Toward Robust and Scalable Collaborative Sensing for Connected and Autonomous Vehicles

by

Ruiyang Zhu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2026

Doctoral Committee:

Professor Z. Morley Mao, Chair
Associate Professor Ang Chen
Associate Professor Neda Masoud
Associate Professor Feng Qian

To my family, my love, and my friends

Ruiyang Zhu
ryanzhu@umich.edu
ORCID iD: 0000-0002-4524-5250

© Ruiyang Zhu 2026

ACKNOWLEDGEMENTS

The completion of a PhD journey is filled with challenges, growth, countless moments of self-reflection, and most importantly, learning. Over the past six years, I faced numerous rejections and moments of self-doubt, often questioning myself whether I am doing well enough in research to earn a PhD degree. Through those difficult times, I learned to accept failure not as a verdict, but as a stepping stone — to correct my mistakes, recalibrate, and continue moving forward with renewed confidence. This growth would not have been possible without the tremendous support of so many wonderful people along the way. I am deeply grateful to each of them.

First and foremost, I want to express my deepest gratitude to my PhD advisor, Professor Zhuoqing Morley Mao. I feel profoundly fortunate to have had the opportunity to work with her. She opened the door to scientific research for me and consistently challenged me to think more rigorously and ambitiously. I am deeply influenced by her passion for research and her resilience in continually striving for excellence. Her keen scientific insight, enduring support, and genuine investment in my development as a researcher shaped not only the work presented in this dissertation but the way I approach problems.

I would also like to express my sincere appreciation to my dissertation committee members: Professor Feng Qian, Professor Ang Chen, and Professor Neda Masoud. I want to thank them for their willingness to serve on my committee. Their thoughtful guidance and insightful feedback pushed me to sharpen my thinking and strengthened the quality of this dissertation in ways I am deeply grateful for.

I would like to thank my former and current lab mates for their help and insightful feedback throughout my Ph.D. journey: Yikai Lin, Xiao Zhu, Xumiao Zhang, Jiachen Sun, Qingzhao Zhang, Can Carlak, Shuowei Jin, Wenyuan Ma, Minkyoung Cho, Xueshen Liu, Jiarui Li, Joseph Brewington, Zesen Zhao, Hui Shen, Boyuan Zheng, and Una Chan. I also owe a great deal of gratitude to my research collaborators, without whose expertise and generosity much of the work presented in this dissertation would not have been possible: Dr. Fan Bai, Dr. Shuqing Zeng, and Dr. Mohammad Naserian from General Motors, Professor Feng Qian from the University of Southern California, and Professor Hang Qiu from the University of California, Riverside. I am also honored to have been a member of the NSF AI Institute for Edge Computing (Athena), where I had the privilege of contributing

to meaningful research advancements and collaborating with many exceptional researchers who broadened both my thinking. Beyond those named here, I have been fortunate over these six years to work with many more outstanding researchers, each of whom has left a meaningful imprint on my journey and my work.

Moreover, I want to thank two of my friends from high school: Yuqi Zhang and Chang Shen, who stayed connected and spent time playing video games with me during our spare time, listening to my complaints when my research ideas did not work out, and encouraging me to look forward. Maintaining a cross-country friendship for more than a decade is by no means easy, and I feel incredibly fortunate to have these friends virtually around.

I want to thank my family members: my parents Jianjun Zhu, Weiqing Chen, and all those who loved and supported me. Leaving home to pursue a doctoral degree on the other side of the world was never a small decision, and it is your unconditional love and unwavering belief in me that made it possible. Though more than six years have passed since I last came home, not a day goes by that I don't think of you and wish you health, happiness, and peace. I also hold deep regret in my heart that I was unable to return to say farewell to my grandparents, who raised me in my earliest years and passed away in 2023 and 2024. Their absence is a weight I will always hold. I hope that wherever they are, they know how much they are loved — and that they would be proud to see their grandson become Dr. Zhu.

Lastly but importantly, I want to give my deepest and most heartfelt thanks to my wife, Jiayi Fang. From our days as undergraduates at the University of Michigan eight years ago to every step of this long and rewarding journey, you have been my most steadfast companion. You celebrated every success with unbridled joy and held me steady through every setback with a compassion I never take for granted. None of this would have felt worth it without you beside me. I look forward to a future together that is filled with love, adventure, and all the treasured moments still to come.

Words are never enough to express how grateful I am to everyone who has been a part of this journey. I remember all your benevolence as I move forward into the next chapter, and I wish each of you a wonderful life.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

LIST OF FIGURES vii

LIST OF TABLES x

LIST OF APPENDICES xii

LIST OF ACRONYMS xiii

ABSTRACT xiv

CHAPTER

1 Introduction 1

1.1 Challenges 2
1.2 Summary of Contributions 3
1.2.1 Network System Support for Collaborative Vehicular Perception . . 3
1.2.2 Robustness to Temporal Inconsistencies in CVP 4
1.2.3 AI Solution for Spatial–temporal Misalignment under Network Imperfections in CVP 4
1.2.4 Crowd-sourcing Multi-vehicle Data for Map Building 5
1.3 Dissertation Organization 5

2 Background 7

2.1 Autonomous Vehicles 7
2.1.1 Connected and Autonomous Vehicles 8
2.1.2 Perception 8
2.1.3 Mapping 9
2.2 Vehicular Networks 10
2.2.1 Vehicle-to-Infrastructure Network 10
2.2.2 Vehicle-to-Vehicle Network 11
2.2.3 Applications of Vehicular Networks 12

3 Boosting Collaborative Vehicular Perception on the Edge with Vehicle-to-Vehicle Communication 13

3.1 Introduction 13

3.2 Background and Motivation . . . 16
3.3 Harbor Design . . . 18
3.3.1 Hybrid System Architecture . . . 19
3.3.2 Modeling Helper Assignment . . . 21
3.3.3 Efficient Helper Assignment Algorithm . . . 25
3.3.4 Timely Delivery of Detection Results . . . 26
3.4 System Implementation . . . 28
3.5 Evaluation . . . 29
3.5.1 Experimental Setup and Methodology . . . 29
3.5.2 Trace-driven Emulation Results . . . 31
3.5.3 Performance on Real-world Testbed . . . 33
3.5.4 Scaling to Large-scale Deployment . . . 35
3.5.5 System Overhead & Microbenchmarks . . . 36
3.6 Discussion and Related Work . . . 39
3.6.1 Summary . . . 40

4 Robust Real-time Multi-vehicle Collaboration on Asynchronous Sensors 41

4.1 Introduction . . . 41
4.2 Motivation . . . 44
4.2.1 Asynchronous Sensor Data . . . 44
4.2.2 Incomplete Occlusion Estimation . . . 46
4.2.3 Design Goals . . . 47
4.3 System Design . . . 48
4.3.1 System Overview . . . 48
4.3.2 System Components . . . 49
4.3.3 Motion-aware Occupancy Map . . . 51
4.3.4 Occupancy Flow Prediction . . . 54
4.3.5 Data Scheduling . . . 54
4.4 Implementation . . . 56
4.5 Evaluation . . . 56
4.5.1 Experimental Setup . . . 56
4.5.2 End-to-end System Performance . . . 58
4.5.3 Impacting Factors . . . 62
4.5.4 Ablation Study . . . 62
4.5.5 System Overhead . . . 63
4.5.6 Case study: *RAO*'s Real-world Benefits . . . 64
4.6 Summary . . . 66

