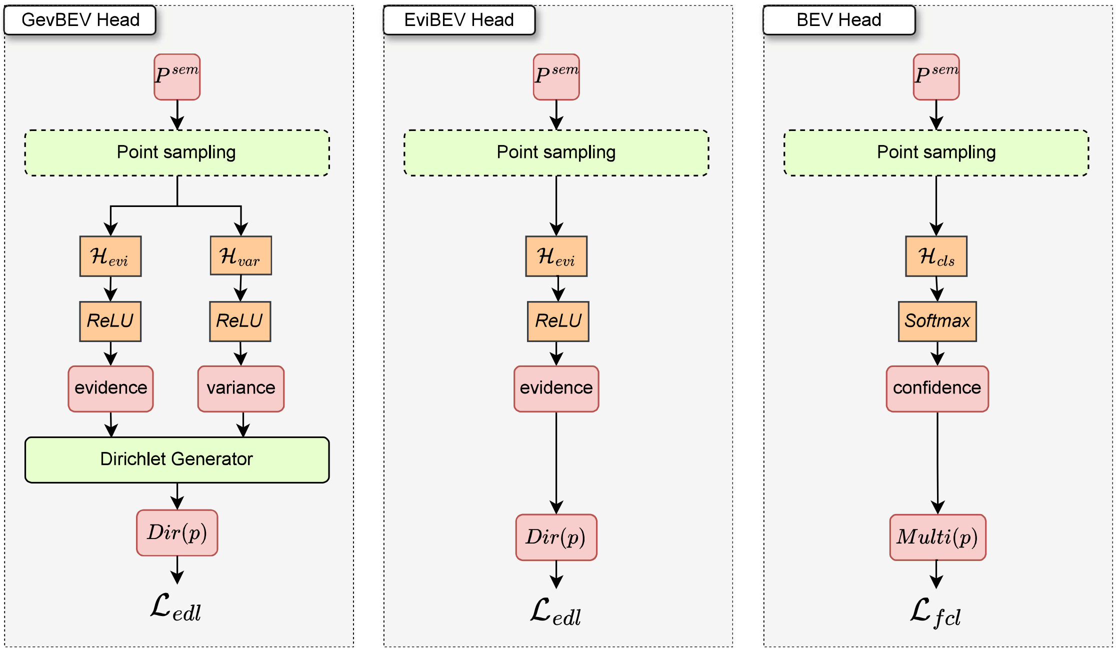


*Figure 7.3: Comparison of three BEV semantic segmentation heads: the GevBEV Head (proposed in this work), the EviBEV Head (a modified version without the Gaussian distribution), and the BEV Head (a simplified version without both Gaussian and Dirichlet distributions).*

features are fed to the *Neck* which contains two modules; the *Height Compression (HC)* module compresses the 3D spatial features along the z-axis to obtain the BEV feature maps and the *Dilated Convolution (DConv)* module is used to expand the spatial coverage of the BEV features and increase the connectivity between the discrete observed feature points. The RoI module is used for selecting the most important BEV points for sharing. More details about this head can be found in Section 7.2.3. By receiving the shared BEV points features from cooperative IAs, the ego-IA fuses these features using the *Spatial Fusion* module as described in Section 7.2.2. Based on these fused features, the *BEV Semseg head* generates the final BEV semantic segmentation maps using the method defined in Section 7.1. The confidences generated in the BEV Semseg head can be utilized as an indicator for verifying the overall confidence of the objects predicted in the object detection head (Figure 7.10). Therefore, an additional object detection head is used in this pipeline as shown with the orange boxes in Figure 7.2.

### 7.2.2 Spatial fusion

As illustrated in Figure 7.2, $F^i$ is the center feature map of the $i$-th IA. The number of center points in the feature map is notated as $N_c^i$. Given the ego center feature map $F^0 \in \mathbb{R}^{N_c^0 \times d_{\text{feat}}}$ and the center point feature maps $\text{CPM}_{\text{feat}} = \{F^i | F^i \in \mathbb{R}^{N_c^i \times d_{\text{feat}}}, 1 \le i < N\}$ received from the $N-1$ cooperative IAs, the *Spatial Fusion* module merges all center feature maps $\mathbf{F} = \{F^i | F^i \in \mathbb{R}^{N_c^i \times d_{\text{feat}}}, 0 \le i < N\}$ with the attention fusion module which is introduced in Section 3.2.2. Then the fused features are fed to the BEV Semseg head to generate the final BEV semantic segmentation result.

### 7.2.3 Heads

In addition to the proposed novel GevBEV Semseg head for *GevBEV*, two additional heads, *EviBEV* and the conventional *BEV* head, are also used in the experiments for comparison. They are illustrated in Figure 7.3. During training, the input BEV features $P^{sem}$ are randomly down-sampled to half of their original amount in order to accelerate the training process. During the

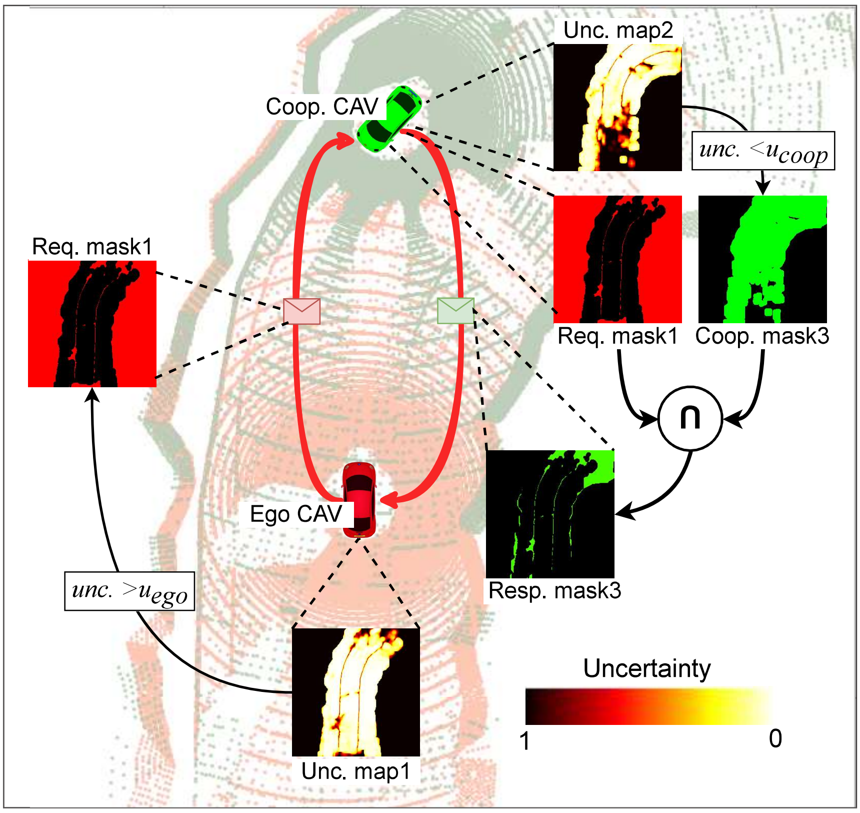


*Figure 7.4: An example of the information shared between CAVs. The ego CAV and its request mask with high uncertainty areas indicated in red color, and the cooperative CAV and its response mask with low uncertainty indicated in green color. In the uncertainty maps, light color indicates low uncertainty and black color indicates no measurements.*

inference stage, all input features are used. In the *GevBEV* head option, a regression head is used to generate the evidence and the variance values for the *Dirichlet Generator* as described in the left sub-figure of Figure 7.3. Using Equation (7.6), the Dirichlet Generator takes the evidences and variances of all center points as input and generates the parameters for the Dirichlet distributions of the new sample points in the continuous driving space. The predicted distributions and the ground truth semantic labels are then used for calculating the evidential loss $\mathcal{L}_{\text{edl}}$ (ref. Section 3.2.4). For comparison purposes, the *EviBEV* and BEV head are defined by gradually removing essential parts of the *GevBEV* head. *EviBEV* head removes the Gaussian-based variances and directly regresses the evidences as the final parameters of the Dirichlet distribution. It uses the same loss function as the *GevBEV* head. Differently, the *BEV* head option uses the softmax activation function to directly generate classification confidences that relate to the multinomial distribution $Multi(p)$. Correspondingly, the focal loss $\mathcal{L}_{\text{fcl}}$ is used for this head.

As described in Section 7.1, two separate Semseg heads are used for classifying the road surface and vehicle objects. The $\mathcal{H}^{\text{road}}$ head distinguishes between the road surface (foreground) and non-road surface (background). The $\mathcal{H}^{\text{obj}}$ head distinguishes between the class vehicle (foreground) and non-vehicle (background). Note that each head can be one of the three implementations: *GevBEV* head that contains two sub-heads $\mathcal{H}_{\text{evi}}$,$\mathcal{H}_{\text{var}}$, the *EviBEV* head that only contains the evidence regression sub-head $\mathcal{H}_{\text{evi}}$, and the conventional *BEV* head with the $\mathcal{H}_{\text{cls}}$ sub-head for generating parameters of a multi-nominal distribution.

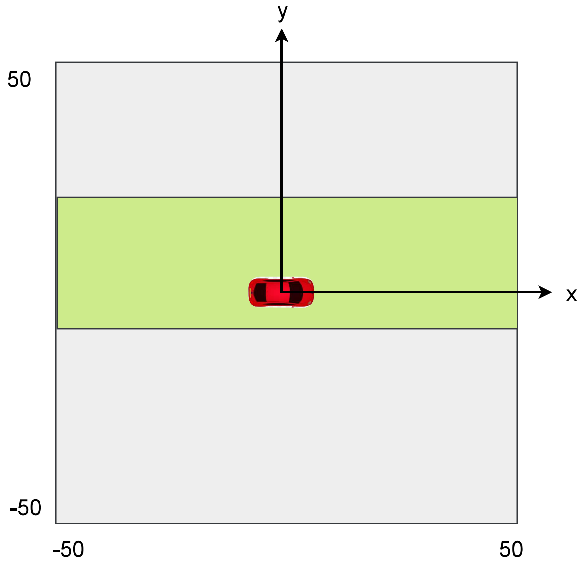


*Figure 7.5: Areas considered for selecting information for CPMs. Grey area ($CPM^{all}$): overall area in the perception range (e.g., $[-50,50]m$). Green area ($CPM^{road}$): drivable road area.*

### 7.2.4 CPM selection

When the communication bandwidth is very limited or the channels are very busy, an uncertainty-based information selection strategy is introduced to reduce the consumption of communication bandwidth. Instead of simply sharing all the information among IAs, the most important information is distilled by a *EviBEV* Head in the RoI module that generates the uncertainty map (Equations (7.8) and (7.9)) as a weight map for CPM selection. Note that *GevBEV* is not used for this purpose because in this stage, only the discretized predictions $\mathbf{o}_{\text{evi}}$ and $\mathbf{o}_{\text{var}}$ should be selected. The *GevBEV* Head, which is used to generate more fine-grained final results, is used after the data sharing and fusion.

As the example with two CAVs shown in Figure 7.4, the red ego CAV first generates a request binary mask (red *Req. mask1*) by thresholding its uncertainty map (*Unc. map1*) with $unc. > u_{\text{ego}}$ and only sends the request for the perception information in the areas where there are high uncertainty (red area of *Req. mask1*). Then, the green cooperative CAV responds with its own masked feature or evidence map (green *Resp. mask3*). This response mask contains the complementary areas where the ego CAV has high uncertainty and the cooperative CAV has low uncertainty for the BEV semantic segmentation. It is generated by thresholding the uncertainty map of the cooperative CAV with the threshold $u_{\text{coop}}$ (*Coop. mask3*) and intersecting with the received request mask (*Req. mask1*) from the ego CAV.

Afterwards, this resulting response mask is used to select the information for the CPM communicated from the cooperative CAV to the ego CAV by only selecting the evidences in the response-masked areas. Because the evidence only contains the two values for fore- and background class at each point, it consumes very few communication resources.

Figure 7.5 shows two different area-based sharing strategies that take either the whole area in the perception range (grey) of the ego vehicle or the drivable road area (green) into consideration for the information selection. The first option for sharing feature maps is then notated as $\text{CPM}^{\text{all}}$. The second option is introduced because the CAVs pay more attention to the situation on the road surface; only sharing information in this area can further reduce the CPM size. In real applications, the road geometric information can be retrieved from prior information, such as maps. The co-perception benchmark, such as OPV2V (Xu et al., 2022c), also provides a HD map acquired beforehand. As a proof-of-concept study, the scene in each frame is registered into the HD map to further eliminate non-surface areas in the masked areas. This sharing strategy is notated as $\text{CPM}^{\text{road}}$ when the extra HD map is already provided to the CAVs. For simplicity of the CPM size evaluation, the uncertainty threshold is fixed to $u_{\text{coop}} = 1.0$ for the cooperative CAVs and the threshold $u_{\text{ego}}$ for the ego CAV in the experiments is varied to evaluate its effectiveness.

## 7.3 Experiments

### 7.3.1 State-of-the-art models for comparison

To evaluate the effectiveness of the proposed model *GevBEV*, it is compared to the state-of-the-art co-perception models on the co-perception benchmarks OPV2V and V2V4Real in both simulated and real driving scenarios. In addition to the aforementioned *FCooper* (Chen et al., 2019a), *AttnFuse* (Xu et al., 2022c), the following models that provide open source implementations are also used for comparison.

- *V2VNet* (Wang et al., 2020a) utilizes a spatially aware graph neural network to aggregate the feature maps received from all the nearby CAVs.
- *DiscoNet* (Li et al., 2021) proposes a distilled collaboration graph model for multi-agent perception. A teacher-student framework is leveraged to facilitate collaboration among the agents, and a matrix-valued edge weight is used to guide inter-agent attention to more informative regions.
- *V2XViT* (Xu et al., 2022b) proposed heterogeneous multi-agent self-attention to update feature maps of different types of agents and multi-scale window attention to fuse the features of all perception agents.
- *CoBEVT* (Xu et al., 2022a) builds a transformer-based framework to fuse feature maps for co-perception. It proposes a fused axial attention module to capture sparsely local and global spatial interactions across multi-views and agents.

Although the above-mentioned models, except CoBEVT, are originally developed based on point-cloud data, their BEV feature map processing and fusion strategies can be shared with the camera-based pipelines. More specifically, the image features are projected to the BEV feature map using the *Cross-View Transformer* (Zhou and Krähenbühl, 2022) (CVT) following CoBEVT. Then these feature maps are processed using different models for comparison purposes.

Unlike *GevBEV*, none of these models listed above provide uncertainty estimation for the BEV Semseg results and select the essential information communicated to the ego CAV. Hence, those models are only compared with *GevBEV* on the BEV Semseg performance. They are not further compared with *GevBEV* in terms of CPM size for the V2V communication in the co-perception application.

### 7.3.2 Ablation studies

A series of ablation studies is conducted to analyze the efficacy of the proposed modules of *GevBEV*.

- *BEV* is the proposed model with the point-based spatial Gaussian and the evidential loss $\mathcal{L}_{edl}$ removed, turning the *GevBEV* model from a probabilistic model into a deterministic model. It uses cross-entropy to train the corresponding heads to classify points of the BEV maps. This model is treated as the baseline model.
- *EviBEV* only has the point-based spatial Gaussian removed. It still uses $\mathcal{L}_{edl}$ to train the distribution head.

– *GevBEV⁻* uses the same model architecture and loss function as *GevBEV* but is trained without free space augmentation (Section 6.3.2).

### 7.3.3 Implementation details

In all experiments, the input voxel size is set to 0.2 m to balance between computational overhead and performance. The free space points (ref. Section 6.3.2 and Figure 6.7) are sampled with the maximum height of $z \leq h_{fs} = -1.5\,m$, the limited distance $d_{fs} = 7.5\,m$, and the sampling step of $s_{fs} = 1.5\,m$ on OPV2V. However, $d_{fs}$ is set to $9m$ for the V2V4Real dataset because it has a longer detection range in $x$-direction (longitudinal direction). During the geometric augmentation, the point cloud coordinates are scaled randomly with the ratio in the range of [0.95,1.05] to make the model learning from more diverse object sizes. Then a normally distributed $\mathcal{N}(0,0.2)\,m$ noise is added to the point clouds to increase the model robustness against small measurement errors.

The whole network is trained from scratch for 50 epochs with weight decay of 0.01 using the Adam optimizer Kingma and Ba (2015) parameterized with $\beta = [0.95,0.999]$ and $\gamma = 0.1$. A multi-step learning-rate scheduler is used with a starting learning rate of $lr = 10^{-3}$ and two times reduction at epoch 20 and 45, respectively. The learning rate decreases at each reduction by a factor of 0.1.

The BEV learning target sample points are generated with *continuous BEV map Assigner* as described in Section 6.3.7. For the static BEV Semseg head, each center point generates $n_{\text{tgt}} = 10$ samples so that they are able to cover all observed areas with a similar density (distant areas are sparse, requires more sample points). For the dynamic BEV Semseg head, $n_{\text{tgt}} = 1$ point is generated as all points inside the object and its buffer region are all kept (ref. Figure 6.11), they are dense enough for learning the details of the object edges. The resolution of the cells for down-sampling of the static head is set to 0.4 m, and $N_{\text{tgt}}$ is set to 3000. For the dynamic head, the width of the buffering area is set to $d_{\text{bf}} = 4\,\text{m}$. From the background, $N_{\text{tgt}} = 50 \cdot n_{\text{gt}}$ points are sampled as negative samples, where $n_{\text{gt}}$ is the number of the ground truth bounding boxes. For these sampled target points, the evidences are drawn by Equation (7.6) in a maximum distribution range of $\nu = 2\,m$. These parameters are all set empirically so that the generated samples are sufficient enough to learn in the continuous space with an affordable computational budget.

## 7.4 Evaluation Metrics

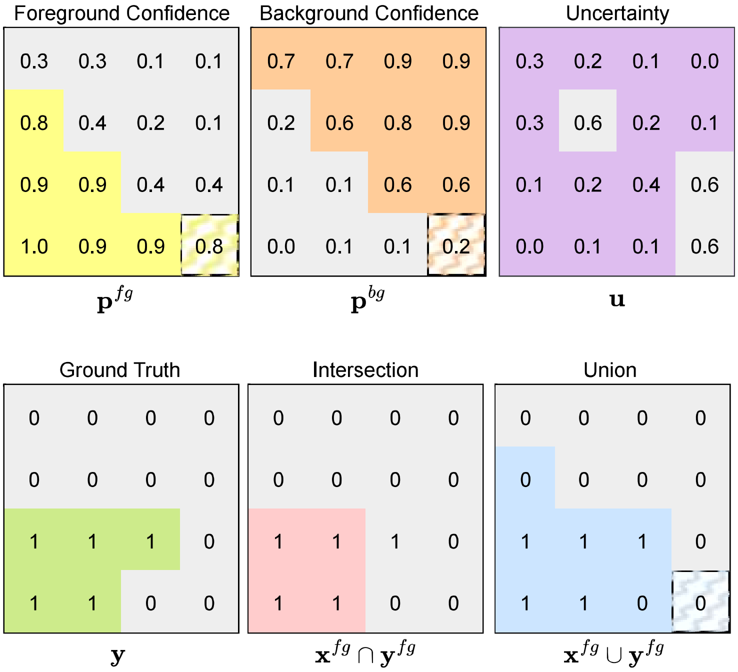
### 7.4.1 IoU

The conventional IoU metric for semantic segmentation (Section 3.4.2) does not consider the uncertainty for evaluation. It only calculates one IoU as the overall performance metric. Considering the uncertainty, this thesis modifies this metric by calculating IoUs under different uncertainty thresholds $u_{thr}$. The modified version is described with Algorithm 6. For a given uncertainty threshold $u_{thr}$, the IoU is computed over the pixels that have uncertainties under the threshold $u_{thr}$ (Algorithm 6, Line 1). As the example illustrated in Figure 7.6, at the uncertainty threshold $u_{thr} = 0.5$, only the purple pixels are considered for the IoU calculation. As a result, the bottom right pixel (colored with yellow zigzag lines) is not contained in $\mathbf{x}^{fg}$, hence ignored during the evaluation. When the threshold is $u_{thr} = 1$, the modified metric equals the original version as all pixels are considered for the IoU calculation. Note that different thresholds are used for generating the calibration plot (Section 3.4.3).

Although the proposed model GevBEV can generate BEV maps of any resolution, only the predicted BEV maps of resolution $0.4\,m$ are evaluated so that it can be compared to other models that only generate the BEV maps of this resolution. Besides, the GevBEV model only generates the results over the observed areas that are covered by the point cloud. The IoUs calculated only on these areas are notated as $IoU_{\text{obs}}$. This is in line with the concept that the prediction is reliable

**Algorithm 6** Uncertainty-based BEV Semseg IoU

```
Ensure: $\mathbf{p} = \{(p_i^{fg}, p_i^{bg}) | i \in 0,1,\ldots,N\}$: predicted confidence map. $\mathbf{u} = \{(u_i | y_i \in \{0,1\}, i \in 0,1,\ldots,N\}$: uncertainty map. $\mathbf{y} = \{(y_i | y_i \in \{0,1\}, i \in 0,1,\ldots,N\}$: ground-truth labels. $N$: the number of pixels in the sample BEV map.
$\mathbf{x}^{fg} = \{x_i | p_i^{fg} > p_i^{bg}, u_i < u_{thr}, i \in \{0,1,\ldots,N\}\}$
$\mathbf{y}^{fg} = \{y_i | y_i = 1, i \in \{0,1,\ldots,N\}\}$
$IoU = |\mathbf{x}^{fg} \cap \mathbf{y}^{fg}| / |\mathbf{x}^{fg} \cup \mathbf{y}^{fg}|$
return $IoU$
```



Foreground Confidence

| | | | |
|---|---|---|---|
| 0.3 | 0.3 | 0.1 | 0.1 |
| 0.8 | 0.4 | 0.2 | 0.1 |
| 0.9 | 0.9 | 0.4 | 0.4 |
| 1.0 | 0.9 | 0.9 | 0.8 |

$\mathbf{p}^{fg}$

Background Confidence

| | | | |
|---|---|---|---|
| 0.7 | 0.7 | 0.9 | 0.9 |
| 0.2 | 0.6 | 0.8 | 0.9 |
| 0.1 | 0.1 | 0.6 | 0.6 |
| 0.0 | 0.1 | 0.1 | 0.2 |

$\mathbf{p}^{bg}$

Uncertainty

| | | | |
|---|---|---|---|
| 0.3 | 0.2 | 0.1 | 0.0 |
| 0.3 | 0.6 | 0.2 | 0.1 |
| 0.1 | 0.2 | 0.4 | 0.6 |
| 0.0 | 0.1 | 0.1 | 0.6 |

$\mathbf{u}$

Ground Truth

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |

$\mathbf{y}$

Intersection

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |

$\mathbf{x}^{fg} \cap \mathbf{y}^{fg}$

Union

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |

$\mathbf{x}^{fg} \cup \mathbf{y}^{fg}$

*Figure 7.6: An example of BEV semantic segmentation result for IoU calculation under uncertainty.*

only when it is conducted over the observed areas. However, in order to compare the proposed models with the state-of-the-art models, the IoU ($IoU_{all}$) over all pixels in the perception range, including the non-observed areas, is also calculated for GevBEV with all points in non-observable areas classified as background. Mathematically, a point $x_i$ is observable if $\exists j \in \{j| \parallel x_i - x_j \parallel_2 < \nu\}$, meaning a point is observable if it is in the range $\nu = 2\,m$ of any Gaussian distributed center points. $\nu$ is the maximum distance of the Gaussian distribution tile that is considered for generating the evidence of the Dirichlet distribution.