5 SCORPION: Robust Spatial-Temporal Collaborative Perception Model on Lossy Wireless Network . . . 67

5.1 Introduction . . . 67
5.2 Motivation . . . 69
5.3 Method . . . 72
5.3.1 Problem Formulation . . . 72
5.3.2 Lost BEV Feature Reconstruction . . . 72

5.3.3 Deformable Spatial Cross-Attention . . . 74
5.3.4 Temporal Alignment . . . 76
5.3.5 Detection Head . . . 77
5.4 Experiments . . . 77
5.4.1 Datasets and Evaluation Metrics . . . 77
5.4.2 Quantitative Results . . . 77
5.4.3 Robustness to Individual Noise Factor . . . 79
5.4.4 Ablation Study . . . 80
5.4.5 System Overhead Analysis . . . 84
5.5 Summary . . . 84
6 GraphMap: Scalable Crowd-Sourced Global Vectorized HD Map Construction via Sparse Visual Graph Fusion . . . 85
6.1 Introduction . . . 85
6.2 Background and Motivation . . . 88
6.3 Design of `GraphMap` . . . 89
6.3.1 System Architecture . . . 89
6.3.2 MapGlue: Type-Aware Structure-Preserving Graph Matching . . . 91
6.3.3 Selective Map Sensing . . . 95
6.3.4 Detecting Map Updates with HD-CCDC . . . 97
6.4 System Implementation . . . 98
6.5 Evaluation of `GraphMap` . . . 99
6.5.1 Experiment Setup . . . 99
6.5.2 End-to-end Global Vectorized Map Results . . . 101
6.5.3 Ablation Study . . . 107
6.6 Summary . . . 110
7 Conclusion . . . 112
7.1 Future Work . . . 112
7.2 Concluding Remarks . . . 114
APPENDICES . . . 116
BIBLIOGRAPHY . . . 128

LIST OF FIGURES

FIGURE

3.1 An example of cellular uplink heterogeneity, where the uplink bandwidth of two carriers is recorded simultaneously at the same place while in urban driving. . . 14
3.2 An example of jointly using V2V and V2I to bridge a disconnected/poor-performing car (red). 15
3.3 An example showing the benefits of CVP by merging a nearby vehicle's data (green) to ego vehicle (blue). 17
3.4 System architecture of Harbor. The control plane and data plane run in parallel and do not block each other. 20
3.5 An example of graph-based interference score calculation (no link between e_2 and r_1 due to coverage). 24
3.6 Timeline of `Harbor`'s vehicle-side and server-side data processing. 27
3.7 Impact of network types on the detection result delivery latency. 27
3.8 Emulation experiment results for E2E drivable space detection performance of `Harbor` and baselines. 31
3.9 Emulation experiment results for E2E object detection performance of `Harbor` and baselines. 32
3.10 Object detection results of various schemes at *Mcity*. The scenario photo is shown in Figure 3.11. 34
3.11 End-to-end driving outcome using Baidu Apollo on real-world scenarios. 34
3.12 E2E live vehicular experiments. 34
3.13 Performance of `Harbor` w/ large number of CAVs. 34
3.14 Performance of `Harbor` under various levels of topology changes. 36
3.15 `Harbor` helper assignment overhead. 36
3.16 Benefits of `Harbor`'s assignment strategy. 36
3.17 Benefits of detection result delivery strategy. 36

4.1 Two scenarios where cooperative perception helps. 43
4.2 Synchronization problem in protocol diagram of cooperative perception. 44
4.3 Merged point clouds from two vehicles (A and B) at various levels of synchronization (Bird-eye view). 44
4.4 Inaccurate blind spot estimation (the consumer's/producer's points and heading in black/blue). 46
4.5 System Architecture of *RAO*. 49
4.6 Demonstration of the generation of motion-aware occupancy maps. 51
4.7 Distribution of perception IoU (accuracy). 59

4.8 Distribution of object density in merged data. 59
4.9 Synchronization of LiDAR data in various schemes. 59
4.10 Coverage of blind spots in *RAO* and AutoCast. 60
4.11 Number of objects detected based on distance. 60
4.12 Perception performance *w.r.t.* CAV density. 60
4.13 Perception performance *w.r.t.* traffic density. 61
4.14 Prediction accuracy *w.r.t.* vehicle motion. 61
4.15 Perception performance *w.r.t.* vehicle motion. 61
4.16 Ablation study on occupancy flow prediction. 63
4.17 Ablation study on the on-demand consumer-centric data scheduling. 63
4.18 Network bandwidth requirements and scalability study. 63
4.19 Latency breakdown of *RAO*. 63
4.20 Case study of multiple cooperative perception schemes on real-world drive scenes at *Mcity*. 65

5.1 Motivating examples of various degrees of spatial and temporal misalignments. Spatial misalignments occur due to sensing errors ($\star$) or dropped network packets, while temporal misalignments arise from sensor asynchronization ($\star$) and network delays. For simplicity and clarity, we focus on the cases marked with ($\star$) in this figure. This scenario is generated using the CoBEVT [208] model on the V2XSet Dataset [209]. TN and FP represent the True Negative and False Positive detections made by the model. 68
5.2 Illustration of detection performance drop suffering from lossy V2X wireless communication based on the V2XSet dataset [209]. The three baseline methods, CoBEVT [208], V2X-ViT [209], and CoAlign [143] are trained with no network packet loss and tested at various level of loss. 70
5.3 The overall architecture of `SCORPION`. The framework consists of three major components: a lost BEV feature reconstruction (L-BEV-R) module, a deformable spatial cross-attention (DSCA) module, and a temporal alignment (TA) module. 71
5.4 `SCORPION`'s Lost BEV Feature Reconstruction architecture. 73
5.5 An example of an encoded feature map into a network packet. 74
5.6 The architecture of the proposed deformable spatial cross-attention (DSCA) module. 75
5.7 Performance comparison across models under various network loss/localization/synchronization errors. 79
5.8 Qualitative comparison results on the V2XSet dataset under **localization error, synchronization error, and network loss altogether**. The green and red boxes are ground truth and prediction outputs, respectively. 81
5.9 Visualization of BEV feature reconstructed at the ego-agent side on an entrance ramp. 83
5.10 Visualization of BEV feature reconstructed at the ego-agent side on a curvy road. 83
5.11 Visualization of BEV feature reconstructed at the ego-agent side on a T-intersection. 83

6.1 System Architecture of GraphMap. Local HD map graphs generated by road agents are selectively uploaded to the cloud based on their sensing value. The cloud incrementally fuses these graphs with MapGlue, keeps track of historical map observations, and performs map updates via HD-CCDC. 89
6.2 Importance of Type-Aware Structure-Preserving Graph Matching: Lane dividers (Category 1) often exhibit larger positional offsets, whereas pedestrian crossings (Category 2) typically appear in consistent locations but vary in shape. Circles in the local observation represent the visual graph node encoding of HD map elements. Hungarian matching [116], a standard distance-based matching algorithm without type and structure-awareness, fails on both cases. 91
6.3 Visualization of a multi-intersection map built by GraphMap in OPV2V Town05. 103
6.4 Comparison of HBM, FDC, PolyMerge, GraphMap, and Ground Truth on DAIR-V2X Map 06. Orange lines represent lane dividers, and blue polygons represent pedestrian crossings. 104
6.5 Comparison of HBM, FDC, PolyMerge, GraphMap, and Ground Truth (GT) on Global Map Construction across another five different road configurations (Map 08, 09, 10, 13, 16). 105
6.6 Map accuracy *vs.* BW saving. 106
6.7 Computation cost of global map construction. 106
6.8 An example of global map element update detection through GraphMap's HD-CCDC algorithm. 106
6.9 Accuracy of map element add/removal detection. 107
6.10 Impact of MapGlue fusion. 107
6.11 Impact of SMS strategy. 107
6.12 An example of a time series plot for selective map sensing on DAIR-V2X Map06. 108
6.13 Impact of PDR. 109
6.14 Impact of pose error. 109
6.15 Impact of ϵ parameter. 109
6.16 GraphMap Latency breakdown. 109

A.1 The detailed transformer architecture of our proposed SCORPION. We use the same MSWin Block (*) from [209]. 119
A.2 Additional Visualization results on the V2XSet dataset under localization error, synchronization error, and network loss altogether. The green and red boxes are ground truth and prediction outputs, respectively. 121

B.1 Visualization of local vectorized HD map construction using infrastructure and vehicle sensors. 125
B.2 Agent trajectories on the other five different road configurations (Map 08-16). . 127

LIST OF TABLES

TABLE

1.1 Summary of projects and their organization in the dissertation. 6

3.1 Notations for helper assignment algorithm. 22
3.2 Summary of experiment settings. 30
3.3 Collaborative perception accuracy under different collaboration schemes. 33
3.4 Additional driving reaction time (compared to Local-only) and outcome of different collaboration schemes. 35
3.5 Control-plane and data-plane message overhead for each point cloud frame per vehicle(mean±std deviation). 37

4.1 Object detection accuracy under different multi-vehicle collaboration schemes and datasets. 58
4.2 Coverage of data sharing. 58
4.3 Data sharing overhead for each provider-consumer pair per frame (mean±standard deviation). 64

5.1 Comparison of Different Works Handling system imperfections. 70
5.2 Performance comparison on the V2XSet, OPV2V, and DAIR-V2X datasets in perfect environments. 78
5.3 Performance comparison under combined network loss, localization and synchronization noise. 78
5.4 Feature comparison and performance metrics on V2XSet. For baseline w/o DSCA, we use attention fusion from [209]. 81
5.5 Ablation study results of `SCORPION` (AP@0.5/0.7). 82
5.6 Table on inference speed of `SCORPION` on V2XSet [209] dataset. The latency is measured on an A100 GPU and shown with mean and standard deviation in ms. The definition of lossy network is the same as Table 3 in the main text. 82
5.7 Table on GPU memory usage during inference for `SCORPION` on V2XSet [209] dataset. The average memory usage is shown in MB. 84

6.1 Performance comparison on global HD map construction on DAIR-V2X. `GraphMap` (C) stands for agents using only camera data, and (C+L) for using both LiDAR and camera for local HD map detection. 99
6.2 Performance comparison on global vectorized HD map construction on V2XPnP. 102
6.3 Performance comparison on Town-level global vectorized HD map construction on OPV2V. 102

6.4 Control-plane and data-plane message overhead for each sensor frame per agent (mean±std). 110

A.1 Detailed architectural specifications for SCORPION. 120
A.2 Effect of feature fusion strategy in DSCA (AP0.5/0.7). 122

B.1 Performance of local vectorized HD map prediction. 124
B.2 Performance on local HD map prediction on larger area. 124

LIST OF APPENDICES

APPENDIX

A Supplementary Material for SCORPION 116

B Supplementary Material for GraphMap 124

LIST OF ACRONYMS

ADAS Advanced Driver Assistance Systems

AI Artificial Intelligence

AV Autonomous Vehicle

BEV Bird's-Eye-View

BW Bandwidth

C-V2X Cellular V2X

CAV Connected and Autonomous Vehicle

CVP Collaborative Vehicular Perception

DSRC Dedicated Short-Range Communications

E2E End-to-end

GNSS Global Navigation Satellite System

GPS Global Positioning System

HD Map High-definition Map

IoU Intersection over Union

mAP Mean Average Precision

PDR Packet Drop Ratio

RTK Real-time Kinematic Positioning

V2I Vehicle-to-Infrastructure

V2V Vehicle-to-Vehicle

ABSTRACT

Connected and Autonomous Vehicles (CAVs) aim to transform transportation by improving safety, efficiency, and accessibility. However, the sensing and decision-making capabilities of individual vehicles remain fundamentally constrained by limited sensor range, occlusions, and adverse environmental conditions. Collaborative sensing, where vehicles and infrastructure share perception and map information, has emerged as a key paradigm to overcome these limitations. Despite rapid research progress, practical deployment of collaborative sensing faces four major challenges: (1) scalable and efficient vehicular network support, (2) robustness to temporal inconsistencies and incomplete observations from individual vehicles, (3) robustness to spatial–temporal misalignment under network imperfections, (4) large-scale and continually updated high-definition (HD) map construction for vehicle navigation.

This dissertation presents a systematic study of robust and scalable collaborative sensing for connected autonomous vehicles, spanning network system design, robust multi-agent perception and global HD map construction. First, we design a hybrid Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) collaborative perception framework that improves bandwidth efficiency, scalability, and real-time performance through helper–helpee assignment and deadline-aware merging. Second, we address robustness to temporal asynchrony in distributed perception. In real-world deployments, vehicles observe and transmit information at different times due to unsynchronized sensors, heterogeneous computation pipelines, and communication delays. We develop predictive modeling and temporal alignment mechanisms that compensate for stale, missing, or delayed observations, enabling consistent collaborative perception under asynchronous conditions. Third, we tackle robustness to spatial–temporal misalignment across vehicles under realistic communication imperfections. Even small discrepancies in pose estimation, dynamic motion evolution, or coordinate frame alignment can significantly degrade multi-agent fusion. We propose unified spatial–temporal feature recovering and attention mechanisms that jointly model geometric uncertainty and dynamic scene evolution, enabling reliable collaborative perception under cross-agent misalignment and network uncertainty. Lastly, we introduce a scalable vehicle–cloud framework for crowdsourced global HD map construction using sparse graph fusion, selective sensing, and continual change detection, enabling bandwidth-efficient and dynamic map maintenance.

Together, these contributions establish robustness, scalability, and cross-layer co-design as fundamental principles for collaborative autonomous driving systems. This dissertation bridges mobile systems, multi-agent perception, mapping, and AI-driven autonomy, providing practical foundations for safe and large-scale deployment of connected autonomous vehicle applications.

CHAPTER 1

Introduction

Connected and autonomous vehicles (CAVs) [22, 21] are poised to revolutionize mobility by improving safety, efficiency, and accessibility. Autonomous driving has made remarkable progress over the past decade, driven by advances in high-resolution sensors, deep learning-based perception, and large-scale datasets. Modern autonomous vehicles can detect objects (*i.e.,* perception), track agents, and plan trajectories with increasing reliability. However, despite rapid advances in onboard perception and autonomy, the sensing capabilities of a single vehicle remain inherently limited. Sensors such as LiDAR and cameras [29] provide only a limited field of view, are prone to occlusions, and often degrade under adverse weather or lighting conditions. These limitations restrict situational awareness and can lead to safety-critical failures in complex traffic environments.