### 7.4.2 Calibration plot

The proposed GevBEV generates uncertainty values for the classification of each point, namely one uncertainty value for both fore- and background class. However, the unbalanced number of fore- and background samples will lead to a biased evaluation. For example, the classification of a point can have a high uncertainty due to the lack of evidence from its neighbors, while it may end up with high classification accuracy because its neighbors belong to a dominant class. To avoid this biased evaluation, each sample is weighted by the ratio of the total number of samples in that particular class. Consequently, the average classification accuracy $ac$ for each uncertainty interval of the calibration plot described in Section 3.4.3 is replaced by the weighted $ac$:

$$wac = ac \cdot \frac{N^{fg}}{N} \tag{7.10}$$

where $N^{fg}$ is the number of foreground pixels and $N$ is the total number of sample pixels. Similarly, the Calibration Error (CE, Equation (3.27)) is also calculated with weighted accuracy $wac$ instead of $ac$.

### 7.4.3 Notations

To avoid confusion in the evaluation, some notations should be defined. In all experiments, the classification results of the two heads $\mathcal{H}^{\text{road}}$ and $\mathcal{H}^{\text{obj}}$ are evaluated. The proposed *GevBEV* only operates on the features of the sparse center points (Figure 7.1, pink cells), and generates classification results only for the observed areas, including the center points and the points in their neighborhood (Figure 7.1, yellow cells). This thesis proposes to evaluate only on the observed areas because only these areas reflect the model performance. This evaluation configuration is notated as $IoU^{\text{sparse}}$. However, the state-of-the-art models operate on dense feature maps and generate the perception result in all areas of the perception range (gray area in Figure 7.5), including unobserved areas (Figure 7.1, white cells). More specifically, all pixels on the dense semantic segmentation map in this range are taken into account in the IoU calculation. This IoU result is notated as $IoU^{\text{dense}}$. In order to be able to compare the proposed methods with these models, all pixels in the unobserved area (Figure 7.1, white cells) are regarded as the background class for both $\mathcal{H}^{\text{road}}$ and $\mathcal{H}^{\text{obj}}$ head.

In addition, two CPM selection strategies are introduced in Section 7.2.4. The superscript *all* is used for the results (IoU and CPM size) of the strategy that selects the CPM information over all areas in the perception range, and the superscript *road* for the selection over the road area.

## 7.5 Results

### 7.5.1 Qualitative analysis

#### BEV semantic segmentation maps

With the proposed probabilistic model, the BEV Semseg maps of an example driving scenario generated by *GevBEV* are visualized in Figure 7.7. From left to right, the three subfigures in the first row show the results of uncertainty, classification confidence, and the ground truth of the road surface. In the uncertainty map, a lighter color indicates lower uncertainty, whereas black areas are regarded as non-observable. Correspondingly, the confidence map shows the confidence score for the road surface (green) and the non-road surface (blue). The bottom subfigure shows the detailed detection results of both road surface and objects overlaid in one figure. Only points classified as roads with uncertainty under the threshold of 0.7 are highlighted in light color in the bottom layer to show the situation of the drivable area. The predicted and corresponding ground truth bounding boxes are plotted in red and green, respectively. Moreover, the bounding boxes are filled with the vehicle confidences drawn from the Dirichlet distribution of the object head, where magenta points are associated with high confidence, while cyan the opposite.

As shown in Figure 7.7, the BEV maps generated by *GevBEV* give a holistic report of the observation status in all areas of the perception range. The following three observation statuses can be described as:

1. area observed with high-certain perception results, *e.g.,* area where the road and vehicle classification uncertainty is smaller than 0.5.

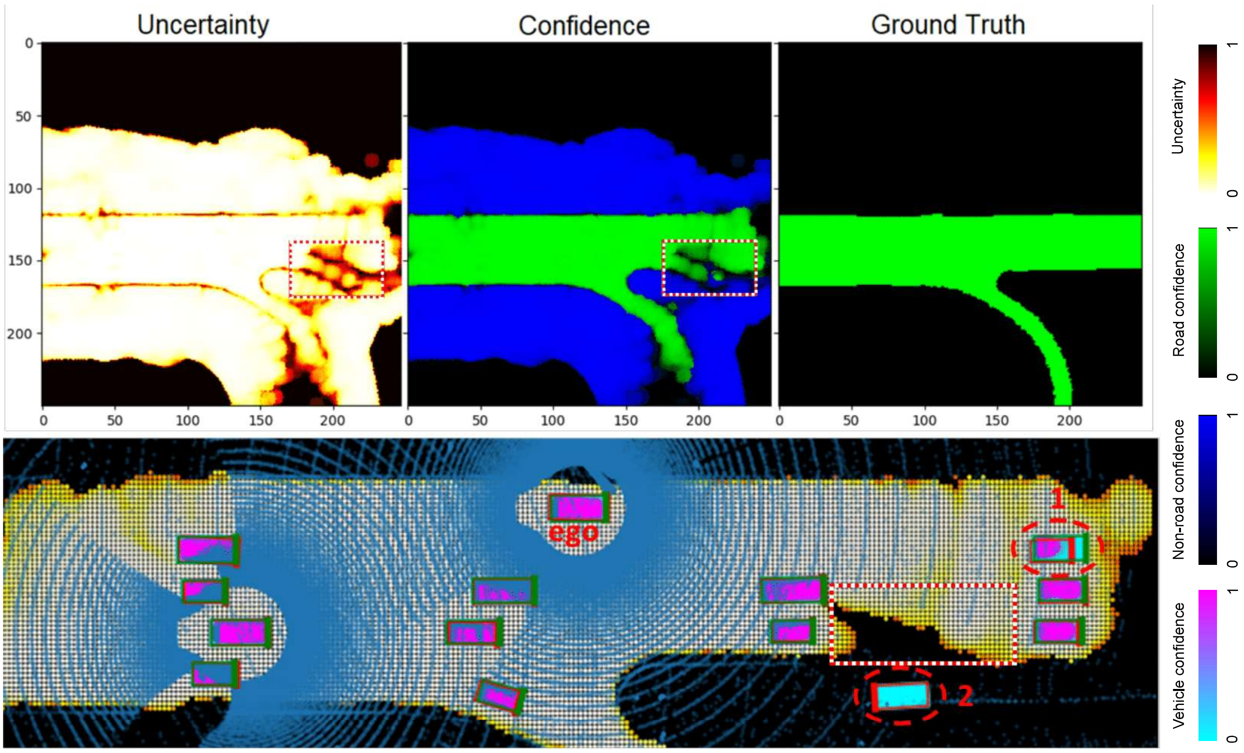


*Figure 7.7: Results of the GevBEV. The first row show the semantic segmentation result. The bottom sub-figure shows more detailed detection results in a larger extent. Specifically, the original input point cloud is denoted by blue points, road points are highlighted by a light color if their uncertainties are under the threshold of 0.7, and the objects are shown in bounding boxes with red color indicating the detection, and with green color the corresponding ground truth, respectively. The thick bar in the front of the bounding boxes denotes the driving direction. Moreover, the bounding boxes are filled with the point confidences generated by dynamic (object) head of GevBEV, where magenta points are associated with high confidence, while cyan indicates the opposite.*

2. area observed but with insufficient perception certainty, *e.g.,* area where the road or vehicle classification uncertainty larger less than 0.5.

3. area not observed, *e.g.,* area where uncertainty is 1.

This observation report provides a reliable information source to support IAs for decision-making. It is more reliable because it can be sourced back to the original measurement points. For example, the IA can usually generate safe driving based on status 1. However, due to occlusions and long detection distance, the ego CAV is not certain about the area marked with a dashed line red rectangle (status 2 or 3). If the ego CAV needs to drive into this uncertain area, it should either request the missing information from other CAVs or slow down waiting for the clearance of the uncertain area. Moreover, the BEV dynamic (object) head generates evidence of areas that could be occupied by vehicles, indicated by the confidence values. Therefore, they are reliable and explainable clues for validating the bounding-box detection. For example, as shown in the bottom sub-figure, most predicted boxes are well aligned with the ground truth except those two marked with red ellipses. The vehicle in ellipse 1 is only partially observed; there is no enough evidence to support this detection. Similarly, there is almost no evidence for false positive detection in ellipse 2. Therefore, this detection can be simply removed.

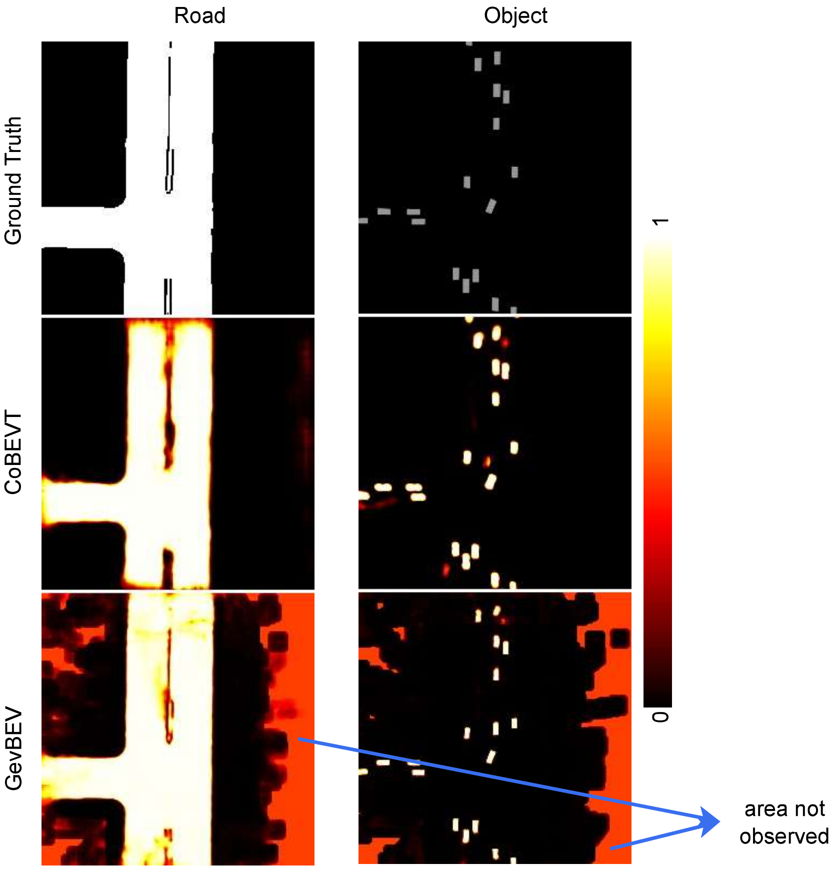


*Figure 7.8: The comparison of classification confidences between GevBEV and CoBEVT on the OPV2V dataset.*

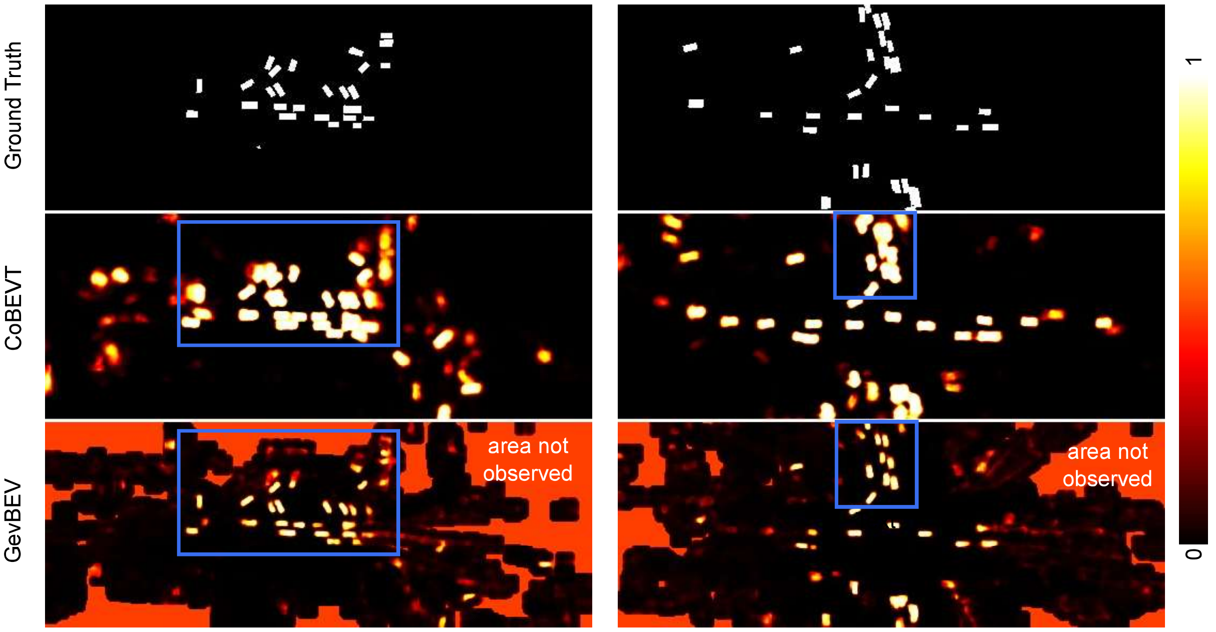


*Figure 7.9: The comparison of classification confidences between GevBEV and CoBEVT on the V2VReal dataset.*

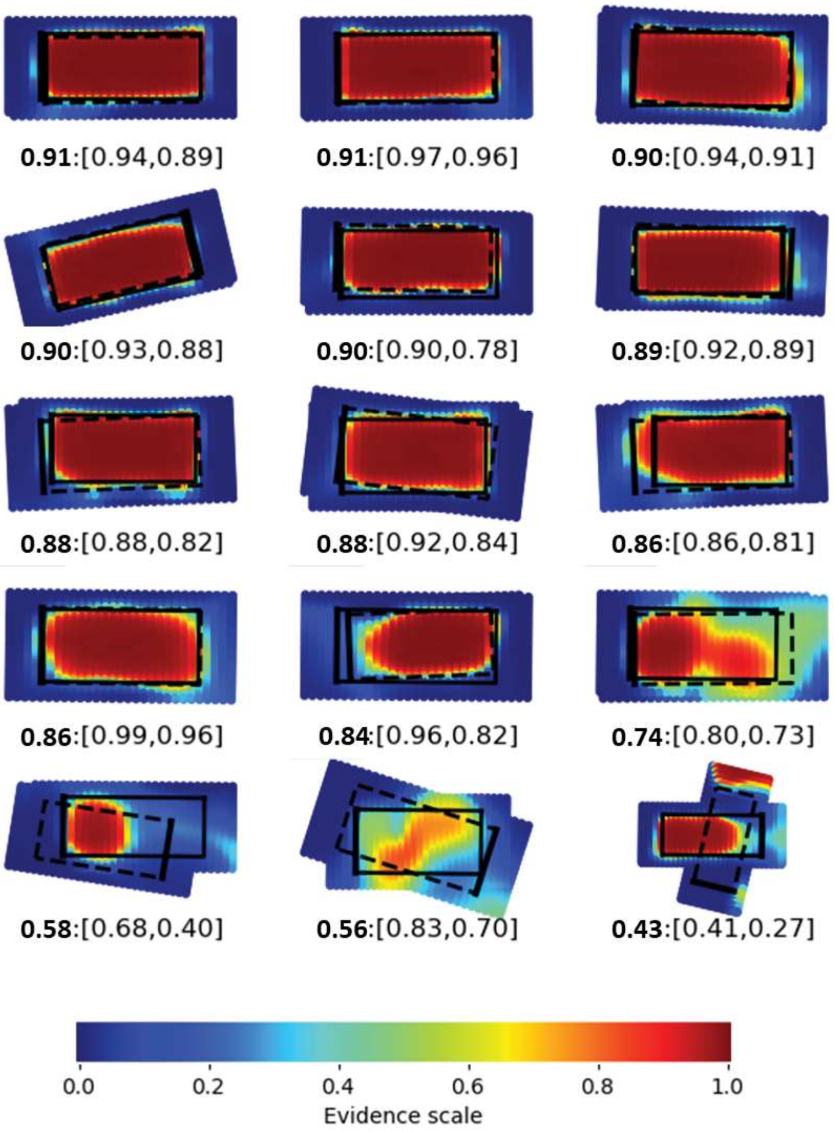


*Figure 7.10: Object point distribution. The dashed line bounding boxes denote the detection, and the solid line bounding boxes denote the corresponding ground truth. The thick bar of each bounding box denotes the driving direction. The statistics under each sub-figure denote the average evidence and [JIoU, IoU].*

### Comparison to baseline

Figures 7.8 and 7.9 illustrate the classification confidences, with the color scale transitioning from dark to light, representing confidence values from 0 to 1. In the right column of fig. 7.8 and in the bottom row of Figure 7.9, the red color indicates that the proposed *GevBEV* retains information on unobserved areas. In the absence of observations, *GevBEV* exhibits a tendency to produce more refined details for both the road and vehicle edges. Notably in Figure 7.9, CoBEVT, the best performing model among the selected comparative models, tends to generate more false positive predictions, and in some cases, vehicles even appear merged together (Figure 7.9, middle row blue box). This is because it only learns from discrete pixel samples, which are not accurate enough to learn the fine-grained details. In contrast, *GevBEV* accurately separates all vehicles (Figure 7.9, bottom row blue box), thanks to its precise edge description learned from the samples generated in the continuous space.

### Evidence of object point distribution

The average evidence score of the detected bounding boxes can be used to show the relations between the quality of the detection and the distribution of observation points of the corresponding object. Inspired by Wang et al. (2020b), the generated uncertainty is also leveraged to calculate the JIoU in addition to the normal IoU over the detection areas. JIoU is a probability version of IoU that better evaluates the probabilistic features of object detection. Slightly different from JIoU in the work of Wang et al. (2020b) that defines JIoU as the IoU between the probability mass covered by the detection and the ground truth bounding boxes, this work defines it as the IoU between the sum of the evidences in the detected bounding boxes and the sum of all the evidence masses describing this object. In addition to JIoU, the average evidence score for a single detected bounding box is also calculated. It is the mean of the drawn confidences of points inside the bounding boxes. A low average evidence score indicates that the predicted box is not fully filled with strong evidence due to inadequate observation. In this case, the geometry of the detected bounding boxes might not be accurate. A low JIoU indicates that the evidences of the detected object do not concentrate in the inside area of the detected bounding box; some evidence masses scatter outside the bounding box.

The example in Figure 7.10 shows that the average evidence score is very closely related to the quality of the detection. As shown in the last row, poorer detections tend to have less evidence inside the predicted bounding boxes. Moreover, JIoU reveals the alignment of the prediction and ground truth bounding boxes over the evidence masses. For objects without enough clues from the measurements, it is hard to define a perfect deterministic ground truth, even by manual labeling. In such cases, a reliable model should not be overconfident but be able to generate a possible result with a fair judgment of its result based on uncertainty. More specifically, both the detection and ground truth bounding boxes may have imperfect alignment with the evidence masses of the *GevBEV* map. This leads to a smaller probabilistic union between these two boxes; hence a higher JIoU is derived. Compared to IoU, this is more reasonable as JIoU decouples the imperfectness of the model and the measurement. For example, the detected bounding box in the lower left subfigure has a low IoU but a relatively high JIoU because there are too little evidences due to the lack of the observation points, *e.g.,* measurement imperfectness. In contrast, compared to the ground truth, the detection in the lower right subfigure has enough evidence but has both low JIoU and IoU, indicating that the inferior detection is not due to the insufficient measurement (*e.g.,* too few points observed) but to the model's limited performance.