Collaborative sensing [62, 61, 163, 208, 209, 228] has emerged as a promising paradigm to overcome these challenges. By enabling vehicles and roadside infrastructure to share complementary sensing information, including LiDAR point clouds [18], camera images, and intermediate feature representations, collaboration significantly extends the perception range [62, 210] of individual vehicles and mitigates occlusions [227, 101] (named collaborative vehicular perception, CVP for short). Beyond perception, shared information can also support more accurate mapping of driving environments and enable vehicles to plan behaviors more safely and efficiently. As such, collaborative sensing represents a key enabler for the next generation of connected and autonomous driving.

Collaborative sensing systems for CAVs consist of two main components: a **wireless vehicular network system** [111, 149] and **AI-powered sensing applications** such as object detection [120], High-definition map (HD map) detection [127], and path planning [69]. Both of these two components play a vital role in promoting automated and intelligent transportation systems. For example, a vehicular network system must provide sufficient bandwidth and low latency to support real-time sharing across CAV agents. These systems are typically built upon wireless communication standards such as Dedicated Short-Range Communica-

tions (DSRC) [111] and cellular V2X (C-V2X) [192]. Meanwhile, sensing applications must coordinate heterogeneous sensing data from multiple vehicles to extract important information efficiently. Moreover, the **interplay between the network system and the sensing application** determines the overall effectiveness of collaborative sensing: network constraints (e.g., bandwidth, latency, packet loss) directly shape which sensing data can be shared, while application-level requirements (e.g., perception accuracy) dictate the priorities and optimizations needed at the network layer.

1.1 Challenges

Despite extensive research in collaborative perception algorithms, its practical deployment faces fundamental challenges, where my research emphasizes and aims to solve.

First, scalability of vehicular networking. Collaborative perception requires frequent data exchange among multiple vehicles and infrastructure nodes under strict latency constraints. Naïvely sharing raw sensor data or dense feature maps quickly overwhelms wireless bandwidth and leads to unstable performance under dynamic traffic density. Efficient network-level coordination and bandwidth-aware scheduling are therefore essential.

Second, robustness to temporal inconsistencies and incomplete observations. In real-world deployments, vehicles do not observe or transmit information synchronously. Sensor capture rates differ, onboard processing pipelines introduce variable latency, and communication delays further desynchronize shared data. As a result, collaborative systems must operate under temporally stale, missing, or partially received information. Without explicit modeling of temporal asynchrony, fusion performance can degrade significantly.

Third, robustness to spatial–temporal misalignment under network imperfections. Collaborative perception also relies on accurate spatial alignment across vehicles. In practice, packet loss, variable transmission delays, partial feature reception, localization noise, and dynamic scene evolution introduce inconsistencies in both space and time. Even small pose discrepancies or missing intermediate updates can severely impact multi-agent feature fusion. Robust mechanisms are needed to recover reliable cross-agent representations under communication imperfections and geometric uncertainty.

Fourth, scalable and continuously updated high-definition (HD) map construction. Beyond real-time perception, CAVs require accurate HD maps for localization and long-term navigation. Constructing and maintaining such maps at city scale demands efficient aggregation of crowd-sourced observations while minimizing communication overhead. Furthermore, maps must adapt to dynamic environmental changes without requiring centralized, high-bandwidth data collection.

These challenges span multiple layers of the autonomous driving computing stack, from

wireless networking to perception algorithms and cloud-based mapping systems. Addressing them requires a cross-layer perspective rather than isolated solutions. In summary:

> My research aims to improve **robustness**, **scalability**, and **performance** of collaborative autonomous driving systems, given the challenge of practical constraints **across multiple system layers**.

1.2 Summary of Contributions

Although primarily focusing on autonomous driving, my research is multifaceted and interdisciplinary, which involves three key components spanning AI, network systems, and cyber-physical system domains. To this end, this dissertation explores four subareas in the interdisciplinary space: (1) *network system support for collaborative vehicular perception (CVP) applications*; (2) *temporal robustness of CVP*; (3) *designing a robust AI model for CVP*; (4) *collaborative and distributed mapping support for autonomous driving.* The remainder of this chapter will summarize the core contributions of each section of research.

1.2.1 Network System Support for Collaborative Vehicular Perception

A fundamental prerequisite for collaborative vehicular perception is a network system capable of sustaining real-time, high-throughput data exchange among distributed agents under dynamic traffic conditions. Existing CVP systems adopt either a vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) paradigm, each with well-known limitations: V2V-only approaches suffer from poor scalability, while V2I-only approaches are susceptible to the inherent unreliability and heterogeneity of cellular uplink connectivity in urban environments.

Hybrid networked system for CVP. In this dissertation, we design `Harbor` [237], a first-of-its-kind hybrid CVP framework. Harbor [237] adopts V2I as its underlying framework and opportunistically employs V2V to bridge vehicles with poor cellular connectivity, enabling them to relay their LiDAR point clouds to an edge server through nearby helper vehicles. We formulate well-connected vehicles as helpers, relaying point cloud data from poorly connected helpees to an edge server, which performs sensor fusion and perception. We formulate the helper-helpee assignment as a weighted bipartite b-matching problem that jointly accounts for V2V distance, wireless interference, and V2I bandwidth constraints, and solve it efficiently in polynomial time to support real-time assignment updates under dynamic traffic topologies. We evaluate Harbor through real-world driving tests at Mcity and

large-scale emulations with up to 100 vehicles. Harbor improves object detection accuracy by up to 36% and reduces end-to-end latency by 18%–57% over V2V- and V2I-only baselines, and enables the ego vehicle to react 0.5–0.7 seconds earlier across all four safety-critical scenarios evaluated with Baidu Apollo [22].

1.2.2 Robustness to Temporal Inconsistencies in CVP

Another fundamental challenge in real-world collaborative vehicular perception is that vehicles capture and transmit sensor data asynchronously. Differences in CAV sensor configurations introduce unavoidable inter-vehicle time gaps, while data processing and wireless transmission add further arrival delays. To address this, we develop RAO [227], a real-time asynchronous occlusion-aware cooperative perception system. RAO introduces two novel designs: (1) motion-compensated occupancy flow prediction, which synchronizes LiDAR point clouds from different vehicles to a common future timestamp by tracking non-ground object clusters and estimating their motion via point cloud registration across consecutive frames; and (2) consumer-centric on-demand data scheduling, which allows each vehicle to accurately identify its own occluded areas and proactively request targeted LiDAR data from producers, overcoming the inaccurate blind spot estimation of prior producer-centric approaches and save network bandwidth. Evaluated across real-world and simulated datasets, RAO improves perception coverage by 34% and perception accuracy by up to 14% over state-of-the-art baselines under asynchronous settings while sharing only 10.9 KB of data per frame on average.

1.2.3 AI Solution for Spatial–temporal Misalignment under Network Imperfections in CVP

Practical deployment of collaborative perception must contend with a confluence of system-layer imperfections that prior work has addressed only in isolation: packet loss in V2X wireless networks corrupts shared intermediate feature maps, GPS localization errors introduce spatial misalignment across agents, and clock deviation combined with network delay causes temporal misalignment during feature fusion. Existing collaborative perception models assume ideal communication environments, and the few that address individual noise sources fail to account for the structured nature of packet encoding in vehicular networks or the joint effect of all three imperfections simultaneously. To address this gap, we design an end-to-end BEV-based intermediate fusion model named SCORPION [236] that co-designs the AI application layer with the underlying network and hardware constraints. SCORPION addresses network packet loss by recovering spatially lost BEV features using a masked autoencoder

that models the structured geometry of V2X packet encoding. To jointly handle spatial and temporal misalignment, SCORPION integrates a Deformable Spatial Cross-Attention module that learns offset locations in misaligned feature maps across agents, and a Temporal Alignment module that aligns historical ego-agent features to the current timestamp using ego-motion. Under the coexistence of all three noise sources, SCORPION significantly outperforms state-of-the-art methods by 5.9–13.2 absolute Average Precision in perception.

1.2.4 Crowd-sourcing Multi-vehicle Data for Map Building

Beyond perception, mapping is another important component for autonomous driving, where CAVs rely on an accurate global high-definition map to perform localization and planning. Constructing accurate global HD maps from crowdsourced vehicle fleets presents fundamental scalability challenges that prior work has not adequately addressed. Individual vehicles suffer from limited sensing range and occlusions, while existing multi-agent fusion approaches rely on dense, ego-centric BEV representations that are computationally expensive, sensitive to spatial-temporal misalignment, and poorly suited for global-scale map construction.

In this dissertation, we show GraphMap, a vehicle-cloud framework that shifts from dense BEV fusion to a graph-based sparse fusion paradigm. GraphMap encodes local vectorized map observations as sparse semantic graphs and incrementally fuses them on the cloud via a Type-Aware Structure-Preserving Graph Matching algorithm that handles spatial misalignment and partial overlaps across heterogeneous agents. To reduce communication overhead, GraphMap incorporates a Selective Map Sensing mechanism that prioritizes uploads from agents with higher Information Gain. Our experimental findings reveal that GraphMap outperforms state-of-the-art baselines by 37.2 mAP while reducing communication cost by over 30% and computation overhead by more than 3×.

1.3 Dissertation Organization

The remainder of the dissertation is organized as follows. Chapter 2 provides the background and related work that contextualize the technical problem domains addressed in this dissertation. Chapter 3-6 demonstrate my completed research projects in detail, with summarized project titles and publication venue in Table 1.1. Lastly, Chapter 7 discusses the potential future work based on the findings from the chapters in this dissertation and concludes the work.

Specifically, Chapter 3 introduces the details on designing network system support for CVP. In chapter 4, we first identify the importance of synchronization in CVP and pro-

Table 1.1: Summary of projects and their organization in the dissertation.

Scope	**Project Name / Paper Title**	**Publication Venue**
I: Designing network system support for collaborative vehicular perception	a. Boosting Collaborative Vehicular Perception on the Edge with Vehicle-to-Vehicle Communication	SenSys'24 [237]
II: Improving robustness of CVP under real-world noises	a. Robust Real-time Multi-vehicle Collaboration on Asynchronous Sensors	MobiCom'23 [227]
	b. SCORPION: Robust Spatial-Temporal Collaborative Perception Model on Lossy Wireless Network	IROS'25 [236]
III: Building large-scale HD mapping support from multi-vehicle data	a. GraphMap: Scalable Crowd-Sourced Global Vectorized HD Map Construction via Sparse Visual Graph Fusion	CVPRW'25 [235] Full paper in submission

pose a systematic solution to solve it. In chapter 5, we further extend the horizon of CVP under various types of realistic noises, and propose a data-driven framework to holistically solve them with a unified transformer model. Chapter 6 looks beyond the perception application, and explores how crowdsourced multi-vehicle data can be leveraged to construct accurate global HD maps at scale, proposing a graph-based sparse fusion framework that jointly addresses communication efficiency, spatial-temporal misalignment, and long-term map maintenance.

CHAPTER 2

Background

2.1 Autonomous Vehicles

Autonomous vehicles (AVs) are ground vehicles capable of sensing their environment and navigating without human input, relying on a combination of onboard sensors, computing hardware, and decision-making software to perform driving tasks. A fully capable AV system typically consists of several tightly integrated modules: perception, which processes raw sensor data such as cameras, LiDAR, and radar to detect and classify surrounding objects and road elements; localization and mapping, which determines the vehicle's precise position within a prior map using techniques such as simultaneous localization and mapping (SLAM) or map-based point cloud registration; prediction, which models the future behavior of dynamic agents such as pedestrians and other vehicles; planning, which generates future trajectories; and control, which translates planned trajectories into low-level actuation commands for steering, throttle, and braking. These modules are orchestrated by high-performance onboard computing platforms that must operate under strict real-time constraints and remain robust to the complex and uncertain conditions inherent in real-world driving. Significant progress in autonomous driving has been demonstrated by a growing number of industry players: Waymo [40] has deployed fully driverless robotaxi services in cities such as San Francisco and Phoenix, and Zoox [42] is developing a purpose-built bidirectional autonomous vehicle designed for dense urban environments. On the software side, commercial stacks such as Uber ATG [39] have pushed the boundaries of large-scale fleet deployment, while open-source platforms including Baidu Apollo [22] and Autoware [21] have lowered the barrier to entry for research and development.

Among these interconnected modules, perception and mapping play a foundational role in enabling safe and reliable autonomous driving. Perception provides the real-time understanding of the surrounding environment upon which all downstream modules depend, while mapping supplies the static prior knowledge that anchors localization and reduces the

burden on onboard sensing. In this dissertation, we primarily focus on enhancing the performance of these two components – specifically by moving beyond the single-vehicle paradigm and leveraging inter-vehicle collaboration to overcome the fundamental limitations of onboard sensing range, occlusions, and map coverage. The following subsections provide more technical background for these two modules, which serve as the basis for the research contributions presented in later chapters.

2.1.1 Connected and Autonomous Vehicles

Connected and Autonomous Vehicles (CAVs) extend the traditional autonomous vehicle paradigm by integrating wireless communication capabilities that allow vehicles to exchange information with other vehicles, roadside infrastructure, and cloud services [114]. Through communication, CAVs form a distributed network of mobile sensing agents that can collaboratively share sensing data, traffic information, and map updates. This connectivity enables a shift from isolated, single-vehicle autonomy toward cooperative driving systems in which vehicles jointly reason about the surrounding environment with improved situational awareness. By leveraging shared observations from multiple viewpoints, CAV systems can potentially overcome the fundamental limitations of onboard sensing, including restricted sensor range, occlusions caused by buildings or other vehicles, and limited map coverage.

2.1.2 Perception

Perception is the process by which an autonomous vehicle constructs an understanding of its surrounding environment from raw sensor data. It encompasses a range of tasks including 3D object detection, semantic segmentation, depth estimation, and object tracking, all of which are essential for safe navigation. Modern AV perception systems primarily rely on two sensor modalities: cameras, which provide rich color and texture information, and LiDAR (Light Detection and Ranging), which produces dense 3D point clouds of the surrounding scene with precise depth measurements.

LiDAR-based 3D object detection has become a cornerstone of AV perception. Raw LiDAR point clouds are sparse and unstructured, requiring specialized processing pipelines. Voxel-based methods such as VoxelNet [232] discretize the 3D space into voxels and apply 3D convolutions, while pillar-based methods such as PointPillars [120] encode points into vertical columns for faster inference. A widely adopted intermediate representation is the Bird's-Eye View (BEV) feature map, which projects 3D point cloud features onto a top-down 2D grid, providing a consistent spatial reference frame that is well-suited for tasks like vehicle detection and lane-level localization [131].

Camera-based perception complements LiDAR by providing dense visual features. Recent methods lift 2D image features into 3D space using depth estimation or transformer-based projection, enabling BEV representations to be constructed from camera images alone [131]. Multi-modal fusion approaches that combine LiDAR and camera data further improve detection accuracy and robustness, particularly in challenging lighting and weather conditions [65].