## 7.5.2 Quantitative Analysis

### Comparison with state-of-the-art models

The *GevBEV* model is compared with the state-of-the-art models for the cooperative BEV semantic segmentation task on the simulated OPV2V dataset (Xu et al., 2022c) and the real dataset V2V4Real (Xu et al., 2023). Table 7.1 lists the results for segmenting roads and dynamic objects. It can be seen that the models (the CVT-based backbone for learning BEV feature maps from 2D images) designed for camera data are inferior to LiDAR data. This is because LiDAR data provide more accurate 3D information, which is essential for projection to a BEV map for the perception task. GevBEV outperforms all the other models, including models that are conducted on the same LiDAR data as GevBEV for a fair comparison. Compared to the CoBEVT model that came second in the OPV2V benchmark, the proposed GevBEV model with distribution heads improves IoU by 22.4% for segmenting dynamic objects and 4.6% for segmenting road surfaces. In the V2V4Real benchmark, it improves the IoU by 16.4% compared to V2XViT. These improvements indicate that the point-based spatial Gaussian effectively provides smoother information about each

| Model | Modality |  | OPV2V $IoU^{dense}\uparrow$ |  | V2V4Real $IoU^{dense}\uparrow$ |
|---|---|---|---|---|---|
|  | Camera | LiDAR | Road | Object | Object |
| AttFuse (Xu et al., 2022c) | ✓ |  | 60.5 | 51.9 | - |
| V2VNet(Wang et al., 2020a) | ✓ |  | 60.2 | 53.5 | - |
| DiscoNet (Li et al., 2021) | ✓ |  | 60.7 | 52.9 | - |
| CoBEVT (Xu et al., 2022a) | ✓ |  | 63.0 | 60.4 | - |
| Fcooper (Chen et al., 2019a) |  | ✓ | 70.3 | 52.1 | 25.9 |
| AttFuse (Xu et al., 2022c) |  | ✓ | 75.3 | 52.0 | 25.5 |
| V2XViT (Xu et al., 2022b) |  | ✓ | 75.0 | 50.4 | <u>29.9</u> |
| CoBEVT (Xu et al., 2022a) |  | ✓ | <u>75.9</u> | <u>52.3</u> | 29.6 |
| GevBEV (proposed) |  | ✓ | **79.5** | **74.7** | **46.3** |

*Table 7.1: Comparison with the state-of-the-art models on OPV2V and V2V4real dataset. Best values are highlighted in boldface and the second best values are underlined.*

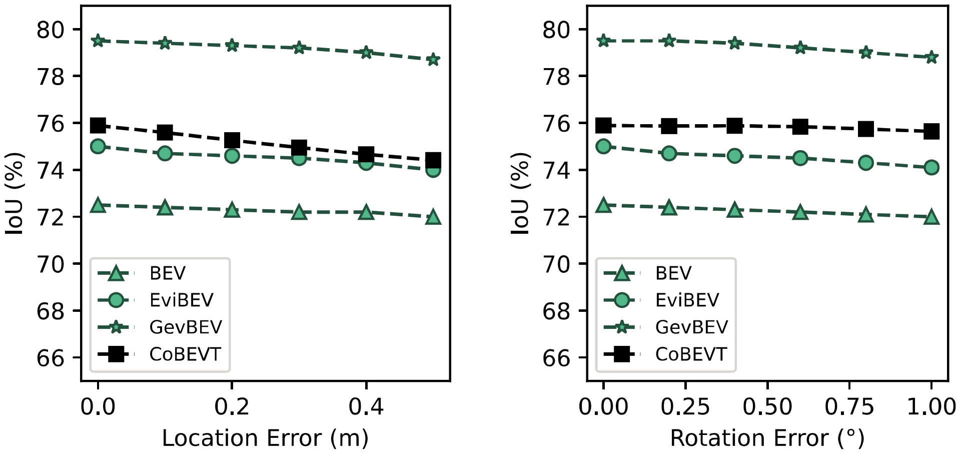


*Figure 7.11: OPV2V road segmentation result ($IoU^{dense}$) with localization noise.*

surface point's neighborhood, leading to more accurate results on both benchmarks. In addition, the proposed target sampling method for training in the continuous driving space is more robust against errors in ground truth. This leads to a remarkable improvement on the real V2V4Real dataset, which contains inaccurate labels in a real-world driving scenario.

However, real-world driving scenarios pose more challenges for the co-perception task by inevitably introducing localization errors, as indicated by the large performance gap on object segmentation between OPV2V (74.7%) and V2V4Real (46.3%), Also, the V2V4Real dataset only provides the communication between two connected vehicles, fewer than that of the simulated OPV2V dataset. Hence, the perception performance on V2V4Real is much worse than that tested on OPV2V.

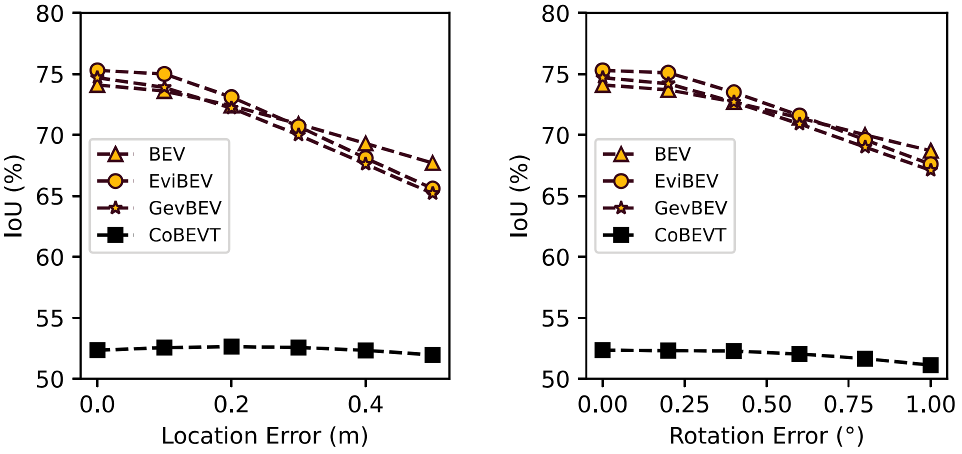


*Figure 7.12: OPV2V object segmentation result ($IoU^{dense}$) with localization noise.*

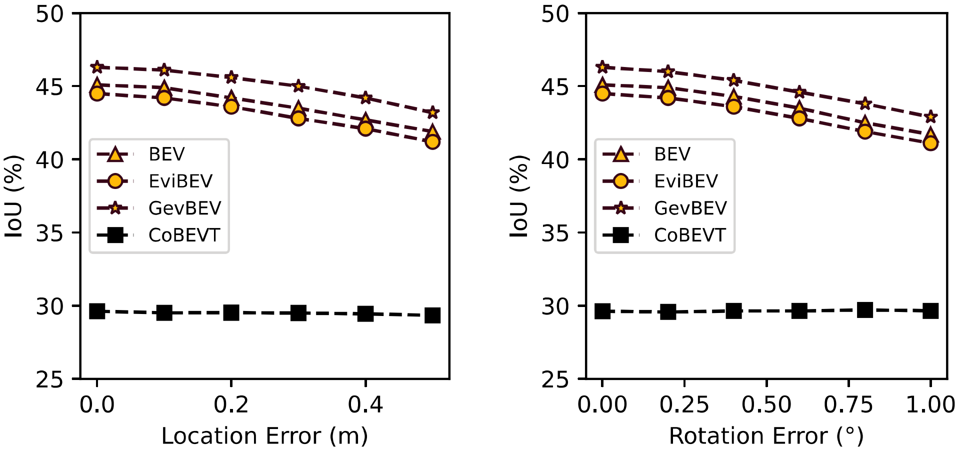


*Figure 7.13: V2V4Real object segmentation result ($IoU^{dense}$) with localization noise.*

### Sensitivity to localization noise

In the previous experiments with the simulated data, perfect localization information is assumed. In order to evaluate the influence of localization noise on the BEV Semseg results, localization errors are generated by normal distributions with a standard deviation ranging from 0 to 0.5 meters for position and from 0 to 1 degree for orientation. Figure 7.11 demonstrates that all models experience slight performance declines in road surface classification as location and rotation errors increase. Still, the proposed *GevBEV* model outperforms CoBEVT, maintaining the best performance with a margin of approximately 4% across all error configurations. Without learning in continuous space with the Gaussian distributions, *BEV* and *EviBEV* perform worse than both CoBEVT and *GevBEV*. In contrast to the road surface, object classification results exhibit a higher sensitivity to localization errors due to their smaller size. In this setting, *GevBEV* still outperforms the CoBEVT runner-up model on the OPV2V and V2V4Real datasets, as depicted in fig. 7.12 and fig. 7.13. This indicates that the proposed approach is more robust than CoBEVT in dealing with localization noise.

### Ablation study

Table 7.2 shows the performance of the ablation models. In general, the baseline model *BEV* without the point-based spatial Gaussian (G.s.) and the evidential loss $\mathcal{L}_{edl}$ is inferior to the other models. This indicates that this conventional deterministic model trained by optimizing the cross-entropy is not as good as the probabilistic models. In contrast, by modeling each point with a spatial continuous Gaussian distribution, it is able to close the gaps caused by the sparsity of point clouds and generate smoother BEV maps. *EviBEV* with the point-based spatial Gaussian performs better than the baseline for the metric $IoU^{\text{dense}}$ and $IoU^{\text{sparse}}$. However, its performance for objects (vehicle) classification is slightly degraded.

| Model | Modules | | | OPV2V $IoU^{dense}\uparrow$ | | V2V4Real $IoU^{dense}\uparrow$ | OPV2V $IoU^{sparse}\uparrow$ | | V2V4Real $IoU^{sparse}\uparrow$ |
|---|---|---|---|---|---|---|---|---|---|
| | G.s. | $\mathcal{L}_{edl}$ | FSA | Road | Object | Object | Road | Object | Object |
| CoBEVT | | | | 75.9 | 74.7 | 29.6 | - | - | - |
| BEV | | | ✓ | 72.5 | 74.1 | 45.1 | 76.1 | 75.8 | 46.1 |
| EviBEV | | ✓ | ✓ | 75.0 | **75.3** | 44.5 | 78.3 | **76.3** | 45.3 |
| GevBEV$^-$ | ✓ | ✓ | | 59.7 | 73.1 | 46.0 | 62.5 | 73.2 | 46.7 |
| GevBEV | ✓ | ✓ | ✓ | **79.5** | 74.7 | **46.3** | **83.1** | 76.1 | **46.9** |

*Table 7.2: Ablation study of the proposed modules. All IoU results are measured in percent. Best values are highlighted in boldface. G.s.: point-based spatial Gaussian, $\mathcal{L}_{edl}$: evidential loss, FSA: free space augmentation.*

Moreover, *BEV* and *EviBEV* perform worse than CoBEVT (Xu et al., 2022a) on $IoU^{\text{dense}}$ for road semantic segmentation. This is because *BEV* and *EviBEV* uses a fully sparse convolutional network, which does not operate in unobserved areas. In order to facilitate a comparison with dense convolution models, certain road surfaces in the proposed sparse BEV Semseg models are considered inadequately observed and thus treated as false predictions when calculating $IoU^{\text{dense}}$. Unlike *GevBEV*, both the BEV and EviBEV models have additional areas designated as unobserved due to the absence of Gaussian tails. Consequently, this leads to lower IoU values. However, it is worth mentioning that the object classification of the ablation models significantly outperforms the runner-up model, CoBEVT. This can be attributed to two factors. Firstly, the proposed models carefully control the network to expand coordinates in a specific range and only make predictions over the observable areas. The coordinate expansion module can cover most of the object areas so that these areas will be given a prediction rather than being treated as unobserved. Secondly, thanks to the benefits of dynamic sampling from continuous driving space during training, the proposed models show a tendency to be cautious when making positive predictions for vehicle points. This cautious approach facilitates the process of learning the edge details of the vehicles more effectively, enhancing the overall object classification performance.

From the comparison between *GevBEV*$^-$ and *GevBEV*, the improved IoUs, especially for roads, indicate that free space augmentation (FSA) provides an explicit cue to the unoccupied space along the ray paths and improves the detection performance. In addition, this module plays an important role in mitigating the problem introduced by the sparsity of point clouds and greatly improves prediction performance. Overall, *GevBEV* outperforms the ablation models in all measurements, confirming the efficacy of each proposed module.

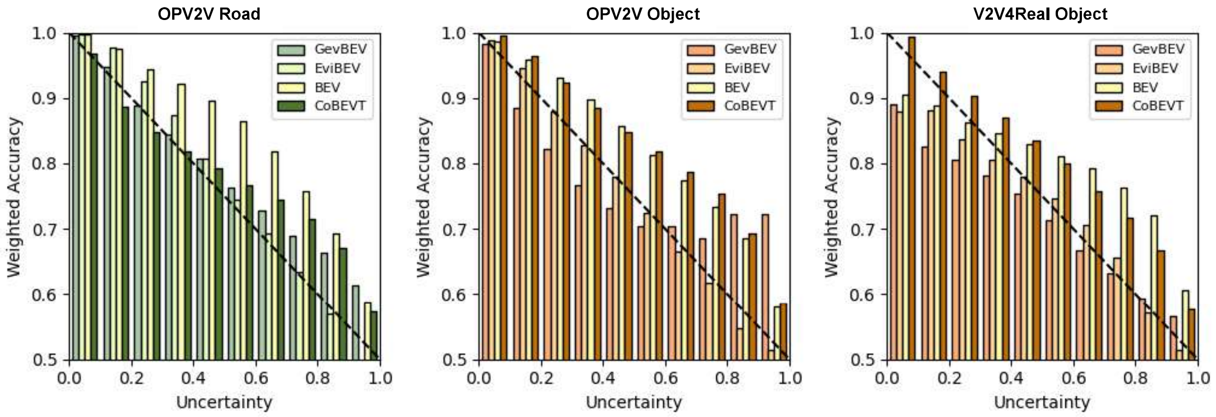


*Figure 7.14: Calibration plots by different models. The perfect calibration line is indicated by the diagonal dashed line.*

| Model | OPV2V | | V2V4real | Average ↓ |
|---|---|---|---|---|
| | Road↓ | Object ↓ | Object↓ | |
| BEV | 0.095 | 0.072 | 0.076 | 0.081 |
| EviBEV | **0.031** | **0.010** | **0.030** | **0.023** |
| CoBEVT | 0.044 | 0.075 | 0.056 | 0.058 |
| GevBEV | 0.044 | 0.066 | 0.040 | 0.050 |

*Table 7.3: Average Calibration Error. Best values are highlighted in boldface and the second best values are underlined.*

### Desired confidence level with calibration plot

The calibration plot (Figure 7.14) and the average calibration error (Table 7.3) between this plot and the perfect uncertainty-accuracy line (dashed black line) are used to analyze the quality of the predictive uncertainty generated by the baseline models CoBEVT and *BEV*, the *EviBEV* model with $\mathcal{L}_{edl}$ loss, and the complete probabilistic model *GevBEV*. Since the baseline model only generates Softmax scores for each class, these scores are converted into entropy to quantify the uncertainty so that they can be compared with the other two models with the evidential uncertainty based on a Dirichlet distribution. As revealed in fig. 7.14, *GevBEV* and *EviBEV* demonstrate better confidence plots relative to the perfect calibration line (indicated by the diagonal dashed line) than the baseline model *BEV* and CoBEVT for both road and object classification. The two baseline models seem to underestimate the uncertainty compared to the other two models, which confirms the concerns that the deterministic model, without particularly accounting for uncertainties, may end up generating less trustworthy scores for making driving decisions. The results, on the other hand, show that assuming a Dirichlet distribution of the point class of the BEV map can provide more reliable probabilistic features for the map and, therefore, is safer to be used in IA perception systems.

Interestingly, the uncertainty quality of *GevBEV* is worse than that of *EviBEV*, especially for object classification. This might be caused by the saturation of the summation of the evidences contributed by the neighboring center points. This saturation weakens the model's ability to represent the uncertainty. One conjecture of this result is that some vehicles are only partially observed due to occlusion. The expandable convolution is limited to expanding the coordinates to cover the whole body of the vehicles. Therefore, only the distribution tiles of these expanded center points can cover the rest of the vehicle body. Points only covered by these distribution tiles

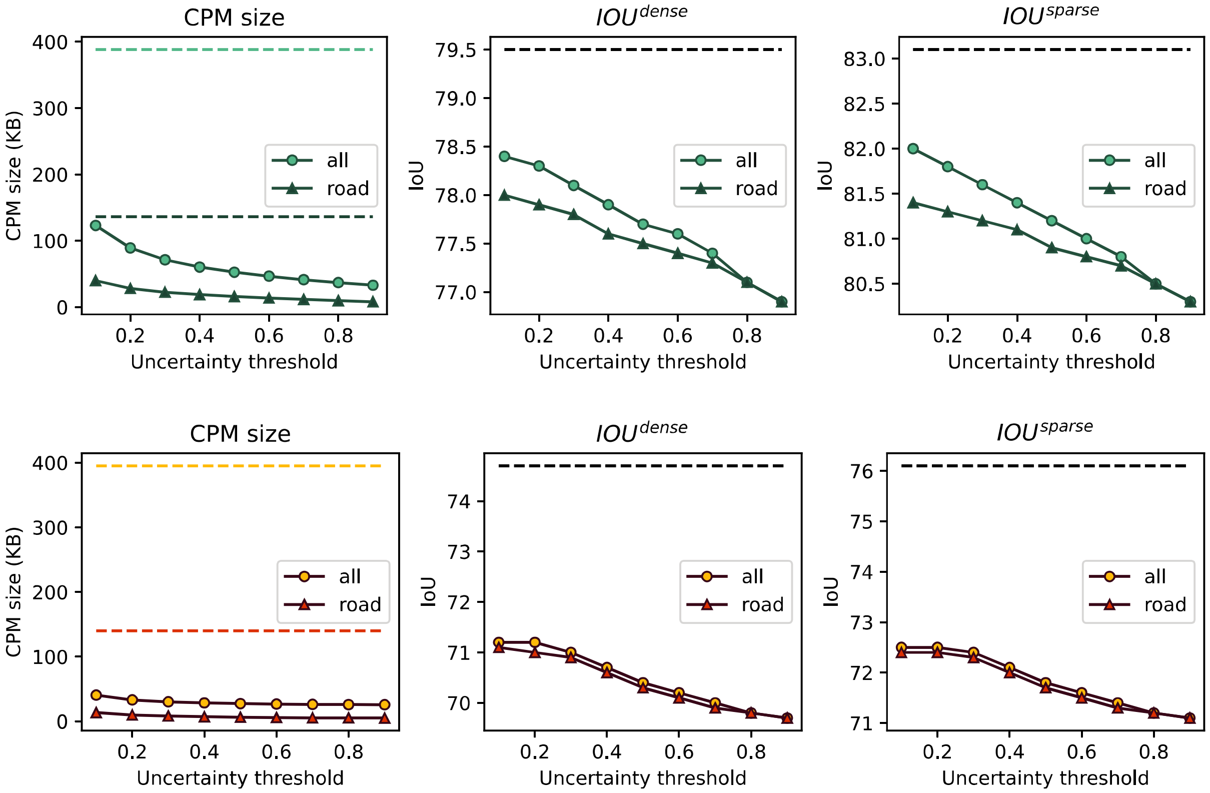


*Figure 7.15: Comparison of different CPM sharing strategies ($CPM^{all}$ and $CPM^{road}$, ref. Section 7.2.4) for co-perception. The first row shows the results of the road head (greenish) and the second the objects head (orangish).*

usually have high uncertainties. However, most of them are classified as foreground because of the large predictive foreground evidence of the center point. This leads to a high true positive rate with high uncertainty and introduces bias in the uncertainty estimation. Despite the uncertainty of *GevBEV* being slightly more conservative than that of *EviBEV*, still, as shown by the higher IoUs in table 7.2, the learned spatial Gaussian distribution generates smoother BEV maps and draws classification distribution of any points in the continuous BEV 2D space.

### 7.5.3 Uncertainty-based information selection for collective perception

In general, the uncertainties generated by *GevBEV* indicate how certain the model is about its predictive classification results. Higher uncertainty reveals a lower observation level, and more useful information is required to be shared by other IAs. Therefore, uncertainty can be used as a metric for information selection to share CPMs among IAs. To implement this selection process, different uncertainty thresholds are tested and evaluated. A lower uncertainty threshold can be used when communication channels are busy. In this way, only the more reliable perception results that have low uncertainty are selected and shared.

The CPM sizes before and after the uncertainty-based information selection with different uncertainty thresholds, as well as the corresponding classification results $IoU^{\text{sparse}}$ and $IoU^{\text{dense}}$, are plotted in Figure 7.15. The first row shows the results of the static head for the road surface class and the second the dynamic head for the object class. As discussed in Section 7.2.4, experiments with two CPM sharing strategies are conducted, one for sharing the masked evidence maps of all perception areas (*all*, plotted with circles) and one only for the road areas constrained by an existing HD map (*road*, plotted with triangles). The dashed lines are the baselines for sharing

CPMs without information selection, which are shown as horizontal lines over different uncertainty thresholds.

The plots in the first column show that the CPM sizes have evidently been reduced after the information selection at all uncertainty thresholds compared to the corresponding baselines. For example, when all perception areas (light green circles) are considered for sharing, the CPM size for the road head dropped by approximately 87% from 388 KB to 52 KB at the uncertainty threshold of 0.5. Consequently, $IoU^{\text{dense}}$ and $IoU^{\text{sparse}}$ only dropped by about 2%. In the same configuration, the CPM sizes for the object head dropped ca. 93%, from 395KB to 27KB, while the IoUs dropped ca. 4%. By only considering the road areas for sharing (solid lines with triangle markers), CPM sizes can be further reduced to about 16 KB for the road head and 6 KB for the objects head at the uncertainty threshold of 0.5 with only an IoU drop within 0.5%. According to the V2V communication protocol (Arena and Pau, 2019), without considering other communication overhead, the data throughput rate can achieve 27 Mbps. Therefore, the time delay for sending the CPMs of both heads has dropped from ca. 28 ms to 0.8 ms by the data selection based on our GevBEV maps from road areas. These significantly reduced time delays are critical for real-time V2V communication.

### 7.5.4 Summary

This chapter proposes a novel method *GevBEV* to generate BEV semantic segmentation maps from sparse and discrete BEV points. GevBEV leverages spatial Gaussian distributions to interpolate neighboring spaces and Dirichlet distributions to estimate uncertainties. These distributions not only establish the statistical properties of the segmentation results but also ensure their reliability in a back-traceable manner. Specifically, any segmentation result for an informative point (uncertainty below 1) in the continuous driving space is supported by evidence from the original observed LiDAR points. Experimental evaluations on the OPV2V and V2V4Real benchmarks demonstrate that GevBEV significantly outperforms state-of-the-art models. By integrating the Dirichlet distribution and evidential learning, GevBEV achieves more reliable classification scores, producing better-calibrated uncertainty estimates. Furthermore, these uncertainties facilitate effective CPM selection, reducing the average CPM size by approximately 90%.

# 8 Time-Aligned Cooperative Object Detection Using Fully Sparse Neural Network

Despite the significant successes of cooperative object detection in previous studies, most approaches rely on dense convolutions or transformers to process sparse point-cloud data, leaving the potential of sparse operations largely unexplored. Furthermore, prior studies often assume synchronized sensors across all IAs, overlooking the challenges that asynchronous sensors can pose to the data fusion process. To address these issues, this chapter introduces *SparseAlign* (Yuan et al., 2025), an efficient and fully sparse framework for Time-Aligned Cooperative Object Detection (TA-COOD) that explicitly accounts for sensor asynchrony. First, the definition of TA-COOD is outlined in Section 8.1. Next, the overall structure of *SparseAlign* is presented in Section 8.2, including a novel 3D backbone network, *SUNet*, and three alignment modules for temporal, pose, and spatial alignment, respectively. Finally, the experiments and their results are discussed in Section 8.4.

## 8.1 Definition of TA-COOD

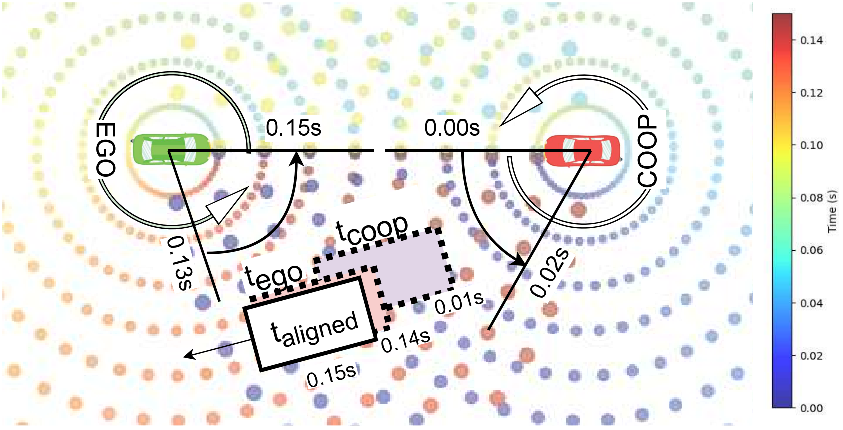


*Figure 8.1: TA-COOD with two CAVs. Dashed boxes: Ground truth bounding box of ego- and coop-CAV observations. Solid box: time-aligned Ground Truth bounding box.*

In a cooperative perception scene, assume that a set of IAs $\mathbf{A} = \{A_i | 0 \leq i \leq N\}$ are interconnected, where $A_0$ is the ego IA. Each IA has its own sensor ticking time $t_i$. All sensors are scanning at the frequency $f$. An example with two CAVs is shown in Figure 8.1. The LiDAR of the ego IA $A_0$ (green) and the cooperative IA $A_1$ (red) scan counterclockwise. They have a ticking frequency of $f = 10Hz$ and the ticking time offset of, say, $\delta t = t_0 - t_1 = 0.05s$. The scanning time of each observation point is shown in gradient colors. As shown in the middle-bottom area of Figure 8.1, the observations of the ego- and the coop-vehicle in this area have a time difference of $0.11s \leq t_{ego} - t_{coop} \leq 0.15s$, because the cooperative IA scans this area in the time range of $0s \leq t \leq 0.02s$ and the ego IA scans in the time range of $0.13s \leq t \leq 0.15s$. At a speed of $60km/h$, this time offset can lead to the target object shift of more than $1.8m$, introducing difficulties for the data fusion of the IAs. Instead of taking the annotation from one IA (*e.g.,*the red bounding box observed by the ego IA at $t_{ego}$), a ground-truth bounding box is generated at the globally aligned time $t_{aligned}$ by interpolating between the annotations of the individual CAVs. To accurately model

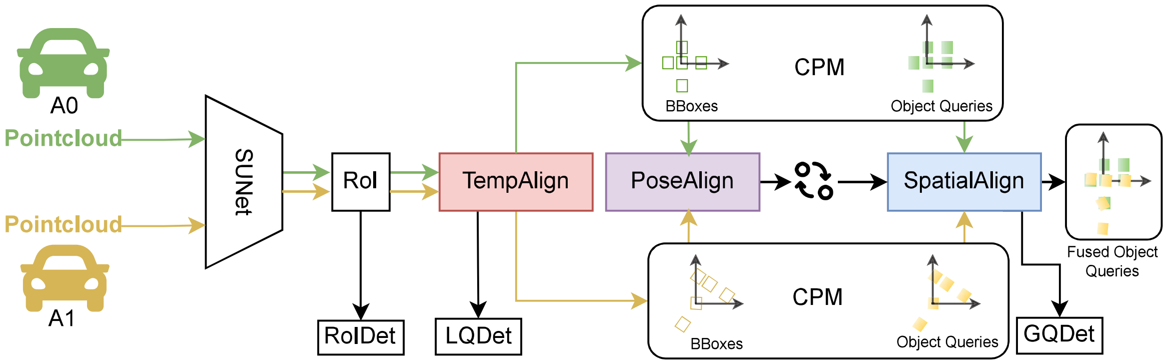


*Figure 8.2: Overview of of the SparseAlign pipeline for TA-COOD, including the sparse SUNet as backbone network and three alignment modules for temporal (TempALign), pose (PoseAlign) and spatial (SpatialAlign) data alignment.*

the movement of the detected object, 3D TA-COOD is introduced, which aims to take point-wise timestamped point clouds as input and predicts the bounding box at globally aligned time $t_{aligned}$, which is the scan end of the reference vehicle in each frame. The detection results are evaluated from the perspective of the ego vehicle, which serves as the reference vehicle. In the example shown in Figure 8.1, the global time is aligned to $t_{aligned} = 0.15$ s, corresponding to the scan end of the ego vehicle.

## 8.2 Architecture of the Framework SparseAlign

### 8.2.1 Overview

As illustrated in Figure 8.2, a fully sparse and highly efficient pipeline *SparseAlign* is built to process sequential point-cloud data from multiple agents for TA-COOD. All agents share an identical pipeline structure and module weights. Each agent first processes its own input point cloud locally using feature extraction modules, and then selectively shares key feature points with others. Considering the cooperative perception system as a whole, the input at time $t_i$ is the point cloud $PC_{t_i}$ of the ego agent and all point clouds $PC_{t_j}$ of the cooperative agents. Note that $t_j <= t_i$ when communication latency exists. All point clouds are processed locally at each IA by a novel 3D Sparse UNet (*SUNet*) backbone network to extract the point cloud features. The Region of Interest (*RoI*) module then selects the most interesting features as the object queries for further processing. For simplicity, object queries are defined as $Q^* = \{(F_i^*, x_i^*, y_i^*) | i \in \{1, \ldots, N^*\}$, where $*$ represents the query set name, $F$ is the feature, and $x$,$y$ are the coordinates of the corresponding $i$-th query. With selected queries, the TempAlign Module (*TAM*) can efficiently align the query features with the globally aligned timestamp $t$ so that errors introduced by communication delay and sensor asynchrony can be compensated. The updated query features and the detections from the Local Query Detection head (*LQDet*) are then shared as CPMs. The PoseAlign Module *PAM* then uses the detected bounding boxes to correct the relative poses between the cooperative and the ego agent and the SpatialAlign Module *SAM* fuses the received cooperative features into the ego coordinate system based on the corrected transformation $T_c^e$ from cooperative to ego coordinates.

### 8.2.2 Sparse UNet (SUNet)

Cooperative perception is a computationally demanding task as it requires processing data from multiple IAs. To improve efficiency and reduce GPU memory consumption, fully sparse 3D convolutions are used to construct the 3D backbone to extract the point cloud features. However, building a fully sparse backbone network that can compete with the dense ones is non-trivial. For

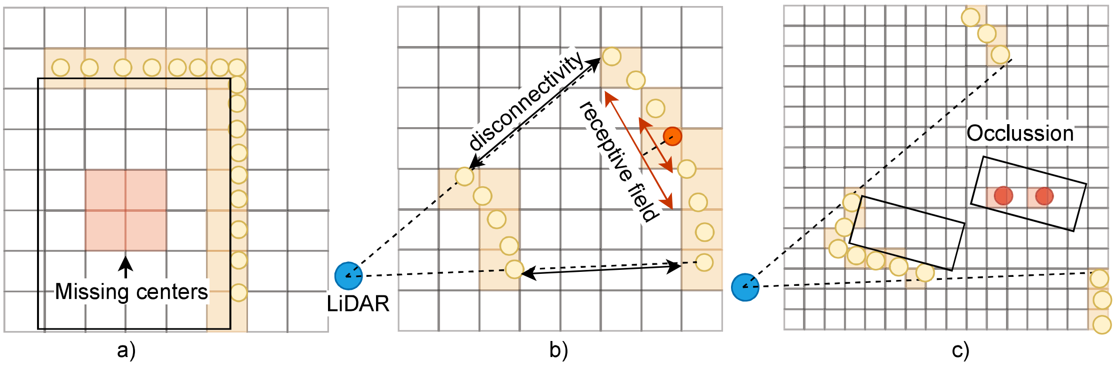


*Figure 8.3: Issues of sparse convolutional backbone networks. a) Center Feature Missing (CFM). b,c) Isolated Convolution Field (ICF) caused by ring disconnectivity and occlusion.*

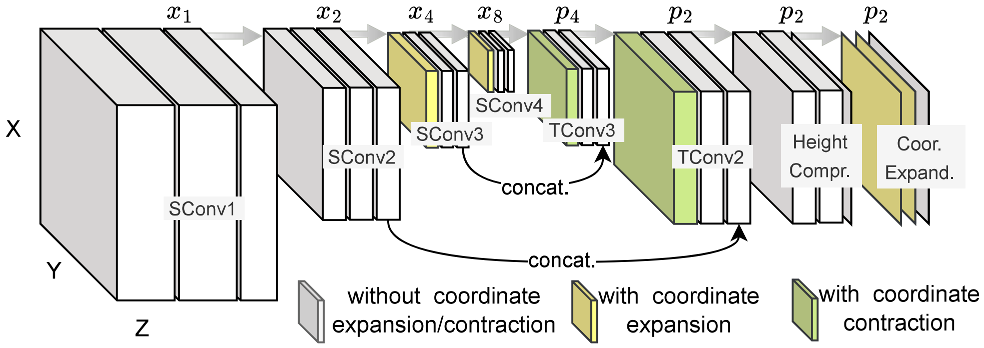


*Figure 8.4: The architecture of the proposed 3D backbone network SUNet (Sparse UNet).*

the object detection task, there are two issues. The first unavoidable issue is *Center Feature Missing* (Fan et al.) (CFM, shown in Figure 8.3 a)). Due to the range-view of on-board LiDARs, there are usually no scan points in the vehicles' center area. However, compared to the edge points, the center points have a better ability to represent the whole object. Lacking center points for learning, the sparse frameworks perform worse than the dense ones that directly learn the features for center points. The second issue is the poor connectivity between points scanned by different laser beams. As shown in Figure 8.3 b), this disconnectivity issue in distant areas may lead to isolated voxel blobs; the receptive field only enlarges to the blob scale as the convolutional layers go deeper. The isolated voxel blobs never exchange information with each other, limiting the backbone's capability of capturing global features. This issue is named as *Isolated Convolution Field* (ICF). It also happens to the occlusion areas as shown in Figure 8.3 c): although the occluded vehicle has very few scan points (red), it is expected to be detected based on the neighboring vehicles and the clues in the previous frames. However, ICF introduces difficulties for these isolated points to aggregate information from neighbors and the global environment.

To overcome the above problems, a sparse UNet backbone network *SUNet* (Figure 8.4) is built with the *Coordinate-Expandable sparse Convolution* (CEC, ref. Section 3.1, sparse convolution), which is implemented in the MinkowskiEngine (Choy et al., 2019) library. Specifically, *SUNet* is a sparse 3D version of UNet (Ronneberger et al., 2015) similar to MinkUnet (Choy et al., 2019) as it naturally combines local and global features in the up-sampling process of the UNet. To overcome the ICF issue, 3D CECs in the sparse convolution (*SConv*) block 3 and 4 are employed. The expanded coordinates after these CEC layers are then contracted in the transposed sparse convolution (*TConv*) block 3 and 2. On the one hand, the output coordinates of the *TConv* blocks $p4$ and $p2$ can match the coordinates in stride $x2$ and $x4$ for the concatenation. On the other hand,

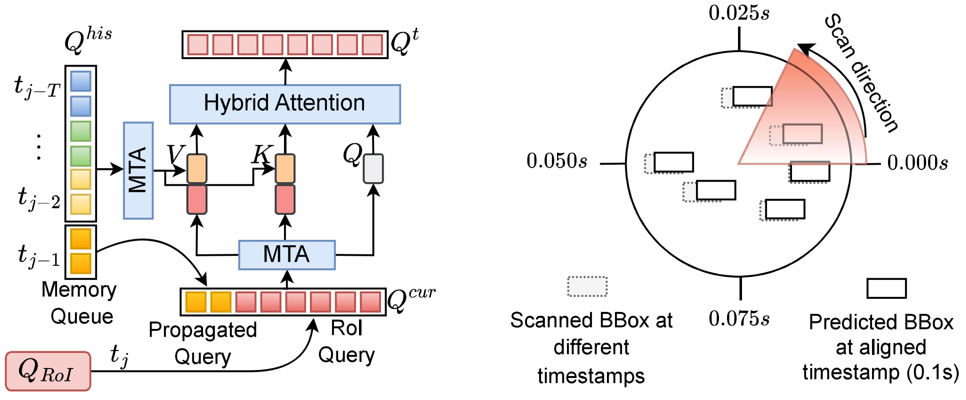


*Figure 8.5: Temporal alignment module (TAM)*

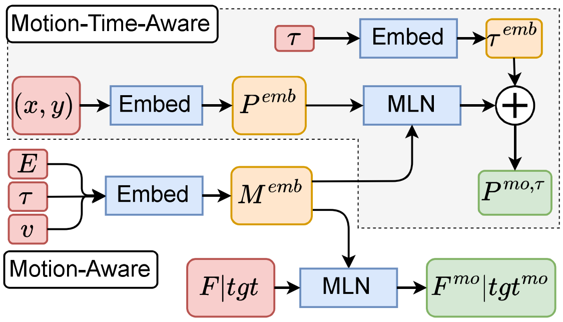


*Figure 8.6: MTA alignment: Motion-Time-Aware embeds the query positions ($Q_p$) and Motion-Aware embeds the query context $Q_c$ and initial queries (tgt).*

it learns better global features without voxel number soaring in the subsequent layers and maintains efficiency. Against CFM, 2D CECs are employed on the 2D BEV sparse features (Figure 8.4: Coor. Expand.) to expand the coordinates to ensure their coverage over the object centers.

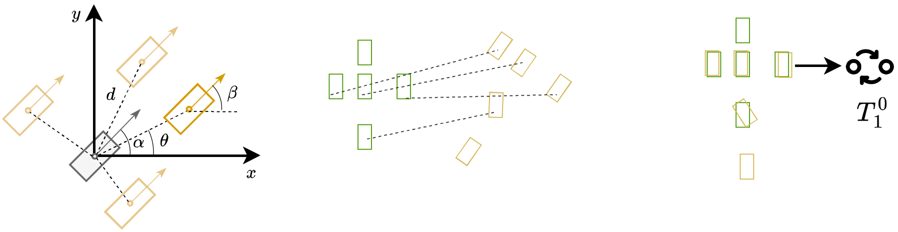
### 8.2.3 Temporal alignment (TAM)

The TempAlign Module (*TAM*, Figure 8.5) aims to learn the temporal context relative to the previous frames to achieve two goals: to improve object detection performance by looking for clues in the history and to compensate for the object displacement errors introduced by asynchronous scanning time (Figure 6.6, right). Technically, the *TAM* is inspired by Wang et al. (2023). It mainly contains a memory queue, a *Motion-Temporal Awareness* (*MTA*) module and a *Hybrid Attention* module.

**Memory queue**: stores $K_q = 256$ object queries for each of the $T$ previous frames. For each selected object query $Q_i$, the predicted object center $x_i$,$y_i$, the context embedding $F_i$, the velocity $v_i$, the pose matrix $E_i$ and the query timestamp $\tau_i$ of the corresponding IA are stored. The timestamp $\tau_i$ is retrieved following Equation (8.1), where $x_i$,$y_i$ are the $x$ and $y$ coordinate of the query point, $P_x^i$, $P_y^i$ and $P_t^i$ are respectively the $x$,$y$ coordinates and the timestamp of the $j$-th point $P^j$ in the point cloud.

$$\tau_i = P_t^j, \quad j = \mathrm{argmin}\,|\,\mathrm{atan2}(x_i,y_i) - \mathrm{atan2}(P_y^j,P_x^j)| \tag{8.1}$$

**MTA**: is used to inject additional information of each query into the query features $F$. As shown in Figure 8.6, the timestamp $\tau$ of each query, the pose $E$ and the velocity $v$ of the corresponding ego vehicle are aligned as a vector and embedded in the feature $M^{emb}$. With the Motion-aware Layer Normalization (MLN, Wang et al. (2023)), this feature is then injected into the embedded query



1. Graph Feature 2. Match 3. Align

*Figure 8.7: Pose alignment module (PAM)*

position $P^{emb}$, the query context features $F$ of previous frames, and the initial query target features $tgt$ at the current frame, to obtain the motion-aware embeddings. Additionally, the embedded time is added to the motion-aware query position embedding ($P^{mo}$) to make it also time-aware ($P^{mo,\tau}$). The final output of the $MTA$ module is then the motion-time-aware position embedding $P^{mo,\tau}$ and the updated query features, either $F^{mo}$ or $tgt^{mo}$. As shown on the left of Figure 8.6, all query features, $Q^{cur}$ from $t_j$ and $t_{j-1}$ and $Q^{his}$ from $t_{j-2} \leq t \leq t_{j-T}$, are updated by the $MTA$ module. In addition, a target query feature $tgt$, in the same feature shape (dimension) as $Q^{cur}$, is initialized with zeros and updated with the $MTA$ module to obtain the motion-aware feature $tgt^{mo}$.

**Hybrid Attention**: takes the target feature $tgt^{mo}$ as the query input $Q$ (Figure 8.6, gray rectangle), the concatenation of the updated query features $Q^{cur}$ and $Q^{his}$ as the input key $K$ and value $V$, to complete the attention process. Note that the position embedding $P^{mo,\tau}$ of each query is also added to the corresponding features to ensure that the queries are spatially aware of their locations. The output of the attention is notated as $Q^t$. It has learned the temporal context from the sequential data and is able to make predictions for the future state. These predicted bounding boxes are synchronized with the same timestamp. Therefore, the subsequent module PoseAlign ($PAM$) only needs to deal with object displacement caused by localization errors.

### 8.2.4 Pose alignment (PAM)

The pose alignment module (PAM) is designed to correct the relative pose between the ego and the cooperative agents. Given the detected bounding boxes $\mathbf{B}_i$ and $\mathbf{B}_j$, respectively, from the perspective of the agent $A_i$ and $A_j$ in the same scene, a human can easily find the correspondences between $\mathbf{B}_i$ and $\mathbf{B}_j$ based on the geometric relationships. Practically, this thesis models the features of these geometric relationships by embedding the neighborhood geometry of each bounding box into deep features and then matching these features of $\mathbf{B}_i$ and $\mathbf{B}_j$ with the Hungarian algorithm. For each bounding box $b$ in a bounding box set $\mathbf{B}$, the nearest $k = 8$ neighbors are selected to embed its neighborhood features. As shown in Figure 8.7, *1.Graph Feature*, the orientation angles of the bounding box are $\alpha$ and $\beta$ for $b$ and its neighbor, respectively. The relative orientation is featured with $\nu_a = [\sin(\beta - \alpha); \cos(\beta - \alpha)]$[1]. Similarly, the relative edge direction is embedded as $\epsilon_a = [\sin(\theta - \alpha); \cos(\theta - \alpha)]$. In addition, the dimension $\nu_{dim} = [l,w,h]$ of the neighboring bounding box, and the Euclidean distance between $b$ and its neighbors $\epsilon_d = d$, are also embedded into the feature. Note that $\epsilon_d$,$\epsilon_a$,$\nu_a$ are all relative features between two bounding boxes that are pose-agnostic, leading to robust pose alignment without dependence on the magnitude of initial pose errors. These features are then concatenated and embedded into high-dimensional features by a *linear* layer with ReLU activation ($\mathbb{R}^5 \rightarrow \mathbb{R}^{d_{emb}}$):

$$f_{nbr} = linear([\epsilon_d; \epsilon_a; \nu_a; \nu_{dim}]) \tag{8.2}$$

---

[1] $[\cdot;\cdot]$: concatenation

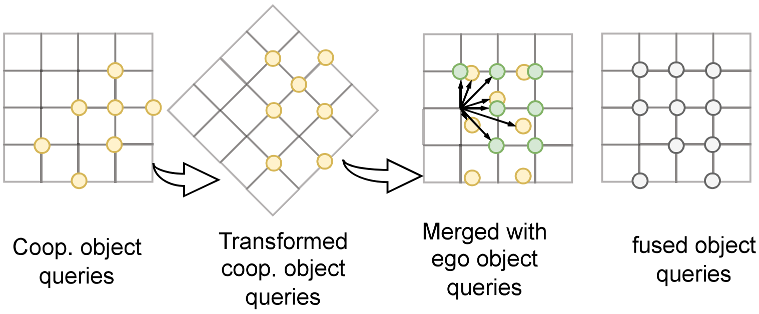

Coop. object queries
Transformed coop. object queries
Merged with ego object queries
fused object queries

*Figure 8.8: Spatial alignment module (SAM)*

Assembling the features relative to all neighbors, the neighborhood feature $F_{nbr} \in \mathbb{R}^{k \times d_{emb}}$ is summarized by a multi-head self-attention module *attn* followed by a *linear* layer operating on the *mean* and *max* of the *attn* outcome, resulting in the feature $F_{nbr} \in \mathbb{R}^{d_{emb}}$:

$$F_{nbr} = attn(F_{nbr}) \tag{8.3}$$

$$F_{nbr} = linear([mean(F_{nbr}); max(F_{nbr})]) \tag{8.4}$$

By calculating the Euclidean distance between the learned neighborhood feature $F_{nbr}$ of the bounding box in $\mathbf{B}_i$ and $\mathbf{B}_j$, the cost matrix is constructed for the linear sum assignment to find the best *Match* (fig. 8.7, *2.Match*) between these two bounding box sets. However, an object in $\mathbf{B}_i$ does not always have a projection in $\mathbf{B}_j$ and vice versa. Thus, the wrong *Matches* that have large distances are rejected. Based on the *2.Match* result, the relative pose transformation between $A_i$ and $A_j$ can be recovered, completing the *3.Align* step. Additionally, pose graph optimization is used to refine alignment by leveraging loop closure among multiple cooperative agent poses following Lu et al. (2023).