Despite significant advances, onboard perception remains fundamentally limited by occlusions, sensor range, and environmental noise. These limitations motivate the development of collaborative perception systems that extend beyond the single-vehicle sensing paradigm, which will be detailed in later chapters.

2.1.3 Mapping

High-definition (HD) maps are a critical component of autonomous driving infrastructure, providing centimeter-level geometric and semantic descriptions of the road environment that go beyond what real-time onboard perception can reliably produce. HD maps typically encode lane-level geometry, road boundaries, traffic signs, pedestrian crossings, and other static road features in structured formats such as OpenDRIVE [37] or NDS [36]. These maps serve multiple functions in an autonomous driving stack: they support precise vehicle localization by providing a prior map for sensor-based registration, supply semantic context for planning and prediction modules, and reduce the real-time perception burden by providing a reliable static scene representation. Industry leaders have made HD mapping a cornerstone of their autonomous driving infrastructure: Waymo maintains a proprietary HD map covering its operational domains that is continuously updated through its fleet of mapping vehicles [41], while Zoox constructs dense semantic maps of its target urban environments to support its purpose-built robotaxi [38].

Conventional HD map construction pipelines rely on specialized mapping vehicles equipped with high-grade LiDAR and GPS systems to collect dense point clouds along road segments. These raw data are then processed through simultaneous localization and mapping (SLAM) algorithms such as LeGO-LOAM [173] and LIO-SAM [174] to produce 3D point cloud maps, followed by extensive manual annotation to label semantic elements. Trained annotators must manually label every lane boundary, road marking, traffic sign, and static road element across hundreds or thousands of kilometers of road, a process that is not only labor-intensive and expensive but also inherently error-prone and difficult to scale. Furthermore, once constructed, HD maps require continuous maintenance as road conditions, lane configurations, and infrastructure change over time.

Recent advances in online HD map construction aim to generate local maps in real

time directly from onboard sensor streams with AI models, eliminating the need for offline annotation. Deep learning methods such as HDMapNet [127], VectorMapNet [141], and MapTR [133] learn to detect vectorized map elements – represented as polylines and polygons – from camera or LiDAR inputs within a local perception range around the ego vehicle in real-time. These vectorized representations are compact, semantically structured, and compatible with downstream planning modules, making them increasingly adopted in production autonomous driving systems [134, 221]. Nevertheless, despite these research advances, widespread production deployment remains limited, as the vehicle's local sensing range fundamentally constrains onboard map construction and remains susceptible to incomplete or erroneous map elements in occluded regions

2.2 Vehicular Networks

2.2.1 Vehicle-to-Infrastructure Network

Vehicle-to-Infrastructure (V2I) communication refers to the wireless exchange of data between vehicles and fixed roadside entities such as base stations, roadside units (RSUs), and edge servers. V2I communication leverages both dedicated vehicular communication standards and general cellular infrastructure. On the cellular side, LTE and 5G NR provide broad geographic coverage and high peak throughput, with 5G in particular offering significantly improved bandwidth, reduced latency, and support for massive device connectivity [153]. Cellular V2X (C-V2X) [5], standardized under the 3GPP Release 14 framework, provide low-latency broadcast communication designed specifically for safety-critical vehicular applications in the 5.9 GHz band.

In practice, V2I links exhibit significant variability in throughput and reliability, posing challenges for latency-sensitive collaborative driving applications. Real-world measurement studies have confirmed that 5G vehicular links experience considerable fluctuations in throughput, latency, and packet loss under typical driving conditions [85, 153]. The variability reflects the dynamic nature of vehicular channels, where signal quality fluctuates with vehicle speed, obstruction, and network load. Vehicular communication commonly adopts UDP [162] as the underlying transport protocol to minimize transmission latency [52, 51]. Studies have shown that TCP [81], which relies on connection-oriented retransmission, suffers severe performance degradation or outright failure under high vehicle speeds and dynamic link conditions [71], making it poorly suited for time-critical vehicular data exchange. Unlike TCP, UDP provides no retransmission or delivery guarantees, meaning packet loss is an inherent and unavoidable characteristic of V2I data transmission. As a result, application-layer

designs for collaborative perception and mapping must be explicitly engineered to tolerate and recover from packet loss, rather than assuming reliable data delivery.

Edge computing architectures place compute resources at RSUs or cellular base stations, enabling vehicles to offload latency-sensitive tasks such as perception and mapping to nearby servers rather than relying solely on onboard compute [228]. This vehicle-edge-cloud continuum allows the system to balance the trade-off between communication latency, computation cost, and result accuracy, and is increasingly adopted as the deployment architecture for large-scale collaborative autonomous driving applications.

2.2.2 Vehicle-to-Vehicle Network

The primary standard for dedicated V2V communication is DSRC, based on the IEEE 802.11p physical layer [111], which operates in the 5.9 GHz band and provides a channel bandwidth of 10 MHz, yielding a maximum physical-layer data rate of 27 Mbps, though effective application-layer throughput is considerably lower in practice due to protocol overhead and channel contention. As introduced in the V2I context, C-V2X also extends to direct V2V communication through its PC5 sidelink interface defined in 3GPP standards [193], making it a unified standard capable of supporting both vehicle-infrastructure and vehicle-vehicle communication within a single framework. Both DSRC and C-V2X are designed for low-latency broadcast communication, supporting safety-critical applications such as cooperative collision avoidance, platooning, and intersection management.

Like V2I, V2V communication relies on UDP as the primary transport protocol [162, 51], inheriting the same trade-off between low latency and unreliable delivery. Measured packet loss rates vary considerably with road type and vehicle density, ranging from approximately 12% in open field scenarios to over 32% on congested rural roads [52]. The Maximum Segment Size (MSS) for V2X communication is constrained to up to a few hundred of bytes [118, 149, 193], which determines how data payloads – such as encoded feature maps – are partitioned into network packets and therefore governs the spatial structure of information loss when packets are dropped.

V2V link quality is also highly dynamic. As vehicles move, the network topology changes rapidly, leading to intermittent connectivity and fluctuating link reliability between vehicle pairs [51, 187]. Distance, relative velocity, obstacles, and multipath fading all contribute to link variability. These characteristics make V2V communication fundamentally different from infrastructure-based networks and impose unique constraints on the design of applications that depend on timely and reliable data exchange between vehicles.