### 8.2.5 Spatial alignment (SAM)

The spatial alignment module (SAM) aims to merge and fuse the features of different IAs. To align the cooperative query features $Q^c$ with the ego query features $Q^e$, there are two issues to be resolved. First, the features $F^c$ are learned in the cooperative coordinate system and must be adapted to the ego coordinate system. This is achieved by a multilayer perceptron (*mlp*) as described in Equation (8.5) where $F^R \in \mathbb{R}^{N \times 9}$ are the rotation parameters obtained by flattening and repeating the rotation matrix $R \in \mathbb{R}^{3 \times 3}$.

$$F^c = mlp([F^c; F^R]) \tag{8.5}$$

Second, the cooperative query coordinate can not be perfectly aligned to the discretized grid in the ego coordinate system, as shown by the yellow point in Figure 8.8, *SAM.* Thus, the cooperative query points are merged into the nearest grid points of the ego coordinate system. The locations of the fused output object query points $Q^{ts}$ are shown with gray points. For each point in $Q^{ts}$, the features of its $k = 8$ nearest neighbors (Figure 8.8, black arrows) in query set $Q^{\cup} = Q_c \bigcup Q_e$ are fused into features $F^{ts}$. Mathematically, it reads as

$$F_{ij} = mlp([Q^{\cup}_j; linear([x^{ts}_i - x^{\cup}_j, y^{ts}_i - y^{\cup}_j])]) \tag{8.6}$$

$$F^{ts}_i = \{F_{ij} | j \in \{1, \ldots, k\}\} \tag{8.7}$$

$$F^{ts}_i = max_j(F^{ts}_i) + mean_j(F^{ts}_i) \tag{8.8}$$

where $i$ is the query index of $Q^{ts}$ and $j$ is the neighbor index of the $i$-th query.

### 8.2.6 Detection heads

To train *SparseAlign*, three center-based object detection heads are attached to each of the *RoI*, *TAM* and *SAM* modules to learn the temporal and spatial alignment features of object queries. These heads are notated as *RoIDet*, *LQDet* and *GQDet*, respectively. They share the same target-encoding method. The Focal Loss is used for object classification; all points inside the ground-truth bounding boxes are positive, and those outside are negative. *Smooth L1* loss is used for the bounding-box regression. The dimensions of the bounding box $l$,$w$,$h$ are regressed directly and the target location of the bounding box is encoded with the coordinate offsets between the feature points and the ground-truth bounding boxes. For the orientation of the bounding boxes, the novel *CompassRose* encoding (See Figure 6.8) with respect to the four anchor angles $\mathbf{r}_a = [0,0.5\pi,\pi,1.5\pi]$ is utilized.

## 8.3 Experiments

### 8.3.1 Comparative models

Similar to the efficiency experiments conducted in Chapter 6, the state-of-the-art models, F-Cooper (Chen et al., 2019a), AttnFusion (Xu et al., 2022c) and FPVRCNN (Yuan et al., 2022) are used as baseline models to conduct comparative experiments with *SparseAlign.* To make a fair comparison of the TA-COOD task, the three alignment modules, *TAM*, *PAM*, *SAM*, are attached to each comparative model to build the temporal interaction. They are notated as *[MODEL]+SA*. For F-Cooper and AttnFusion, intermediate BEV features at the corresponding resolution are taken for the RoI selection. For FPVRCNN, the keypoints generated by the Voxel-Set-Abstraction (Shi et al., 2020) module are used for this purpose. Since the original implementation of the model AttnFusion (Xu et al., 2022c) uses the memory demanding dense 3D convolutions, making it not possible to process streaming LiDAR data on a single GPU, the dense 3D convolutions in this model are replaced with sparse 3D convolutions. Besides, all comparative models use the anchor-based detection head as *RoIDet*, following their original implementations. The baseline method *StreamLTS* (Yuan and Sester, 2024), which is specifically designed for TA-COOD, is included for comparison.

In addition, the proposed model is also compared with previous works on the COOD task to validate the performance gain on the benchmarks. For this task, the reported performances are directly used for comparison, without reimplementation.

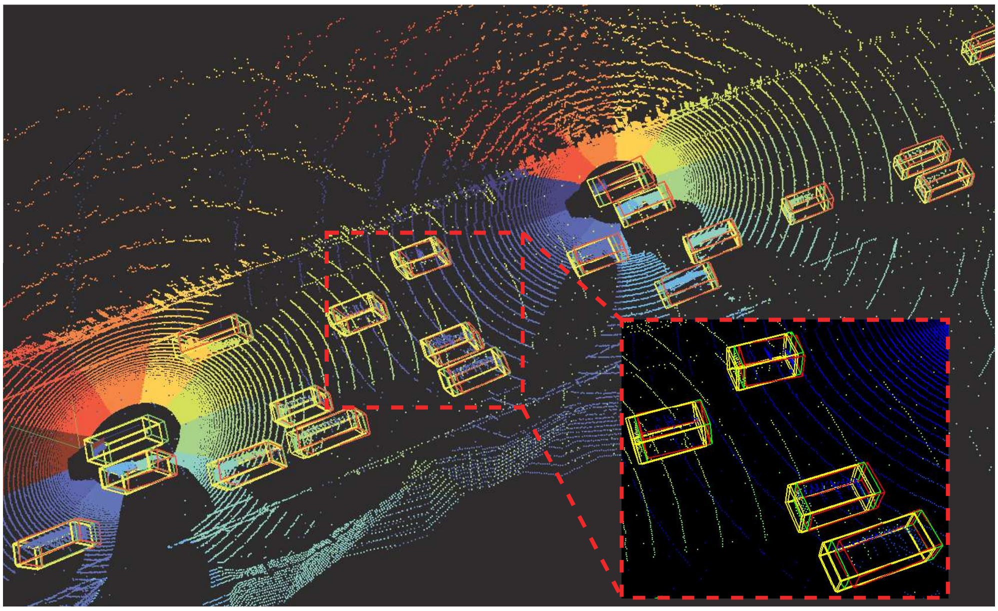
### 8.3.2 Training

#### Datasets

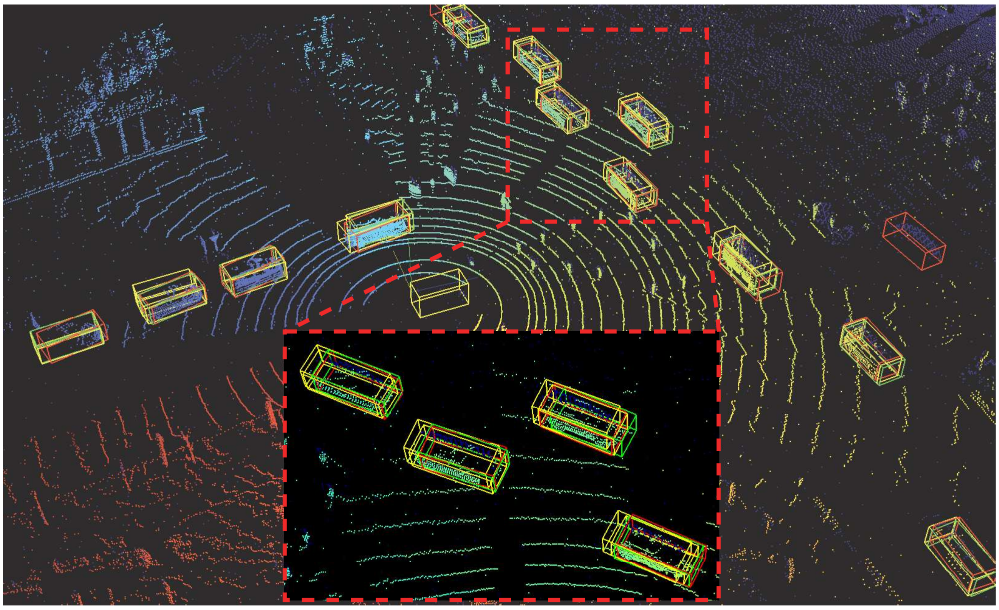
The proposed method *SparseAlign* is evaluated on the OPV2Vt and DairV2Xt datasets for TA-COOD (Yuan and Sester, 2024). Besides, it is also tested on the OPV2V (Xu et al., 2022c) and DairV2X (Yu et al., 2022) for the COOD task, in order to compare with more state-of-the-art models that are implemented on these two datasets.

#### Implementation details

All models are trained on a single GTX-4090 for 50 epochs with the Adam optimizer and batch size four. The learning rate is set to $2e-4$ with a warm-up stage of 4000 iterations. The input point clouds are augmented with random flipping along the $x$- and $y$-axis, and rotation with a random angle in range $[-90°,90°]$. For models incorporating the *TempAlign* module, Free Space Augmentation Yuan et al. (2023) (FSA, Section 6.3.2) is also used to enhance temporal context learning. Besides, four sequential frames of data are loaded during training and losses are calculated



*(a) OPV2Vt*



*(b) DairV2Xt*

*Figure 8.9: TA-COOD result of StreamLTS. Yellow bounding boxes: ego-view annotation. Green bounding boxes: time-aligned annotation. Red bounding boxes: time-aligned detection.*

solely for the last frame. To reduce GPU memory consumption, gradient calculations are only enabled for data from one ego agent and one cooperative agent. To ensure that the model generalizes well to the significant time latency introduced by communication, data from cooperative vehicles are loaded with a random latency ranging from 0 to 200 ms relative to the ego vehicle.

## 8.4 Results

### 8.4.1 Qualitative results

The TA-COOD result of *SparseAlign* is shown in Figure 8.9. The yellow bounding boxes are the ground truth bounding boxes at different observation times, and the green bounding boxes are the

*Table 8.1: TA-COOD Average Precision on OPV2Vt and DairV2Xt. AP0.5: AP at IoU threshold 0.5; AP0.7: AP at IoU threshold 0.7.*

| Method | OPV2Vt | | DairV2Xt | |
|---|---|---|---|---|
| | AP0.5↑ | AP0.7↑ | AP0.5↑ | AP0.7↑ |
| Fcooper (Chen et al., 2019a)+SA | 0.763 | 0.553 | 0.597 | 0.282 |
| FPVRCNN (Yuan et al., 2022)+SA | 0.640 | 0.474 | 0.598 | 0.307 |
| OPV2V (Xu et al., 2022c)+SA | <u>0.881</u> | 0.787 | 0.702 | 0.366 |
| StreamLTS (Yuan and Sester, 2024) | 0.853 | 0.721 | 0.640 | 0.404 |
| StreamLTS (Yuan and Sester, 2024)+SA | 0.850 | <u>0.790</u> | <u>0.757</u> | <u>0.497</u> |
| SparseAlign (full) | **0.893** | **0.818** | **0.796** | **0.548** |

globally time-aligned ground truth. As shown in the dashed red box in the middle of Figure 8.9a, the objects in this area are observed by both CAVs at different times. The top right CAV scans this area (blue points) about $0.05s$ earlier than the bottom left CAV (yellow points). The global bounding boxes are aligned to the last time point in this frame (the same as the timestamps of the red points); they moved a little forward in comparison to the yellow bounding boxes. This can also be observed in Figure 8.9b. The red bounding boxes are the predictions generated by *SparseAlign*. The result on the OPV2Vt and DairV2Xt datasets shows that the predictions have better matches to the globally time-aligned ground truth bounding boxes (green) than any locally annotated bounding boxes (yellow). This reveals that *SparseAlign* has successfully captured the time relationships between the object observations from different IAs and correctly predicted the bounding boxes at the aligned future timestamp.

### 8.4.2 Comparison to SOTA models

The framework *SparseAlign* is applied both for TA-COOD and COOD to show its general applicability and performance gain compared to the state-of-the-art models.

#### Average Precision (AP) of TA-COOD

As shown in Table 8.1, the proposed framework *SparseAlign* achieves the highest AP on both datasets. Especially at the IoU threshold of 0.7, the AP of *SparseAlign* has increased 30.7% on OPV2Vt and 6.4% on DairV2Xt compared to the second-best framework AttnFusion. The *SparseAlign* also outperforms *StreamLTS* by a great margin. By applying our three alignment modules (*StreamLTS+SA*), the performance of *StreamLTS* is also improved. In addition, the proposed framework *SparseAlign* is also compared to *Fcooper*, *FPVRCNN* and *OPV2V* method by attaching the three alignment modules to them to achieve TA-COOD. All show worse performance than the full *SparseAlign* because of worse 3D backbone or RoI selection.

Among all comparative methods, *Fcooper* has the worst performance as it uses a simpler backbone network that only consists of a PointPillar encoder and several 2D convolution layers, to encode the point cloud features. More importantly, its *Maxout* operation for fusing the features from different IAs is not learnable, hence not able to handle the spatial misalignment of the object queries caused by different observation times. In comparison, all other methods, with enhanced backbone encoding and the learnable module for multi-IA feature fusion, have much better performance. Because of the high reliability on the accuracy of the first stage detection, *FPVRCNN* might ignore some potential object points for the next processing steps, hence achieve worse performance than *OPV2V* and *StreamLTS*.

*Table 8.2: COOD Average Precision on OPV2V dataset. BW: bandwidth; AP0.5: AP at IoU threshold 0.5; AP0.7: AP at IoU threshold 0.7.*

| Method | BW(Mb) ↓ | AP0.5 ↑ | AP0.7↑ |
|---|---|---|---|
| Fcooper (Chen et al., 2019a) | 72.08 | 0.887 | 0.790 |
| V2VNet (Wang et al., 2020a) | 72.08 | 0.917 | 0.822 |
| FPVRCNN (Yuan et al., 2022) | 0.24 | 0.873 | 0.820 |
| OPV2V (Xu et al., 2022c) | 126.8 | 0.905 | 0.815 |
| CoAlign (Lu et al., 2023) | 72.08 | 0.902 | 0.833 |
| CoBEVT (Xu et al., 2022a) | 72.08 | 0.913 | 0.861 |
| V2VAM (Li et al., 2023) | 72.08 | 0.916 | 0.850 |
| V2VFormer++$^{L+C}$ (Yin et al., 2024) | 72.08 | **0.935** | **0.895** |
| SparseAlign | $<$ **1.3** | 0.930 | 0.892 |

*Table 8.3: COOD Average Precision on DairV2X dataset. BW: bandwidth; AP0.5: AP at IoU threshold 0.5; AP0.7: AP at IoU threshold 0.7.*

| Method | BW(Mb) ↓ | AP0.5 ↑ | AP0.7↑ |
|---|---|---|---|
| Fcooper (Chen et al., 2019a) | 48.8 | 0.734 | 0.559 |
| V2VNet (Wang et al., 2020a) | 48.8 | 0.664 | 0.402 |
| FPVRCNN (Yuan et al., 2022) | 0.24 | 0.665 | 0.505 |
| V2XViT(Xu et al., 2022b) | 48.8 | 0.704 | 0.531 |
| DiscoNet(Xu et al., 2022a) | 48.8 | 0.736 | 0.583 |
| OPV2V (Xu et al., 2022c) | 97.6 | 0.733 | 0.553 |
| CoAlign (Lu et al., 2023) | 48.8 | 0.746 | 0.604 |
| DI-V2X (Xiang et al., 2024) | 48.8 | 0.788 | 0.662 |
| SparseAlign | $<$ **1.3** | **0.845** | **0.685** |

**Average Precision (AP) of COOD**

To assess the overall efficacy of the *SparseAlign* design, the proposed model is compared with state-of-the-art models on standardized benchmarks. Table 8.2 presents the COOD results on the OPV2V dataset. All methods use only LiDAR data, except for *V2VFormer++* (denoted by L+C), which incorporates both LiDAR and camera data. Compared to the state-of-the-art, the proposed *SparseAlign* achieves the best performance in LiDAR-based COOD, approaching the results of *V2VFormer++*, which benefits from the LiDAR-Camera fusion. Table 8.3 shows the results on the DairV2X dataset. Here, *SparseAlign* approach outperforms all other methods by a large margin, achieving a 2.3% improvement in AP at the IoU threshold of 0.7 and a 5.7% improvement at the IoU threshold of 0.5. This is mainly due to the enhanced temporal and global reasoning ability of the model for detecting distant and large objects. Compared to other methods that rely on dense BEV feature maps, *SparseAlign* consumes the least communication bandwidth (BW) of less than 1.3MB (without compression) thanks to efficient query-based operations.

### 8.4.3 Training efficiency

Less memory consumption and training time is crucial for faster model development on limited computing resources. To investigate the training efficiency of *SparseAlign*, it is compared to other models with the peak memory usage because it is crucial for determining the devices the model can be trained on and the training batch size on limited GPU resources. In addition, the epochal training time of all comparative models on the same device is tested. The results of AP against

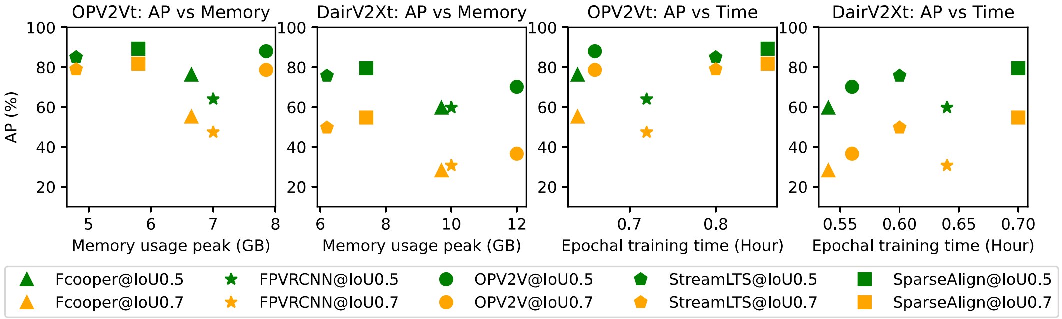


*Figure 8.10: Comparison of memory usage peak and epochal training time.*

*Table 8.4: Ablation study on feature learning modules: 2D and 3D CEC layers, TempAlign Module (TAM), and Dilation Convolution layers (Dil.)*

| CEC 2D | CEC 3D | TAM | Dil. | OPV2V AP0.5 | OPV2V AP0.7 | DairV2X AP0.5 | DairV2X AP0.7 |
|---|---|---|---|---|---|---|---|
| | | | 2 | 0.890 | 0.827 | 0.707 | 0.637 |
| | | | 1 | 0.924 | 0.885 | 0.773 | 0.638 |
| ✓ | | | 1 | 0.940 | 0.901 | 0.796 | 0.662 |
| ✓ | ✓ | | 1 | 0.950 | 0.914 | 0.813 | 0.682 |
| ✓ | ✓ | ✓ | 1 | **0.951** | **0.929** | **0.845** | **0.685** |

these two measures are demonstrated in fig. 8.10. As shown in the left two sub-plots, *SparseAlign* (squares) requires less training memory resource than the dense models (Fcooper, OPV2V) and has the best overall TA-COOD performance. The model *OPV2V* (baseline model proposed in the dataset release of OPV2V (Xu et al., 2022c)) has a comparable detection performance; however, its GPU memory demand is much higher than *SparseAlign*. Regarding the epochal training time, *SparseAlign* seems less efficient. However, thanks to the low memory usage, *SparseAlign* can reduce the training time by increasing the batch size.

### 8.4.4 Ablation study

The ablation study is conducted on the configurations of the 3D backbone and *TAM*. The results are shown in Table 8.4. By changing the dilation size of the first convolution layer in each block of *SUNet* from one to two, the performance is deteriorated (Table 8.4, row1 vs. row2), revealing that dilation convolution, in this case, does not effectively increase the receptive field of the convolution and mitigate the ICF issue, but hurts the local feature learning. Instead, the performance is improving gradually by replacing some of the standard sparse convolutional layers with CEC layers. At last, the *TAM* further enhanced the APs by looking for clues in the historical frames.

In Table 8.5, *CompassRose* direction encoding is compared with three conventional methods: directly regress the ground-truth angle (*gt-angle*), second-style Yan et al. (2018) directional encoding (*second*), and the sine-cosine encoding (*sin_ cos*). For simplicity, the module configurations in the second row of Table 8.4 is used for the comparison. The results show that *CompassRose* has the best performance.

### 8.4.5 Communication latency and sensor asynchrony

Table 8.6 and Table 8.7 show the results of TA-COOD on the OPV2Vt and the DairV2Xt dataset. The proposed *SparseAlign* demonstrates a significant performance improvement over the baseline

*Table 8.5: Comparison study on target encoding of bounding box angle: ground-truth angle (gt-angle), second-style (second), sine-cosine (sin_cos) and compass rose (compass).*

| Dir. Enc. | OPV2V | | DairV2X | |
|---|---|---|---|---|
| | AP0.5 | AP0.7 | AP0.5 | AP0.7 |
| *gt-angle* | 0.871 | 0.768 | 0.620 | 0.427 |
| *second* | 0.908 | 0.874 | 0.743 | 0.590 |
| *sin_cos* | 0.915 | 0.881 | 0.748 | 0.603 |
| *compass* | **0.924** | **0.885** | **0.773** | **0.638** |

*Table 8.6: OPV2Vt: Average Precision of TA-COOD with communication latency. SA no TA, PT, TL, FSA: SparseAlign without TempAlign, Point-wise Timestamps, Train Latency and Free Space Augmentation, respectively.*

| Latency | 0ms | | 100ms | | 200ms | |
|---|---|---|---|---|---|---|
| AP threshold | 0.5 | 0.7 | 0.5 | 0.7 | 0.5 | 0.7 |
| SteamLTS | 0.853 | 0.721 | 0.816 | 0.680 | 0.787 | 0.647 |
| SA no TA | 0.890 | 0.802 | 0.856 | 0.471 | 0.631 | 0.209 |
| SA no PT | 0.835 | 0.615 | 0.831 | 0.625 | 0.815 | 0.581 |
| SA no TL | 0.885 | **0.843** | 0.862 | 0.619 | 0.723 | 0.279 |
| SA no FSA | 0.881 | 0.816 | 0.875 | **0.795** | 0.857 | 0.762 |
| Full SA | **0.893** | 0.818 | **0.885** | **0.795** | **0.867** | **0.765** |

method, *StreamLTS* (Yuan and Sester, 2024), with regard to the communication latency and sensor asynchrony. Specifically, the AP at an IoU threshold of 0.7 increased by 9.7% on the OPV2Vt dataset and by 14.4% on the DairV2Xt dataset. As the communication latency increases to 200 ms, the AP at IoU 0.5 of *SparseAlign* only dropped 2.6%, while the baseline method *StreamLTS* has lost 6.6% of its accuracy. This reveals that *SparseAlign* is more robust against communication latency. To investigate modules or configurations that are making an effect on this robustness, an ablation study is conducted by removing the *TAM*, Point-wise Timestamps (TP), Training Latency (TL, communication latency during training) and Free Space Augmentation (FSA), respectively.

Without the TAM, the performance noticeably declines as latency increases, despite achieving comparable results to the full *SpareAlign* model at 0 ms latency. A similar effect is observed when the full SA model is trained without exposure to data with communication latency (SA without TL). Remarkably, without TL, the model outperforms the full *SpareAlign* model in terms of AP at an IoU threshold of 0.7 on the OPV2Vt dataset (0.843 against 0.818). This is because the model focuses more on predicting bounding boxes within the same frame, where only minimal latency is introduced by asynchronous sensors.

Without PT, the model performs much worse than the full *SpareAlign*, showing the importance of fine-grained point-wise timestamps in the point clouds for the model to compensate for the errors introduced by sensor asynchrony and to capture the accurate temporal context for prediction. Finally, the ablation study on the FSA module also shows a positive influence on the model's performance. For example, AP at the IoU threshold of 0.7 increased by 4% on DairV2Xt by utilizing the FSA module. This improvement can also be observed in Figure 8.11. The free space points (blue) also contain the timestamps calculated according to their angles in the polar coordinate system. This enhances the temporal context for learning, especially in distant areas where the scanned observation points are very sparse.

*Table 8.7: DairV2Xt: Average Precision of TA-COOD with communication latency. SA no TA, PT, TL, FSA: SparseAlign without TempAlign, Point-wise Timestamps, Train Latency and Free Space Augmentation, respectively.*

| Latency | 0ms | | 100ms | | 200ms | |
|---|---|---|---|---|---|---|
| AP threshold | 0.5 | 0.7 | 0.5 | 0.7 | 0.5 | 0.7 |
| SteamLTS | 0.642 | 0.404 | 0.613 | 0.379 | 0.590 | 0.364 |
| SA no TA | 0.720 | 0.457 | 0.672 | 0.393 | 0.635 | 0.350 |
| SA no PT | 0.760 | 0.469 | 0.737 | 0.459 | 0.722 | 0.449 |
| SA no TL | 0.793 | 0.534 | 0.749 | 0.474 | 0.698 | 0.423 |
| SA no FSA | 0.769 | 0.508 | 0.755 | 0.502 | 0.740 | 0.496 |
| Full SA | **0.796** | **0.548** | **0.786** | **0.543** | **0.772** | **0.532** |

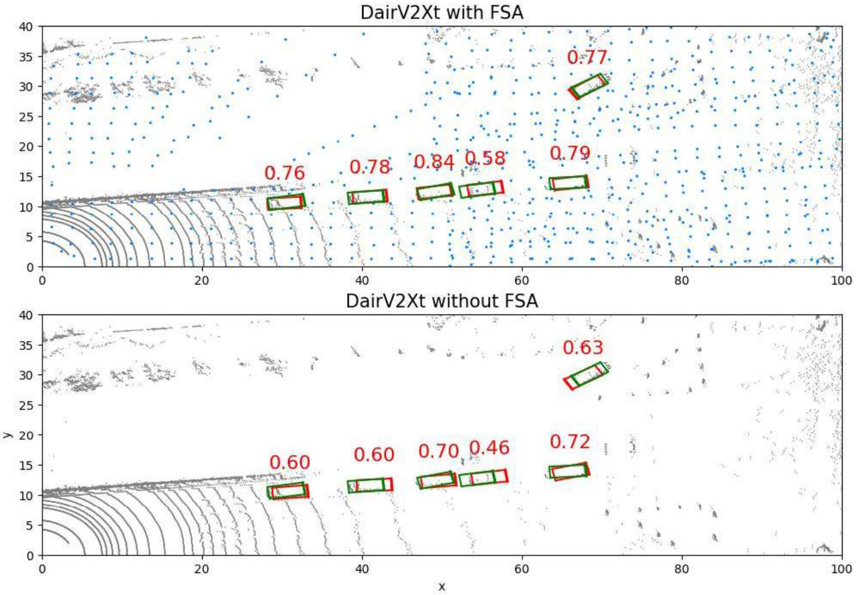


*Figure 8.11: SparseAlign performance with and without FSA. The FSA points are in blue. Red texts are IoUs between the detected (red) and the ground-truth (green) bounding boxes.*

### 8.4.6 Localization errors

Figure 8.12 shows the performance of *SpareAlign* against different localization errors. Random localization errors are added to the poses of both ego ($T_e$) and cooperative ($T_c$) vehicles. The translation errors $x_\epsilon$,$y_\epsilon$ along the $x$- and $y$-axis and the rotation error $r_\epsilon$ around the $z$-axis are all assumed to be normally distributed with $\mathcal{N}(0,1) \cdot \epsilon$, where $\epsilon \in [0,1]$ is the error scaling factor. Results are reported by gradually increasing $\epsilon$ with a step size of 0.2. Note that the translation and rotation errors exist in the poses of both the ego and the cooperative vehicle, the resulting relative translation error of the transformation $T_c^e = inverse(T_e) \cdot T_c$ from cooperative to ego coordinate system will be amplified as the rotation error increases. The *PAM* is specially designed to mitigate the influence of large relative errors between the ego and the cooperative vehicle.

The proposed *PAM* and the baseline method *CoAlign* Lu et al. (2023) both significantly reduce the impact of pose errors on AP performance compared to the configuration lacking any correction module. However, *PAM* demonstrates greater robustness against large errors. For instance, on the OPV2V dataset, the AP for *CoAlign* decreased by approximately 27% at the error of $1.0m$, $1.0m$,$1.0°$, whereas *PAM* experienced only a 12% drop. This validates the efficacy of the proposed method, which relies on the pose-agnostic relative neighborhood geometries to match the detected

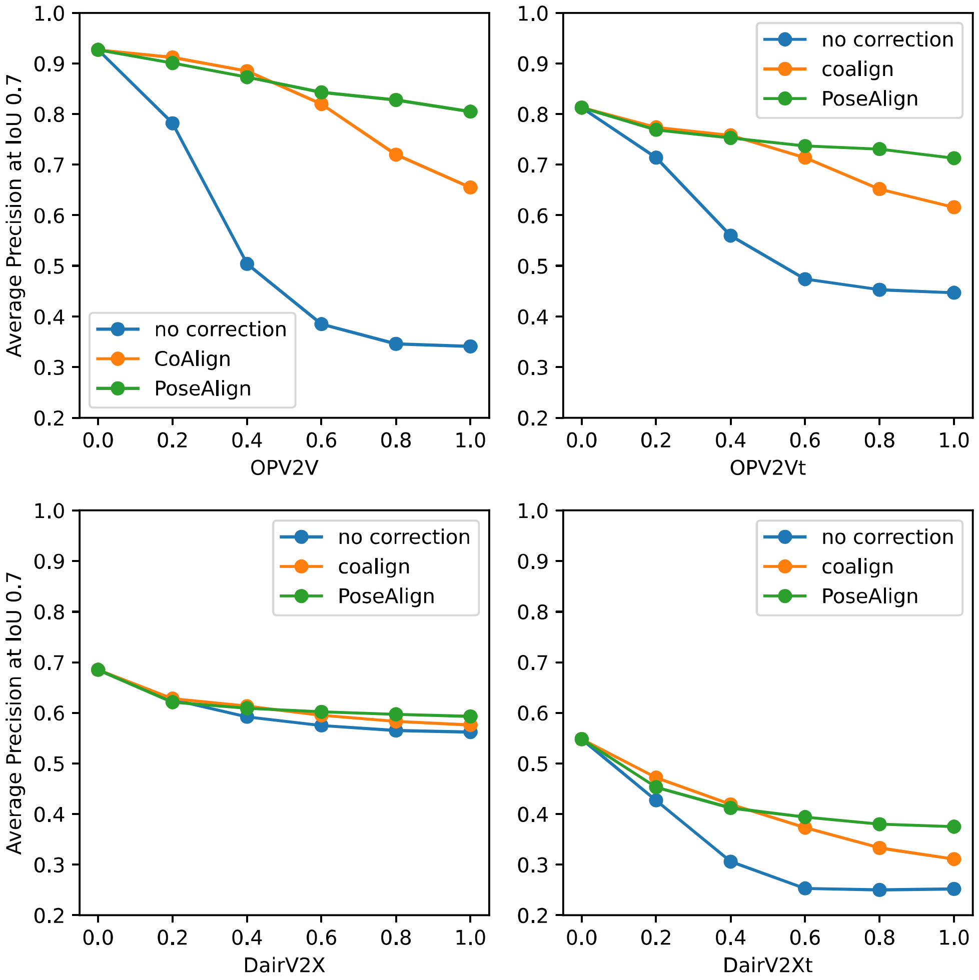


*Figure 8.12: AP at IoU threshold of 0.7 with translation errors ranging from 0m to 1m along x- and y-axis, and rotation errors from* 0° *to* 1.0° *(horizontal axis) for the different datasets.*

bounding boxes of the ego and the cooperative IAs. Note that the DairV2X dataset exhibits smaller changes in AP as localization errors increase, because the pose parameters are not well calibrated. In contrast, Yuan and Sester (2024) refined the poses of DairV2X for generating the DairV2Xt dataset, making the better-calibrated DairV2Xt more sensitive to newly introduced errors.

### 8.4.7 CPM sizes

The proposed *SparseAlign* significantly reduces communication bandwidth without relying on computationally intensive dense feature maps, unlike state-of-the-art methods. This efficiency is achieved by selecting only the top $K = 1024$ object queries for further processing. In addition to this intrinsic reduction, the size of the CPMs can be further minimized through score-based selection. Specifically, the query-based detection scores from the *LQDet* head are employed to identify and share only the most critical queries. As Figure 8.13 shows, using a CPM score threshold below 0.5 does not lead to a noticeable degradation in performance across all datasets. Remarkably, at a threshold score of 0.5, the average size of the CPMs is reduced to less than 400 KB on all datasets. The final CPM sizes after selection are influenced by the driving scenes. Specifically, scenes with more vehicles tend to have larger CPM sizes. Consequently, the OPV2V and OPV2Vt datasets,

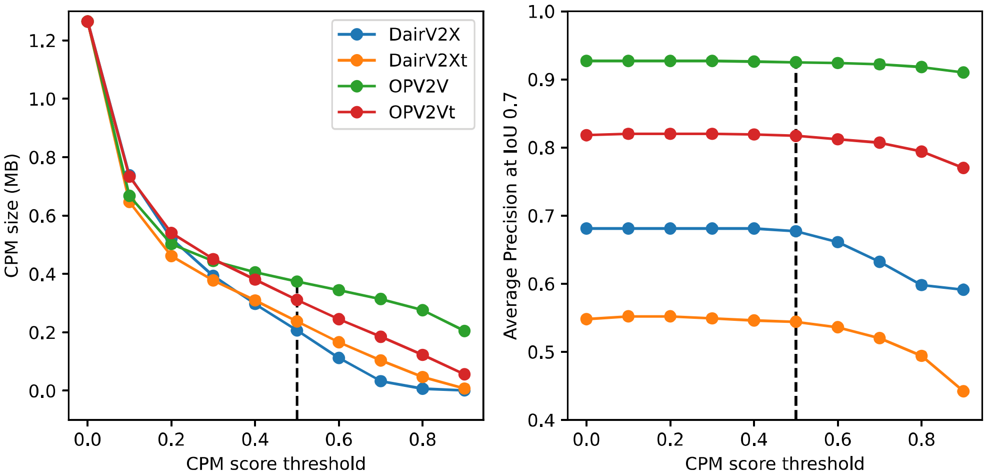


*Figure 8.13: CPM sizes with different information selection scores. Object query features with detection scores larger than the threshold (x-axis) are shared as CPMs.*

which contain more vehicles in their scenes, result in larger CPM sizes compared to the DairV2X and DairV2Xt datasets.

## 8.5 Summary

To address sensor asynchrony, this chapter proposed a new benchmark, *Time-Aligned Cooperative Object Detection (TA-COOD)*,which processes asynchronized data to detect and predict object bounding boxes at a given timestamp. To tackle this challenge, an efficient fully sparse framework *SparseAlign* is proposed. This framework integrates a novel fully sparse 3D backbone network *SUNet* and three alignment modules: temporal, pose, and spatial alignment. SparseAlign significantly outperforms state-of-the-art methods on the OPV2V and DairV2X datasets for COOD, as well as on their variants OPV2Vt and DairV2Xt for time-aligned COOD (TA-COOD). This superior performance stems from its innovative design. The SUNet backbone extracts more robust features than the baseline MinkUnet used by StreamLTS. The temporal alignment module learns more accurate temporal context from the pointwise timestamps of the sequential point clouds. The pose alignment module *PAM* demonstrates improved robustness to large localization errors compared to the baseline method CoAlign. The spatial alignment module demonstrates resilience to spatial perturbations caused by transformations between different IA coordinate systems. Moreover, SparseAlign's fully sparse design enables it to consume less computational resources and reduces communication bandwidth requirements for data sharing between IAs.

# 9 Summary and Conclusion

Motivated by the benefits of collective perception in expanding the field-of-view and reducing occlusions for IVs, this work aims to design neural network-based frameworks for collective perception. Within this scope, object detection and BEV semantic segmentation are chosen as the primary tasks, as they are essential for autonomous driving systems to avoid collisions and navigate safely and efficiently. For both tasks, the proposed frameworks only share the most important information that is selected by the confidence scores of the classification or the uncertainty learned from neural networks in order to balance between the information gain and communication resource consumption. The frameworks developed in this work address three key issues often neglected in previous studies: data processing efficiency (Chapter 6), the uncertainty estimation for BEV semantic segmentation results (Chapter 7), and sensor asynchrony (Chapter 8).

### Data processing efficiency

As the number of IAs increases, the scale of the data also increases significantly, especially when sequential data should be processed to model temporal information. To increase the efficiency and accelerate the model development process, this work designed a highly efficient framework development tool *CoSense3D* for collective perception. It modularizes the collective perception pipeline into four parts: *IA Manager* manages the working logic of individual IAs, *Data Manager* manages the data distribution to and gathering from the IAs, *Task Manager* summarizes and batches the tasks that are scheduled by the IA manager. *Forward Runner* contains all the shared deep learning models and runs the tasks given by the *Task Manager*. Over these four modules, the *Central Controller* plans the holistic working pipeline by calling these modules to achieve specific tasks. Based on the clear management of data, task distribution and gathering, and the prototyping for the behaviour of individual IAs, this work proposed an agent-based training strategy – only calculate gradients for specific agent – to increase the training efficiency and reduce the GPU memory consumption without noticeable performance drop with regard to accuracy.

The comparative object detection experiments with full gradient calculation for all IAs versus a reduced number of IAs for gradient calculation have shown the training efficiency and performance with the *CoSense3D* tool. The result shows that reducing the number of CAVs for gradient calculation can significantly save GPU memory and training time without a noticeable performance drop, if an appropriate fusion module without dropping learnable features is used. On the OPV2V dataset, the training resources needed have been reduced by more than half for the best-performing model *AttnFusion*. In addition, F-Cooper with Maxout fusion and EviBEV with Naive mix-up fusion have shown that non-learnable fusion operations might reduce performance if agent-based training is used.

### Uncertainty estimation for BEV semantic segmentation

Interpreting the driving environment with BEV semantic segmentation is beneficial for autonomous driving systems to build local real-time dynamic maps for route planning. The accuracy and trustworthiness of the semantic segmentation results directly influence the safety of the planned routes. Although some areas of BEV maps might be occluded, previous works conduct predictions over these unobserved areas according to the distribution of the historical training data. These predictions might lead to dangerous driving behaviors because dynamic objects might exist in the unobserved areas, and the model might even classify these areas as empty drivable road with high probability. To generate more reliable BEV maps, this work proposes a novel method to interpret driving environments with observable probabilistic BEV maps. These maps interpret the LiDAR sensory data in a back-traceable manner so that each prediction is supported by evidences provided by the original observation points. More specifically, an evidential-learning-based probabilistic classification framework *GevBEV* is built to generate statistics for this interpretation. This model assumes a spatial Gaussian distribution for each voxel of a predefined resolution so that any point in the continuous driving space can draw itself a Dirichlet distribution of the classification based on the evidences drawn from the spatial Gaussian distributions of the neighboring voxels.

The proposed *GevBEV* framework is tested on benchmarks OPV2V and V2V4Real of BEV map interpretation for cooperative perception in simulated and real-world driving scenarios, respectively. The experiments show that *GevBEV* outperforms the baseline of the image-based BEV map by a large margin. By analyzing the predictive uncertainty, it is proved that evidential classification can score the classification result in a less overconfident and better-calibrated manner than the deterministic counterpart of the same model. Furthermore, the spatial Gaussian distribution assigned to each observable point was also proven beneficial in closing the gaps of sparse point clouds with a controllable range and smoothing the BEV maps. By virtue of this spatial distribution, one can draw the Dirichlet classification result for any point in the continuous driving space. This probabilistic result can be used to make safer decisions for autonomous driving by its ability to quantify the uncertainty using the measurement evidences. Besides, the estimated uncertainty can be used to reduce the CPM data size for collective perception. For example, by selecting the evidences that have uncertainty lower than 0.5, the CPM average size on both OPV2V and V2VReal datasets can be reduced by more than 80% with only less than 4% performance drop.

### Sensor asynchrony

Sensor asynchrony can lead to large displacement of objects at high speed (*e.g.,* 60km/h leads to 1.6m displacement) and introduce inaccuracy into the data fusion process of collective perception. Previous works assume the data in one frame are all synchronized and ignore the simple-to-obtain but informative point-wise timestamps of point clouds. This will lead to inaccurate temporal modeling of dynamic objects. To alleviate this problem, this work utilized point-wise timestamps to model the temporal features of objects and proposes a new benchmark called Time-Aligned Cooperative Object Detection (*TA-COOD*) which takes the point clouds and their point-wise timestamps as input and predicts the bounding boxes at the globally aligned timestamps. To validate the importance of point-wise timestamps, this work built a query-based fully sparse framework *SparseAlign* to model the temporal feature context from the sequential point cloud data so as to predict the object location at the aligned global time. With this design, the SparseAlign significantly outperforms the state-of-the-art models, confirming the efficacy of modeling the precise temporal context with the point-wise timestamps. Besides, SparseAlign is also more efficient on training resource consumption than the other models thanks to its fully sparse operations. Through the ablation study, the proposed 3D backbone network *SUNet* showed strong ability in extracting more effective and robust features from the point clouds for further processing. Besides, the ablation study on the encoding method for BBox angle regression also validated that the proposed *Compass Rose* encoding has the best performance in comparison to the encoding used in the previous works. The ablation study on the time-related module *TAM* validated the great importance of point-wise timestamps on capturing the accurate temporal context for predicting the locations of objects at future timestamps.

# 10 Outlook

In this work, two frameworks, *GevBEV* and *SparseAlign*, were developed and trained for BEV semantic segmentation and collective object detection, separately. They both utilize fully sparse operations to enhance model efficiency and share a similar UNet-like point cloud encoding backbone network, making it straightforward to merge these two frameworks into a single multi-task model. However, this would increase the model size and GPU memory requirements during the training process, as more learning heads would result in more learning targets and gradient calculations. Due to limited training resources, training this unified framework is left for future work.

Aside from sensor asynchrony, communication latency and efficiency, localization errors also pose a significant challenge for collective perception. This work tested the proposed frameworks by introducing random localization errors and observed a performance drop of up to approximately 10% on both the BEV semantic segmentation task and the TA-COOD task. In future work, improved algorithms should be explored to mitigate the impact of localization errors on the overall perception performance. For example, the predicted bounding boxes in the previous frames may facilitate more effective inter-agent bounding box matching.

Additionally, this work evaluates collective perception performance with respect to the ego vehicle and a fixed number of cooperative agents in each scenario. To enhance communication efficiency, future research could investigate the overall perception performance of all vehicles within the scenario by optimizing the sharing topology among agents. This would improve the efficiency of data exchange. For example, a dynamic communication scenario, where the number and identity of IAs change as they move, introduces a more complex communication topology. If every IA—not just a single ego vehicle—aims to optimize its perception, a new algorithm could be developed to jointly optimize the data-sharing process. Such an algorithm would determine who should share data, when sharing should occur, and how data should be exchanged, ensuring efficient and effective communication.

Lastly, it is worth mentioning that collective perception frameworks may also face the conventional challenges associated with deep neural networks. For example, adversarial attacks from cooperative vehicles could introduce malicious data into the ego vehicle's system, potentially causing the collapse of the ego perception system and leading to serious traffic accidents. Another concern lies in the shared Collective Perception Messages (CPMs), which may exhibit domain gaps due to variations in perception models and training datasets used by different manufacturers. Although the CPM format could be standardized, the learned features in the CPM might not be mutually understandable by different models. Therefore, exploring domain adaptation technologies could be crucial in addressing this issue.