2.2.3 Applications of Vehicular Networks

Vehicular networks enable vehicles and roadside infrastructure to communicate and collaborate in real time, supporting a broad range of CAV applications [136, 163, 165]. Early applications focused on safety-critical messaging [137, 163], such as cooperative collision avoidance, emergency brake warnings, and intersection management, where low-latency broadcast communication enables vehicles to react to hazards beyond their individual sensing range. As network bandwidth and onboard computing capabilities have grown, more data-intensive collaborative applications have emerged, including cooperative object detection [101, 210, 62, 209, 144], distributed mapping [60], and offloading of compute-intensive tasks to edge servers. **Collaborative vehicular perception** (also referred to as cooperative or collaborative perception) is a prominent example, which aims to enhance CAVs' awareness of the surrounding environment by enabling vehicles to share and fuse complementary sensor observations in real time. A vehicle's onboard sensors can miss the detection of critical objects due to occlusions caused by other obstacles, buildings, or roadside structures. The occluded object may be detected too late to allow a safe response, posing significant safety risks. Furthermore, onboard sensors are constrained by a predefined perception range due to technological limitations, and there are scenarios where this range is insufficient for conducting safe maneuvers – particularly at high speeds or in complex intersections. By sharing sensor data or derived representations over vehicular networks, neighboring vehicles and infrastructure nodes can collectively construct a richer and more complete view of the shared environment, effectively overcoming these individual limitations. Existing collaborative perception schemes can be grouped in several ways. (1) **Data sharing stage:** Early-fusion schemes [62, 163, 228] share raw sensor data in its original format, offering maximum information fidelity but demanding high communication bandwidth; feature-level schemes [123, 101, 210, 209] transmit intermediate neural network features extracted from sensor data, offering a balance between bandwidth efficiency and perception accuracy; late-fusion schemes [136, 178] share lightweight perception outputs such as object bounding boxes, minimizing bandwidth at the cost of fusion flexibility. (2) **Network communication topology:** V2V schemes [163, 69, 164] enable decentralized peer-to-peer sharing directly among vehicles without relying on fixed infrastructure; V2I schemes [228, 47] leverage roadside units or edge servers to aggregate data from multiple vehicles and distribute enhanced perception results back to participants. (3) **Sensor modality:** Collaborative perception can in principle be built on various sensor types [163, 105], though existing works primarily focus on LiDAR-based representations due to their geometric precision and compatibility with 3D object detection pipelines.

CHAPTER 3

Boosting Collaborative Vehicular Perception on the Edge with Vehicle-to-Vehicle Communication

3.1 Introduction

Connected and autonomous vehicles (CAVs) use various on-board 3D sensors such as LiDAR [18] and stereo cameras [20] to perceive the environment [8, 7]. However, a single vehicle's sensing range is limited, and its view can be obstructed by obstacles [88, 144, 67]. To overcome these limitations, Collaborative Vehicular Perception (CVP) [163, 228, 164, 227] enables multiple vehicles to cooperatively extend their views through wirelessly sharing their sensor data, as powered by the increasingly mature onboard wireless infrastructure [3, 4, 1]. CVP can boost a wide range of autonomous driving and Advanced Driving Assistance Systems (ADAS) such as drivable space detection [83, 159] and object detection [120, 218, 177].

Early works of CVP only involve two vehicles sharing their views, relying on a simple vehicle-to-vehicle (V2V) view exchange paradigm [163, 62]. As researchers realize the importance of larger-scale view sharing, they incorporate edge nodes into CVP systems, which combine all the views uploaded by participating vehicles and distribute the processed results back to the vehicles [228, 144, 178]. This centralized vehicle-to-infrastructure (V2I) paradigm is more scalable from the computation perspective: a powerful cloud server can efficiently merge multiple vehicles' views in real-time. The challenges, however, stem from the unreliable wireless communication. Take the most promising cellular access for V2I as an example, from the most recent FCC cellular coverage report [12] and National Highway Traffic Safety Administration's (NHTSA) fatality analysis reports [9], the average coverage of major U.S. carriers ranges from 38.5% to 71.6% (rural and urban combined). Another recent study [129] suggests that "5-bar" 5G connectivity is statistically more unreliable than weaker signal strength in densely populated urban areas. Our measurement in a major U.S. city also confirms that onboard cellular (uplink) connectivity is highly heterogeneous. Vehicles may experience very poor performance during urban driving (*e.g.,* 8 to 20 seconds of low

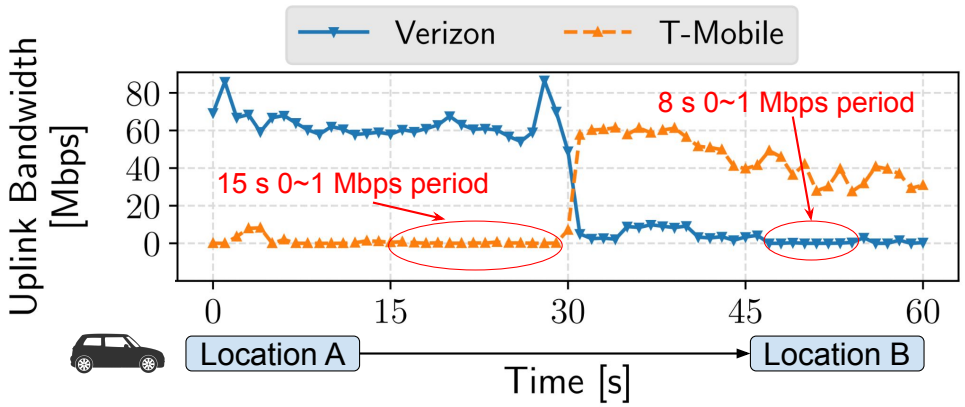


Figure 3.1: An example of cellular uplink heterogeneity, where the uplink bandwidth of two carriers is recorded simultaneously at the same place while in urban driving.

uplink bandwidth less than 1 Mbps, see Figure 3.1). Such heterogeneity and unreliability are *inherent* due to vehicles' mobility, which incurs frequent handovers [91], fluctuating channel quality, and continual interference.

In this paper, we advocate a hybrid design by leveraging the synergy between V2I and V2V. We instantiate the design into `Harbor`, a full-fledged CVP framework. `Harbor` adopts V2I as its fundamental underlying framework, and *opportunistically* employs V2V to boost the performance. As shown in Figure 3.2, when some vehicles (called *helpees*) have poor V2I connectivity, other vehicles (called *helpers*) can relay their sensor data to the edge. The edge will then merge the views and send processed results (*e.g.,* detected vehicles) back to the vehicles. Participation in `Harbor` can be incentivized through enhanced safety and CAV features, conceptually similar to the recent commercially deployed peer-to-peer CDN (PCDN)[1] infrastructure [200]. Nevertheless, realizing this seemingly straightforward idea involves making several non-trivial design decisions, as elaborated next.

Partitioning the Workload between the Edge and Vehicles. There exist many V2X works that bridge V2V and V2I links [155, 192, 186, 195]. `Harbor` fundamentally differs from these approaches by not only focusing on improving *network* connectivity but also addressing both computation (point cloud merging and detection) and communication. For communication, as described before, helpers relay traffic for helpees. For computation, we may also let some helpers merge point clouds for helpees. This design, however, significantly increases the system complexity and enlarges the attack surface (in contrast, a helpee–helper–edge communication path can be end-to-end encrypted). Therefore, in `Harbor`, only the edge is responsible for merging vehicles' views. A vehicle can only perform *local merging*: merge its own view with the results returned by the edge without further distributing the

[1]Internet users contribute their under-utilized edge devices as "mini" CDN nodes, which work in conjunction with traditional centralized CDN nodes.

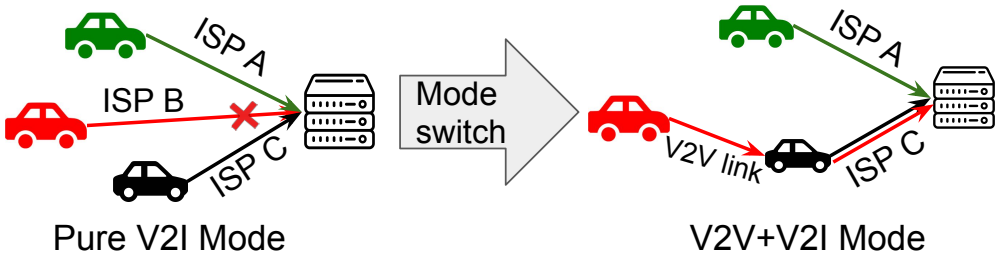


Figure 3.2: An example of jointly using V2V and V2I to bridge a disconnected/poor-performing car (red).

merged view to downstream. The vehicle-side merge can occur when the vehicle's uploaded view misses the edge-side merging deadline (to be detailed soon). This design simplifies `Harbor` by reducing the solution space and avoiding reconciling the helpee–helper paths of computation and communication.