# List of Figures

1.1 Collective perception scenario[1] . . . 16

2.1 CPM sharing strategy. . . . 29

3.1 Multi-Layer Perceptron . . . 32
3.2 An example of a two-dimensional convolution . . . 32
3.3 Variants of convolution kernels. . . . 33
3.4 Atomic operations of convolutions . . . 34
3.5 Different output variants of sparse convolutions . . . 34
3.6 Attention mechanism for retrieving information from an animal database[2] . . . 35
3.7 The overall architecture of transformer . . . 37
3.8 The architecture of VoxelNet . . . 39
3.9 The encoded pillar vectors of PointPillar network . . . 40
3.10 The overall architecture of PointPillar network . . . 40
3.11 The architecture of SpConv 3D backbone network . . . 41
3.12 The architecture of MinkUnet . . . 42
3.13 Attention fusion. Green, blue and yellow feature maps are from $N = 3$ different IAs. . . . 43
3.14 3D object detection . . . 44
3.15 Anchors for regression target encoding . . . 44
3.16 Semantic segmentation of 2D images (Geiger et al., 2012) . . . 45
3.17 Semantic segmentation of 3D points . . . 45
3.18 Semantic segmentation of a BEV map . . . 45
3.19 Dirichlet distribution . . . 48
3.20 An example scene for optimizing the LiDAR poses while scanning the arena. . . . 50
3.21 AP: **A**rea **U**nder Recall-Precision **C**urve (AUC). The green areas (A1, A2, A3) are utilized to ensure that the AP metric is less sensitive to minor variations in the ranking of P-R pairs that define the P-R curve. The interpolated precision for these green regions is denoted as $p_{interp,*}$. . . . 52
3.22 Intersection and Union of two bounding boxes. The resulting intersection and union are illustrated with the yellow region. . . . 52
3.23 Intersection over Union (IoU) for BEV semantic segmentation. . . . 54
3.24 Examples of calibration plot . . . 55

5.1 OPV2V dataset: cyan and purple points are the LiDAR data of the ego and cooperative vehicle, repectivly. Green boxes are the ground-truth annotations of the vehicular participants in the scenario. . . . 65
5.2 OPV2Vt dataset: The points scanned in one frame are observed at different timestamps (blue to red scaled colors). The objects observed by two sensors are captured at different times, resulting in unaligned ground truth bounding boxes (yellow boxes). The bounding boxes at the last time point of this frame are taken as the final ground truth for the time-aligned cooperative object detection (green boxes). . . . 66

5.3 V2Vreal dataset: LiDAR data with 3D bounding box annotations . . . . . . . . . . . . . 67
5.4 DairV2X dataset: LiDAR data with 3D bounding box annotations (green). Magenta points: point cloud from vehicle. Cyan points: point cloud from infrastructure. . . . . . . . . . . 68
5.5 DairV2Xt dataset: The points scanned in one frame are observed at different timestamps (blue to red scaled colors). The objects observed by two sensors are captured at different times, resulting in unaligned ground truth bounding boxes (yellow boxes). The bounding boxes at the last time point of this frame are taken as the final ground truth for the time-aligned cooperative object detection (green boxes). . . . . . . . . . . . . . . . . . . . . 69

6.1 Collective perception formalization in two communication modes. Left: handshaking communication. Right: broadcasting communication. . . . . . . . . . . . . . . . . . . . . . . . 73
6.2 CoSense3D: Agent-based training framework. Black errors indicate the instruction passing direction, green arrows indicate the data passing direction. . . . . . . . . . . . . . . . . . 74
6.3 CoSense3D GUI. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
6.4 CoSense3D GUI OutputViwer. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
6.5 Workflow of CoSense3D Central Controller. . . . . . . . . . . . . . . . . . . . . . . . . . 77
6.6 Overview of the pipeline for shared modules . . . . . . . . . . . . . . . . . . . . . . . . 79
6.7 Sampling free space points. The red point is the origin of the LiDAR coordinate system, $z$-axis indicates the vertical direction in the driving space, $x$-axis indicates the horizontal direction. $f_{s0}$ is the intersection point of the LiDAR ray (red dashed line) and the ground. $f_{s1}$,$f_{s2}$, and $f_{s3}$ are the sampled free space points belonging to the ray path. . . . . . . . 80
6.8 Compass rose encoding for orientation of bounding box . . . . . . . . . . . . . . . . . . 82
6.9 Anchor assignment to fore- and background class. GT box in green, anchor box in yellow. . 83
6.10 Bounding box Center Assigner. Box with dished lines represents the ground-truth bounding box, white point is the center of this box and the orange points that lies in the red circle is assigned as positive (foreground) object sample. . . . . . . . . . . . . . . . . . . . . . 84
6.11 Random sampling of continuous BEV Assigner. Left: global cell-based down-sampling, right: buffer-based sampling around vehicle (yellow). . . . . . . . . . . . . . . . . . . . . . . . 85
6.12 Relation between object detection performance and GPU memory usage (left) as well as training time (right) with different gradient configurations. . . . . . . . . . . . . . . . . . 88

7.1 Left: Gaussian-based Evidential BEV Semantic Segmentation. Red to cyan rings represent Gaussian density from high to low. Right: BEV maps of the binary classification for vehicle and road, respectively. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
7.2 Overall model architecture of BEV Semseg . . . . . . . . . . . . . . . . . . . . . . . . . 93
7.3 Comparison of three BEV semantic segmentation heads: the *GevBEV Head* (proposed in this work), the *EviBEV Head* (a modified version without the Gaussian distribution), and the *BEV Head* (a simplified version without both Gaussian and Dirichlet distributions). . . 94
7.4 An example of the information shared between CAVs. The ego CAV and its request mask with high uncertainty areas indicated in red color, and the cooperative CAV and its response mask with low uncertainty indicated in green color. In the uncertainty maps, light color indicates low uncertainty and black color indicates no measurements. . . . . . . . . . . . 95
7.5 Areas considered for selecting information for CPMs. Grey area ($CPM^{all}$): overall area in the perception range (*e.g.,* $[-50{,}50]m$). Green area ($CPM^{road}$): drivable road area. . . . . 96
7.6 An example of BEV semantic segmentation result for IoU calculation under uncertainty. . 99

7.7 Results of the GevBEV. The first row show the semantic segmentation result. The bottom sub-figure shows more detailed detection results in a larger extent. Specifically, the original input point cloud is denoted by blue points, road points are highlighted by a light color if their uncertainties are under the threshold of 0.7, and the objects are shown in bounding boxes with red color indicating the detection, and with green color the corresponding ground truth, respectively. The thick bar in the front of the bounding boxes denotes the driving direction. Moreover, the bounding boxes are filled with the point confidences generated by dynamic (object) head of GevBEV, where magenta points are associated with high confidence, while cyan indicates the opposite. . . . . . 101
7.8 The comparison of classification confidences between GevBEV and CoBEVT on the OPV2V dataset. . . . . . 102
7.9 The comparison of classification confidences between GevBEV and CoBEVT on the V2VReal dataset. . . . . . 102
7.10 Object point distribution. The dashed line bounding boxes denote the detection, and the solid line bounding boxes denote the corresponding ground truth. The thick bar of each bounding box denotes the driving direction. The statistics under each sub-figure denote the average evidence and [JIoU, IoU]. . . . . . 103
7.11 OPV2V road segmentation result ($IoU^{dense}$) with localization noise. . . . . . 105
7.12 OPV2V object segmentation result ($IoU^{dense}$) with localization noise. . . . . . 106
7.13 V2V4Real object segmentation result ($IoU^{dense}$) with localization noise. . . . . . 106
7.14 Calibration plots by different models. The perfect calibration line is indicated by the diagonal dashed line. . . . . . 108
7.15 Comparison of different CPM sharing strategies ($CPM^{all}$ and $CPM^{road}$, ref. Section 7.2.4) for co-perception. The first row shows the results of the road head (greenish) and the second the objects head (orangish). . . . . . 109

8.1 TA-COOD with two CAVs. Dashed boxes: Ground truth bounding box of ego- and coop-CAV observations. Solid box: time-aligned Ground Truth bounding box. . . . . . 111
8.2 Overview of of the SparseAlign pipeline for TA-COOD, including the sparse SUNet as backbone network and three alignment modules for temporal (TempALign), pose (PoseAlign) and spatial (SpatialAlign) data alignment. . . . . . 112
8.3 Issues of sparse convolutional backbone networks. a) Center Feature Missing (CFM). b,c) Isolated Convolution Field (ICF) caused by ring disconnectivity and occlusion. . . . . . 113
8.4 The architecture of the proposed 3D backbone network SUNet (Sparse UNet). . . . . . 113
8.5 Temporal alignment module (TAM) . . . . . . 114
8.6 MTA alignment: *Motion-Time-Aware* embeds the query positions ($Q_p$) and *Motion-Aware* embeds the query context $Q_c$ and initial queries ($tgt$). . . . . . 114
8.7 Pose alignment module (PAM) . . . . . . 115
8.8 Spatial alignment module (SAM) . . . . . . 116
8.9 TA-COOD result of StreamLTS. Yellow bounding boxes: ego-view annotation. Green bounding boxes: time-aligned annotation. Red bounding boxes: time-aligned detection. . . . . . 118
8.10 Comparison of memory usage peak and epochal training time. . . . . . 121
8.11 SparseAlign performance with and without FSA. The FSA points are in blue. Red texts are IoUs between the detected (red) and the ground-truth (green) bounding boxes. . . . . . 123
8.12 AP at IoU threshold of 0.7 with translation errors ranging from 0m to 1m along x- and y-axis, and rotation errors from 0° to 1.0° (horizontal axis) for the different datasets. . . . . . 124

8.13 CPM sizes with different information selection scores. Object query features with detection scores larger than the threshold (x-axis) are shared as CPMs. . . . . . . . . . . . . . . . . 125

# List of Tables

1.1 Levels of Automation in Autonomous Driving (SAE J3016 SAE International (2021)) . . . 17

2.1 VANET communication technologies classified with communication ranges. . . . . . . . . 26
2.2 Definition of different data processing levels . . . . . . . . . . . . . . . . . . . . . . . . 28

6.1 Average precision of object detection on OPV2V benchmark with different gradient configurations. AP@*iou*: average precision at IoU threshold *iou*. $\text{AP}^{-}$@*iou*: the AP drop of reduced gradient computation in comparison to the full gradient computation. . . . . . . . . . . 87

7.1 Comparison with the state-of-the-art models on OPV2V and V2V4real dataset. Best values are highlighted in boldface and the second best values are underlined. . . . . . . . . . . 105
7.2 Ablation study of the proposed modules. All IoU results are measured in percent. Best values are highlighted in boldface. G.s.: point-based spatial Gaussian, $\mathcal{L}_{edl}$: evidential loss, FSA: free space augmentation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107
7.3 Average Calibration Error. Best values are highlighted in boldface and the second best values are underlined. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108

8.1 TA-COOD Average Precision on OPV2Vt and DairV2Xt. AP0.5: AP at IoU threshold 0.5; AP0.7: AP at IoU threshold 0.7. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 119
8.2 COOD Average Precision on OPV2V dataset. BW: bandwidth; AP0.5: AP at IoU threshold 0.5; AP0.7: AP at IoU threshold 0.7. . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
8.3 COOD Average Precision on DairV2X dataset. BW: bandwidth; AP0.5: AP at IoU threshold 0.5; AP0.7: AP at IoU threshold 0.7. . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
8.4 Ablation study on feature learning modules: 2D and 3D CEC layers, TempAlign Module (TAM), and Dilation Convolution layers (Dil.) . . . . . . . . . . . . . . . . . . . . . . . 121
8.5 Comparison study on target encoding of bounding box angle: ground-truth angle (*gt-angle*), second-style (*second*), sine-cosine (*sin_ cos*) and compass rose (*compass*). . . . . . . . . 122
8.6 OPV2Vt: Average Precision of TA-COOD with communication latency. SA no TA, PT, TL, FSA: *SparseAlign* without TempAlign, Point-wise Timestamps, Train Latency and Free Space Augmentation, respectively. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 122
8.7 DairV2Xt: Average Precision of TA-COOD with communication latency. SA no TA, PT, TL, FSA: *SparseAlign* without TempAlign, Point-wise Timestamps, Train Latency and Free Space Augmentation, respectively. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123

# Bibliography

Abboud, K., Omar, H.A., Zhuang, W., 2016. Interworking of DSRC and Cellular Network Technologies for V2X Communications: A Survey. IEEE Transactions on Vehicular Technology 65, 9457–9470.

Ahangar, M.N., Ahmed, Q.Z., Khan, F.A., Hafeez, M., 2021. A Survey of Autonomous Vehicles: Enabling Communication Technologies and Challenges. Sensors 21, 706.

Ahmed, Q.Z., Park, K.H., Alouini, M.S., 2015. Ultrawide Bandwidth Receiver Based on a Multivariate Generalized Gaussian Distribution. IEEE Transactions on Wireless Communications 14, 1903–1915.

Allig, C., Leinmüller, T., Mittal, P., Wanielik, G., 2019. Trustworthiness Estimation of Entities within Collective Perception, in: IEEE Vehicular Networking Conference (VNC), pp. 1–8.

Aoki, S., Higuchi, T., Altintas, O., 2020. Cooperative Perception with Deep Reinforcement Learning for Connected Vehicles, in: IEEE Intelligent Vehicles Symposium (IV), pp. 328–334.

Arena, F., Pau, G., 2019. An Overview of Vehicular Communications. Future Internet 11, 27.

Ba, J.L., Kiros, J.R., Hinton, G.E., 2016. Layer Normalization, *ArXiv* preprint ArXiv:1607.06450 [stat.ML].

Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural Machine Translation by Jointly Learning to Align and Translate, in: International Conference on Learning Representations (ICLR), pp. 1–15.

Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O., 2020. NuScenes: A Multimodal Dataset for Autonomous Driving, *ArXiv* preprint ArXiv:1903.11027 [cs.LG].

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S., 2020. End-to-End Object Detection with Transformers, in: European Conference on Computer Vision (ECCV), pp. 213–229.

Chen, Q., Ma, X., Tang, S., Guo, J., Yang, Q., Fu, S., 2019a. F-Cooper: Feature based Cooperative Perception for Autonomous Vehicle Edge Computing System Using 3D Point Clouds, in: ACM/IEEE Symposium on Edge Computing, pp. 88–100.

Chen, Q., Tang, S., Yang, Q., Fu, S., 2019b. Cooper: Cooperative Perception for Connected Autonomous Vehicles Based on 3D Point Clouds, in: IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 514–524.

Chorowski, J., Bahdanau, D., Cho, K., Bengio, Y., 2014. End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results, in: NeurIPS Workshop on Deep Learning.

Choy, C., Gwak, J., Savarese, S., 2019. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3075–3084.

Cornick, M.T., Koechling, J., Stanley, B., Zhang, B., 2016. Localizing Ground Penetrating RADAR: A Step Toward Robust Autonomous Ground Vehicle Localization. Journal of Field Robotics 33, 82–102.

Dempster, A.P., 1968. A Generalization of Bayesian Inference. Journal of the Royal Statistical Society: Series B (Methodological) 30, 205–232.

Dickmanns, E.D., Zapp, A., 1987. Autonomous High Speed Road Vehicle Guidance by Computer Vision. IFAC Proceedings Volumes 20, 221–226.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V., 2017. CARLA: An Open Urban Driving Simulator, in: Conference on Robot Learning (CoRL), pp. 1–16.

Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q., 2019. CenterNet: Keypoint Triplets for Object Detection, in: IEEE/CVF International Conference on Computer Vision (ICCV)., pp. 6568–6577.

Dudek, G., Jenkin, M., Milios, E., Wilkes, D., 1993. A Taxonomy for Swarm Robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 441–447.

Durrant-Whyte, H., Bailey, T., 2006. Simultaneous Localization and Mapping: Part I. IEEE Robotics and Automation Magazine 13, 99–110.

Elbaz, G., Avraham, T., Fischer, A., 2017. 3D Point Cloud Registration for Localization Using a Deep Neural Network Auto-Encoder, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4631–4640.

Elfes, A., 2013. Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception, *ArXiv* preprint ArXiv:1304.1098 [cs.RO].

ETSI, 2023. Intelligent Transport System (ITS);Vehicular Communications; Basic Set of Applications; Collective Perception Service; Release 2. Technical Specification 103 324. European Telecommunications Standards Institut. F-06921 Sophia Antipolis Cedex, FRANCE.

Fan, L., Wang, F., Wang, N., Zhang, Z., . Fully Sparse 3D Object Detection, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 351–363.

Feng, D., Cao, Y., Rosenbaum, L., Timm, F., Dietmayer, K.C.J., 2020. Leveraging Uncertainties for Deep Multi-modal Object Detection in Autonomous Driving, in: IEEE Intelligent Vehicles Symposium (IV), pp. 877–884.

Feng, D., Rosenbaum, L., Timm, F., Dietmayer, K., 2019. Leveraging Heteroscedastic Aleatoric Uncertainties for Robust Real-Time LiDAR 3D Object Detection, in: IEEE Intelligent Vehicles Symposium (IV), pp. 1227–1234.

G., R.R., R., R., 2018. An Empirical Study on MAC Layer in IEEE 802.11p/WAVE Based Vehicular Ad-Hoc Networks. Procedia Computer Science 143, 720–727.

Gal, Y., 2016. Uncertainty in Deep Learning. Ph.D. thesis. University of Cambridge. URL: `https://www.cs.ox.ac.uk/people/yarin.gal/website/thesis/thesis.pdf`.

Gawlikowski, J., Tassi, C.R.N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A.M., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., Zhu, X.X., 2023. A Survey of Uncertainty in Deep Neural Networks. Artificial Intelligence Review 56, 1513–1589.

Geddes, N.B., 1940. Magic Motorways. Random House, New York. URL: `https://archive.org/details/magicmotorways00geddrich`.

Geiger, A., Lenz, P., Urtasun, R., 2012. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3354–3361.

Gillan, W.J., 1989. PROMETHEUS and DRIVE: Their Implications for Traffic Managers, in: IEEE Vehicle Navigation and Information Systems Conference (VNIS), pp. 1–6.

Graham, B., der Maaten, L.V., 2017. Submanifold Sparse Convolutional Networks, *ArXiv* preprint ArXiv:1706.01307 [cs.NE].

Günther, H.J., Mennenga, B., Trauer, O., Riebl, R., Wolf, L., 2016. Realizing Collective Perception in a Vehicle, in: IEEE Vehicular Networking Conference (VNC), pp. 1–8.

Hall, D.L., Llinas, J., 1997. An Introduction to Multisensor Data Fusion. Proceedings of the IEEE 85, 6–23.

He, C., Wang, H., Chen, L., Luo, T., Cai, Y., 2023. V2X-AHD:Vehicle-to-Everything Cooperation Perception via Asymmetric Heterogenous Distillation Network, *ArXiv* preprint ArXiv:2310.06603 [cs.AI].

He, F., Gao, N., Jia, J., Zhao, X., Huang, K., 2022. QueryProp: Object Query Propagation for High-Performance Video Object Detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 12345–12352.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep Residual Learning for Image Recognition, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778.

Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computation 9, 1735–1780.

Hu, Y., Fang, S., Lei, Z., Zhong, Y., Chen, S., 2022. Where2comm: Communication-Efficient Collaborative Perception via Spatial Confidence Maps, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 4874–4886.

Itti, L., Koch, C., Niebur, E., 1998. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 1254–1259.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., Qu, R., 2019. A Survey of Deep Learning-Based Object Detection. IEEE Access 7, 128837–128868.

Jøsang, A., 2016. Subjective Logic: A Formalism for Reasoning Under Uncertainty. Artificial Intelligence: Foundations, Theory, and Algorithms, Springer. URL: `https://link.springer.com/book/10.1007/978-3-319-42337-1`, doi:`10.1007/978-3-319-42337-1`.

Kazemnejad, A., Padhi, I., Ramamurthy, K.N., Das, P., Reddy, S., 2023. The Impact of Positional Encoding on Length Generalization in Transformers, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 12345–12356.

Kendall, A., Gal, Y., 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 5574–5584.

Kenney, J.B., 2011. Dedicated Short-Range Communications (DSRC) Standards in the United States. Proceedings of the IEEE 99, 1162–1182.

Kingma, D.P., Ba, J., 2015. Adam: A Method for Stochastic Optimization, in: International Conference on Learning Representations (ICLR), pp. 1–15.

Kingma, D.P., Welling, M., 2014. Auto-Encoding Variational Bayes, in: International Conference on Learning Representations (ICLR), pp. 1–14.

Kirillov, A., He, K., Girshick, R.B., Rother, C., Dollár, P., 2018. Panoptic Segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9404–9413.

Kourabbaslou, S.S., Zhang, A., Atia, M.M., 2019. A Novel Design Framework for Tightly Coupled IMU/GNSS Sensor Fusion Using Inverse-Kinematics, Symbolic Engines, and Genetic Algorithms. IEEE Sensors Journal 19, 11424–11436.

Lakshminarayanan, B., Pritzel, A., Blundell, C., 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 6402–6413.

Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O., 2019. PointPillars: Fast Encoders for Object Detection From Point Clouds, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12697–12705.

LeCun, Y., 1989. Generalization and Network Design Strategies, in: Connectionism in Perspective. Elsevier, Zurich, Switzerland, pp. 143–155. URL: `https://masters.donntu.ru/2012/fknt/umiarov/library/lecun.pdf`.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep Learning. Nature 521, 436–444.

Lee, K., Kim, J., Park, Y., Wang, H., Hong, D., 2017. Latency of Cellular-Based V2X: Perspectives on TTI-Proportional Latency and TTI-Independent Latency. IEEE Access 5, 15800–15809.

Lei, Z., Ren, S., Hu, Y., Zhang, W., Chen, S., 2022. Latency-Aware Collaborative Perception, in: European Conference on Computer Vision (ECCV), pp. 316–332.

Levinson, J., Thrun, S., 2010. Robust Vehicle Localization in Urban Environments using Probabilistic Maps, in: International Conference on Robotics and Automation (ICRA), pp. 4372–4378.

Li, C., Dai, B., Wu, T., 2013. Vision-based Precision Vehicle Localization in Urban Environments, in: Proceedings of the Chinese Automation Congress (CAC), pp. 480–485.

Li, J., Xu, R., Liu, X., Ma, J., Chi, Z., Ma, J., Yu, H., 2023. Learning for Vehicle-to-Vehicle Cooperative Perception Under Lossy Communication. IEEE Transactions on Intelligent Vehicles 8, 2650–2660.

Li, J., Xu, R., Ma, J., Zou, Q., Ma, J., Yu, H., 2022a. Domain Adaptive Object Detection for Autonomous Driving under Foggy Weather, in: IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 612–622.

Li, T., Zhang, H., Gao, Z., Chen, Q., Niu, X., 2018. High-Accuracy Positioning in Urban Environments Using Single-Frequency Multi-GNSS RTK/MEMS-IMU Integration. Remote Sensing 10.

Li, Y., Ma, D., An, Z., Wang, Z., Zhong, Y., Chen, S., Feng, C., 2022b. V2X-Sim: Multi-Agent Collaborative Perception Dataset and Benchmark for Autonomous Driving. IEEE Robotics and Automation Letters 7, 10914–10921.

Li, Y., Ren, S., Wu, P., Chen, S., Feng, C., Zhang, W., 2021. Learning Distilled Collaboration Graph for Multi-Agent Perception, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 5987–5999.

Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., Dai, J., 2022c. BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers, in: European Conference on Computer Vision (ECCV), pp. 1–18.

Lin, J.R., Talty, T., Tonguz, O.K., 2015. On the Potential of Bluetooth Low Energy Technology for Vehicular Applications. IEEE Communications Magazine 53, 267–275.

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2020. Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 42, 318–327.

Liu, Y., Wang, T., Zhang, X., Sun, J., 2022a. PETR: Position Embedding Transformation for Multi-view 3D Object Detection, in: European Conference on Computer Vision (ECCV), pp. 684–700.

Liu, Y., Yan, J., Jia, F., Li, S., Gao, Q., Wang, T., Zhang, X., Sun, J., 2022b. PETRv2: A Unified Framework for 3D Perception from Multi-Camera Images, in: IEEE/CVF International Conference on Computer Vision (ICCV)., pp. 3092–3101.

Loukkal, A., Grandvalet, Y., Drummond, T., Li, Y., 2021. Driving among Flatmobiles: Bird-Eye-View Occupancy Grids from a Monocular Camera for Holistic Trajectory Planning, in: IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 51–60.

Lu, C., van de Molengraft, M., Dubbelman, G., 2019. Monocular Semantic Occupancy Grid Mapping With Convolutional Variational Encoder-Decoder Networks. IEEE Robotics and Automation Letters 4, 445–452.

Lu, Y., Li, Q., Liu, B., Dianati, M., Feng, C., Chen, S., Wang, Y., 2023. Robust Collaborative 3D Object Detection in Presence of Pose Errors, in: International Conference on Robotics and Automation (ICRA), pp. 4812–4818.

MacKay, D.J., 1992. A Practical Bayesian Framework for Backpropagation Networks. Neural Computation 4, 448–472.

Malinin, A., Gales, M.J.F., 2018. Predictive Uncertainty Estimation via Prior Networks, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 7047–7058.

Martín, J.S., Cortés, A., Zamora-Cadenas, L., Svensson, B.J., 2020. Precise Positioning of Autonomous Vehicles Combining UWB Ranging Estimations with On-Board Sensors. Electronics 9, 1238.

Marvasti, E.E., Raftari, A., Marvasti, A.E., Fallah, Y.P., Guo, R., Lu, H., 2020a. Cooperative LIDAR Object Detection via Feature Sharing in Deep Networks, in: IEEE Vehicular Technology Conference (VTC), pp. 1–5.

Marvasti, E.E., Raftari, A., Marvasti, A.E., Fallah, Y.P., Guo, R., Lu, H., 2020b. Feature Sharing and Integration for Cooperative Cognition and Perception with Volumetric Sensors, *ArXiv* preprint ArXiv:2011.08317 [cs.CV].

Meinhardt, T., Kirillov, A., Leal-Taixe, L., Feichtenhofer, C., 2022. TrackFormer: Multi-Object Tracking with Transformers, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8844–8854.

Meyer, G.P., Laddha, A., Kee, E., Vallespi-Gonzalez, C., Wellington, C.K., 2019. LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12677–12686.

Middlehurst, T., 2017. 25 Concept Cars from the Jet Age. `https://www.standard.co.uk/lifestyle/motors/25-concept-cars-from-the-jet-age-a3644241.html`. Accessed: 2024-12-09.

Miller, D., Dayoub, F., Milford, M., Sünderhauf, N., 2019. Evaluating Merging Strategies for Sampling-based Uncertainty Techniques in Object Detection, in: International Conference on Robotics and Automation (ICRA), pp. 8798–8804.

MMLab, 2020. MMDetection3D: OpenMMLab Next-generation Platform for General 3D Object Detection. `https://github.com/open-mmlab/mmdetection3d`.

Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K., 2014. Recurrent Models of Visual Attention, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 2204–2212.

Moradi-Pari, E., Tian, D., Bahramgiri, M., Rajab, S., Bai, S., 2023. DSRC Versus LTE-V2X: Empirical Performance Analysis of Direct Vehicular Communication Technologies. IEEE Transactions on Intelligent Transportation Systems 24, 1234–1245.

Neal, R.M., 1995. Bayesian Learning for Neural Networks. Springer New York, NY.

Niels, T., Mitrovic, N., Bogenberger, K., Stevanovic, A., Bertini, R.L., 2019. Smart Intersection Management for Connected and Automated Vehicles and Pedestrians, in: International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), pp. 1–8.

Ort, T., Gilitschenski, I., Rus, D., 2020. Autonomous Navigation in Inclement Weather Based on a Localizing Ground Penetrating Radar. IEEE Robotics and Automation Letters 5, 3267–3274.

Padilla, R., Netto, S.L., da Silva, E.A.B., 2020. A Survey on Performance Metrics for Object-Detection Algorithms, in: International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 237–242.

Pan, B., Sun, J., Leung, H.Y.T., Andonian, A., Zhou, B., 2020a. Cross-View Semantic Segmentation for Sensing Surroundings. IEEE Robotics and Automation Letters 5, 4867–4873.

Pan, H., Wang, Z., Zhan, W., Tomizuka, M., 2020b. Towards Better Performance and More Explainable Uncertainty for 3D Object Detection of Autonomous Vehicles, in: International Conference on Intelligent Transportation Systems (ITSC), pp. 1–7.

Parra, I., Ángel Sotelo, M., Llorca, D.F., Ocaña, M., 2010. Robust Visual Odometry for Vehicle Localization in Urban Environments. Robotica 28, 441–452.

Philion, J., Fidler, S., 2020. Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D, in: European Conference on Computer Vision (ECCV), pp. 194–210.

Pilsak, O., 1986. EVA—An Electronic Traffic Pilot for Motorists. SAE Transactions 95, 593–600.

Pomerleau, D.A., 1988. ALVINN: An Autonomous Land Vehicle in a Neural Network, in: Advances in Neural Information Processing Systems 1 (NIPS 1988), pp. 305–313.

Poole, D., Mackworth, A., Goebel, R., 1998. Computational Intelligence: A Logical Approach. Oxford University Press, New York.

Qiu, H., Yu, B., Tao, D., 2022. GFNet: Geometric Flow Network for 3D Point Cloud Semantic Segmentation. Transactions on Machine Learning Research 3, 1–15.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 234–241.

Rosen, D., Mammano, F., Favout, R., 1970. An Electronic Route-Guidance System for Highway Vehicles. IEEE Transactions on Vehicular Technology 19, 5–16.

SAE International, 2021. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. `https://www.sae.org/standards/content/j3016_202104/`. Accessed: 2024-12-09.

Sensoy, M., Kaplan, L., Kandemir, M., 2018. Evidential Deep Learning to Quantify Classification Uncertainty, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 3183–3193.

Shannon, Elwood, C., 1948. A Mathematical Theory of Communication. Bell System Technical Journal 27, 379–423.

Shelhamer, E., Long, J., Darrell, T., 2014. Fully Convolutional Networks for Semantic Segmentation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431–3440.

Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H., 2020. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10529–10538.

Statistisches Bundesamt, 2017. Unfallentwicklung auf deutschen Straßen 2017. `https://www.destatis.de/DE/Presse/Pressekonferenzen/2018/Verkehrsunfaelle-2017/pressebroschuere-unfallentwicklung.pdf?__blob=publicationFile`. Accessed: January 2024.

Statistisches Bundesamt, 2023. Zahl der Verkehrstoten sinkt im Jahr 2023 voraussichtlich leicht auf 2750. `https://www.destatis.de/DE/Presse/Pressemitteilungen/2023/12/PD23_471_46241.html`. Accessed: January 2024.

Suhr, J.K., Jang, J., Min, D., Jung, H.G., 2017. Sensor Fusion-Based Low-Cost Vehicle Localization System for Complex Urban Environments. IEEE Transactions on Intelligent Transportation Systems 18, 1078–1086.

Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to Sequence Learning with Neural Networks, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 3104–3112.

Sweeney, L., 1993. An Overview of Intelligent Vehicle Highway Systems (IVHS), in: Proceedings of WESCON, pp. 445–450.

Tan, H.S., Huang, J., 2006. DGPS-Based Vehicle-to-Vehicle Cooperative Collision Warning: Engineering Feasibility Viewpoints. IEEE Transactions on Intelligent Transportation Systems 7, 415–428.

Tsugawa, S., 1992. Road-to-Vehicle and Vehicle-to-Vehicle Communication Systems for Intelligent Vehicle-Highway Systems. Journal of the Society of Instrument and Control Engineers 31, 1257–1263.

Tsugawa, S., Watanabe, N., Fujii, H., 1991. Super Smart Vehicle System—Its Concept and Preliminary Works, in: Proceedings of the Vehicle Navigation and Information Systems Conference, pp. 429–434.

Uradziński, M., 2011. Range Analysis of RTK Base Station in Urban Environment, in: FIG Working Week 2011 & 6th National Congress of ONIGT, pp. 1–10.

Vasudevan, V.T., Sethy, A., Ghias, A.R., 2019. Towards Better Confidence Estimation for Neural Models, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7335–7339.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Łukasz Kaiser, Polosukhin, I., 2017. Attention is All You Need, in: Conference on Neural Information Processing Systems (NeurIPS), pp. 5998–6008.

Wahyudi, Listiyana, M.S., Sudjadi, Ngatelan, 2018. Tracking Object based on GPS and IMU Sensor, in: International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), pp. 214–218.

Wang, S., Liu, Y.H., Wang, T., Li, Y., Zhang, X., 2023. Exploring Object-Centric Temporal Modeling for Efficient Multi-View 3D Object Detection, in: IEEE/CVF International Conference on Computer Vision (ICCV)., pp. 1234–1243.

Wang, T.H., Manivasagam, S., Liang, M., Yang, B., Zeng, W., Tu, J., Urtasun, R., 2020a. V2VNet: Vehicle-to-Vehicle Communication for Joint Perception and Prediction, in: European Conference on Computer Vision (ECCV), pp. 605–621.

Wang, W., Saputra, M.R.U., Zhao, P., de Gusmão, P.P.B., Yang, B., Chen, C., Markham, A., Trigoni, N., 2019. DeepPCO: End-to-End Point Cloud Odometry through Deep Parallel Neural Network, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3248–3254.

Wang, Z., Feng, D., Zhou, Y., Zhan, W., Rosenbaum, L., Timm, F., Dietmayer, K., Tomizuka, M., 2020b. Inferring Spatial Uncertainty in Object Detection, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1036–1043.

Ward, E., Folkesson, J., 2016. Vehicle Localization with Low Cost Radar Sensors, in: IEEE Intelligent Vehicles Symposium (IV), pp. 864–870.

Wei, P., Wang, X., Guo, Y., 2020. 3D-LIDAR Feature Based Localization for Autonomous Vehicles, in: IEEE International Conference on Automation Science and Engineering (CASE), pp. 726–731.

Wolf, S.K., Begun, S.J., 1940. Tune 550-Highway Radio Ahead. Electronics 13, 45–47.

Xiang, L., Yin, J., Li, W., Xu, C.Z., Yang, R., Shen, J., 2024. DI-V2X: Learning Domain-Invariant Representation for Vehicle-Infrastructure Collaborative 3D Object Detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3208–3215.

Xu, R., Tu, Z., Xiang, H., Shao, W., Zhou, B., Ma, J., 2022a. CoBEVT: Cooperative Bird's Eye View Semantic Segmentation with Sparse Transformers, in: Conference on Robot Learning (CoRL), pp. 1–16.

Xu, R., Xia, X., Li, J., Li, H., Zhang, S., Tu, Z., Meng, Z., Xiang, H., Dong, X., Song, R., Yu, H., Zhou, B., Ma, J., 2023. V2V4Real: A Real-World Large-Scale Dataset for Vehicle-to-Vehicle Cooperative Perception, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13712–13722.

Xu, R., Xiang, H., Tu, Z., Xia, X., Yang, M.H., Ma, J., 2022b. V2X-ViT: Vehicle-to-Everything Cooperative Perception with Vision Transformer, in: European Conference on Computer Vision (ECCV), pp. 366–382.

Xu, R., Xiang, H., Xia, X., Han, X., Li, J., Ma, J., 2022c. OPV2V: An Open Benchmark Dataset and Fusion Pipeline for Perception with Vehicle-to-Vehicle Communication, in: International Conference on Robotics and Automation (ICRA), pp. 2583–2589.

Yan, Y., Mao, Y., Li, B., 2018. SECOND: Sparsely Embedded Convolutional Detection. Sensors 18, 3337.

Yang, B., Luo, W., Urtasun, R., 2018. PIXOR: Real-time 3D Object Detection from Point Clouds, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7652–7660.

Yang, K., Yang, D., Zhang, J., Li, M., Liu, Y., Liu, J., Wang, H., Sun, P., Song, L., 2023. Spatio-Temporal Domain Awareness for Multi-Agent Collaborative Perception, in: IEEE/CVF International Conference on Computer Vision (ICCV)., pp. 23326–23335.

Yin, H., Tian, D., Lin, C., Duan, X., Zhou, J., Zhao, D., Cao, D., 2024. V2VFormer++: Multi-Modal Vehicle-to-Vehicle Cooperative Perception via Global-Local Transformer. IEEE Transactions on Intelligent Transportation Systems 25, 2153–2166.

Yin, T., Zhou, X., Krähenbühl, P., 2021. Center-based 3D Object Detection and Tracking, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11784–11793.

Yu, H., Luo, Y., Shu, M., Huo, Y., Yang, Z., Shi, Y., Guo, Z., Li, H., Hu, X., Yuan, J., Nie, Z., 2022. DAIR-V2X: A Large-Scale Dataset for Vehicle-Infrastructure Cooperative 3D Object Detection, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2139–2148.

Yu, H., Yang, W., Ruan, H., Yang, Z., Tang, Y., Gao, X., Hao, X., Shi, Y., Pan, Y., Sun, N., Song, J., Yuan, J., Luo, P., Nie, Z., 2023. V2X-Seq: A Large-Scale Sequential Dataset for Vehicle-Infrastructure Cooperative Perception and Forecasting, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5486–5495.

Yuan, Y., Cheng, H., Sester, M., 2022. Keypoints-Based Deep Feature Fusion for Cooperative Vehicle Detection of Autonomous Driving. IEEE Robotics and Automation Letters 7, 3054–3061.

Yuan, Y., Cheng, H., Yang, M.Y., Sester, M., 2023. Generating Evidential BEV Maps in Continuous Driving Space. ISPRS Journal of Photogrammetry and Remote Sensing 204, 27–41.

Yuan, Y., Sester, M., 2021. COMAP: A Synthetic Dataset for Collective Multi-Agent Perception of Autonomous Driving, in: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, pp. 255–263.

Yuan, Y., Sester, M., 2022. Leveraging Dynamic Objects for Relative Localization Correction in a Connected Autonomous Vehicle Network, in: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, pp. 101–109.

Yuan, Y., Sester, M., 2024. StreamLTS: Query-based Temporal-Spatial LiDAR Fusion for Cooperative Object Detection, in: European Conference on Computer Vision Workshop (ECCVW), pp. 1–8.

Yuan, Y., Xia, Y., Cremers, D., Sester, M., 2025. SparseAlign: A Fully Sparse Framework for Cooperative Object Detection, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Zeng, F., Dong, B., Zhang, Y., Wang, T., Zhang, X., Wei, Y., 2022. MOTR: End-to-End Multiple-Object Tracking with Transformer, in: European Conference on Computer Vision (ECCV), pp. 659–675.

Zeng, X., Tao, C., Chen, Z., 2009. The Application of DSRC Technology in Intelligent Transportation System, in: IET International Communication Conference on Wireless Mobile and Computing (CCWMC), pp. 265–268.

Zhang, F., Stähle, H., Chen, G., Simon, C.C.C., Buckl, C., Knoll, A., 2012. A Sensor Fusion Approach for Localization With Cumulative Error Elimination, in: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 200–205.

Zheng, W., Tang, W., Chen, S., Jiang, L., Fu, C.W., 2021. CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3555–3562.

Zhou, B., Krähenbühl, P., 2022. Cross-view Transformers for Real-time Map-view Semantic Segmentation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13760–13769.

Zhou, Y., Tuzel, O., 2018. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4490–4499.

Zworykin, V.K., Flory, L.E., Ress, L.N., Mara, O.J., 1957. Electronic Control of Motor Vehicles on the Highway. Proceedings of the Highway Research Board 37, 436–451.

## Curriculum Vitae

### *Personal Information*

| | |
|---|---|
| Name | Yunshuang Yuan |
| Date of Birth | 25 September 1992 |
| Place of Birth | Hubei, China |
| Nationality | Chinese |
| Address | Husarenstraße 7, 30163 Hannover, Germany |

### *Education*

| | | |
|---|---|---|
| **PhD candidate** | Institute of Cartography and Geoinformatics,<br>Leibniz University Hannover | *Since 06.2020* |
| **M. Sc.** | Navigation and Field Robotics,<br>Leibniz University Hannover | *2017 - 2020* |
| **B. Sc.** | Building Electrical and Intellectualization,<br>Beijing University of Civil Engineering and Architecture | *2010 - 2014* |

### *Experience*

| | | |
|---|---|---|
| **Exchange scholar** | Mobility lab,<br>University of California, Los Angeles | *1/2023 - 3/2023* |
| **Intern** | BMW Group, Munich | *5/2019 - 2/2020* |
| **Research assistant** | Institute of Photogrammetry and Geoinformation &<br>Institute for Information Processing,<br>Leibniz University Hannover | *5/2018- 1/2019* |
| **Team member** | RoboCup team *LUHBots*,<br>Leibniz University Hannover | *2017* |

# Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover

*(Eine vollständige Liste der Wiss. Arb. ist beim Geodätischen Institut, Nienburger Str. 1, 30167 Hannover erhältlich.)*

| | | |
|---|---|---|
| Nr. 382 | GOLDSCHMIDT, Jürgen: | Die Berücksichtigung künftiger Entwicklungen bei der Verkehrswertermittlung (Diss 2022) |
| Nr. 383 | KRUSE, Christian: | Impact maps from bomb craters detected in aerial wartime images using marked point processes (Diss. 2023) |
| Nr. 384 | ZOURLIDOU, Stefania: | Traffic Regulation Recognition from GPS Data (Diss. 2023) |
| Nr. 385 | SLEDZ, Artuom: | Thermal anomaly detection based on information fusion from optical and infrared images (Diss. 2023) |
| Nr. 386 | WITTICH, Dennis: | Deep Domain Adaptation for the Pixel-wise Classification of Aerial and Satellite Images (Diss. 2023) |
| Nr. 387 | SINGH, Vishwa Vijay: | Lunar Laser Ranging - Improved Modelling and Parameter Estimation (Diss. 2023) |
| Nr. 388 | HARTMANN, Jens: | Hochgenaue 3D-Erfassung von Großstrukturen durch kinematisches terrestrisches Laserscanning (Diss. 2023) |
| Nr. 389 | ZHUGE, Xia: | Characterizing slope instability kinematics by integrating multi-sensor satellite remote sensing observations (Diss. 2023) |
| Nr. 390 | DOROZYNSKI, Mareike Marianne: | Image Classification and Retrieval in the Context of Silk Heritage using Deep Learning (Diss. 2023) |
| Nr. 391 | KNABE, Annike: | New Concepts for Gravity Field Recovery using Satellites (Diss. 2023) |
| Nr. 392 | KALIA, Andre: | Landslide activity detection based on nationwide Sentinel-1 PSI datasets (Diss. 2023) |
| Nr. 393 | BROCKMEYER, Marco: | Modellierung von Bodenbewegungen anhand heterogener Messverfahren am Beispiel der niedersächsischen Landesfläche (Diss. 2023) |
| Nr. 394 | ZHANG, Mingyue: | Characteristics and Benefits of Differential Lunar Laser Ranging (Diss. 2023) |
| Nr. 395 | DENNIG, Dirk: | Entwicklung eines kinematischen Profilvermessungssystems am Beispiel Kranbahnvermessung (Diss. 2024) |
| Nr. 396 | FUEST, Stefan: | Nudging travelers to societally favorable routes by means of cartographic symbolization (Diss. 2024) |
| Nr. 397 | MOFTIZADEH, Rozhin: | Advanced Particle Filtering for Vehicle Navigation based on Collaborative Information (Diss. 2024) |
| Nr. 398 | VASSILEVA, Magdalena Stefanova: | Satellite Radar Interferometry for Geohazards: from ground deformation to processes understanding (Diss. 2024) |
| Nr. 399 | MALINOVSKAYA, Anna: | Statistical Process Monitoring of Networks (Diss. 2024) |
| Nr. 400 | BANNERT, Jörn: | Der Einfluss von Straßenverkehrslärm und Umgehungsstraßen auf Grundstückswerte in Ortslagen - Bestimmung mittels Expertenbefragung nach der Delphi-Methode (Diss. 2024) |
| Nr. 401 | AXMANN: Jeldrik: | Maximum consensus localization using LiDAR (Diss. 2024) |
| Nr. 402 | TENNSTEDT, Benjamin: | Concept and Evaluation of a Hybridization Scheme for Atom Interferometers and Inertial Measurement Units (Diss. 2024) |
| Nr. 403 | HAKE, Frederic: | Schadenserkennung an Bauwerken mittels maschinellem Lernens (Diss. 2025) |
| Nr. 404 | KARIMIDOONA, Ali: | On Integrity Prediction for Network-RTK Positioning in Urban Environments (Diss. 2025) |
| Nr. 405 | ORTEGA, Mabel: | Domain Adaptation for Deforestation Detection in Remote Sensing: Addressing Class Imbalance and Performance Estimation (Diss. 2025) |
| Nr. 406 | KUPRIYANOV, Alexey: | Investigation of Optical Accelerometry and Novel Satellite Formations for Future Gravimetry Missions (Diss. 2025) |
| Nr. 407 | BREVA, Yannick: | On the Observation Quality of Robot-based GNSS Antenna Calibration for Determining Codephase Corrections (Diss. 2025) |
| Nr. 408 | GUO, Zelong: | Co- and Post-seismic Slip Models Inferred from InSAR Geodesy (Diss. 2025) |
| Nr. 409: | YUAN, Yunshuang: | Collective Perception: A Fully-Sparse Deep Learning Framework for Multi-Agent Data Fusion (Diss. 2025) |