Efficient Helper-helpee Assignment. Even with simplified computation flows, making helper-helpee assignments needs to consider three types of dynamics: V2V network connectivity, V2I network connectivity, and vehicle mobility. We formulate helper-helpee assignment as an optimization problem taking into account the above dynamics. In our model, a vehicle only sends its lightweight state (location, V2I bandwidth measurements, *etc.*) to the edge, either directly or via some helper.[2] A challenge here is how to measure the V2V bandwidth. A naïve approach to sending active probes is not scalable due to its quadratic complexity. Instead, `Harbor` introduces a novel approach that uses robust analytical models of V2V distance and wireless interference to implicitly approximate the V2V bandwidth, without actually probing it. This leads to an ultra-lightweight control plane. While it is difficult to precisely solve the above formulation at runtime, we develop a polynomial-time approximation algorithm based on efficient weighted bipartite matching. This enables `Harbor` to frequently update helper-helpee pairs under dynamic, complex traffic scenarios.

Optimizations to Further Improve `Harbor`'s Performance. We introduce two system-level mechanisms to boost `Harbor`'s performance. On the edge side, the edge server enforces a deadline to determine when to start the data merging and perception tasks. It ensures timely delivery of perception results to the vehicles. On the vehicle side, the same wireless V2V medium is used to carry both the point cloud (uplink) and returned perception results (downlink). Vehicles properly prioritize between the two types of traffic at the MAC layer to tackle their unbalanced network resource usage.

Scalable and Lightweight Design. `Harbor` is designed to be both lightweight and

[2]The state information (control plane) is much lighter than the point cloud data (data plane). We thus assume the former can be delivered to the edge (otherwise the vehicle will fall back to offline mode).

scalable. This is achieved through leveraging V2V to assist V2I in an *opportunistic* manner. By utilizing spare V2I resources from vehicles through V2V, Harbor provides *lower end-to-end latency* compared to using V2I alone. Additionally, the lightweight control plane design and polynomial-time approximation optimization for helper-helpee assignment, ensures that the system can make real-time decisions with *low computation overhead*. These design choices make `Harbor` well-suited for complex traffic scenarios with dynamic network conditions.

We integrate the above components into `Harbor`, a distributed CVP framework that can support a wide range of CAV applications. We comprehensively evaluate its performance through real-world driving tests and emulations over real cellular networks, using two predominant CAV applications: drivable space detection [159] and object detection [120]. We highlight the key results as follows.

• Through emulations on various traffic densities, `Harbor` reduces end-to-end drivable space detection and object detection latency by 18% to 57% compared to using only V2V or V2I, without sacrificing the detection accuracy.

• We deploy `Harbor` and conduct driving tests at a real-world CAV testbed, *Mcity* [27]. The results reveal *abundant opportunities of leveraging V2V in practice*: `Harbor` improves the object detection and drivable space detection accuracy by 12% on average (up to 36%) compared to pure V2V- or V2I-based CVP.

• We conduct a larger emulation experiment consisting of 20 to 100 vehicles and demonstrate that `Harbor` outperforms the V2V- or V2I-only baselines by an average of 43% and 14% in detection latency and accuracy under fast-changing traffic topology, showcasing superior scalability and performance.

Despite the encouraging results above, `Harbor` does bear several limitations (§3.6.1). We thus do not claim any "optimality" of our solution (which is also difficult to define given the highly dynamic, multi-objective, and multi-stakeholder nature of CVP). Instead, we explore the design space of fusing V2V into the V2I-based CVP design. Through real-world deployment, we demonstrate that `Harbor`, a first-of-its-kind hybrid CVP architecture, is practical, beneficial, and scalable.

3.2 Background and Motivation

Autonomous vehicles rely on various on-board 3D vision sensors to understand the physical world consisting of road segments, pedestrians, other vehicles, and more, to make driving decisions. For example, LiDAR [18] is a major on-board vision sensor, which fires laser lights at different angles and measures how long it takes for the lights to return to the sensor after reflection from objects. Based on this, LiDAR calculates the distance of these objects

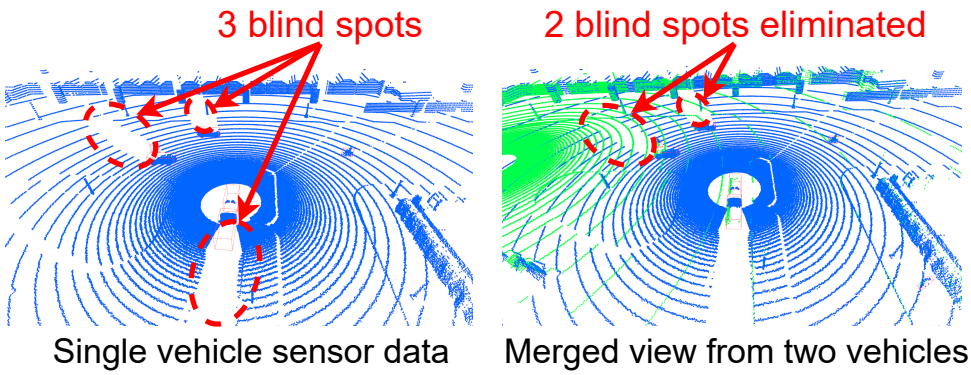


Figure 3.3: An example showing the benefits of CVP by merging a nearby vehicle's data (green) to ego vehicle (blue).

and produces 3D point clouds to represent the surrounding environment. Different software modules will further process the collected point cloud data and make appropriate driving decisions.

Collaborative Vehicular Perception. As mentioned in §3.1, collaborative vehicular perception (CVP) has the potential to benefit autonomous driving by extending its view by merging sensing information from different vehicles. Figure 3.3 shows an example that visualizes point cloud data generated from the state-of-the-art autonomous driving simulator, CARLA [26]. The blue points represent a single vehicle's data while the green ones represent data from a nearby vehicle. As illustrated, merging the two point clouds eliminates two of the three blind spots.

Sharing raw sensor data *vs.* sharing processed data. Existing CVP schemes can be grouped into three main categories based on the stage of data sharing: (1) Raw-data sharing schemes [228, 118, 164, 227] share the original raw sensor data; (2) Feature-level sharing schemes [61, 69, 222, 198, 209] send intermediate features of perception; (3) Object-level sharing schemes [136, 178, 179] directly share lightweight perception results such as object bounding boxes. In this work, we build `Harbor` based on sharing raw sensor data (point clouds) instead of detection results or processed features for two reasons. *First*, sharing raw data preserves the high level of detail provided by sensors. Based on the state-of-the-art work [130, 197] in computer vision, raw sensor data sharing achieves higher perception accuracy compared to sharing detection results or intermediate features. Various efforts [228, 164, 227] have made raw data sharing more scalable and practical. *Second*, sharing raw data is more versatile. Vehicles that receive processed features are constrained to use the same format and application for detection. In contrast, raw sensor data can be used by a wide range of CAV applications for multiple purposes. Although `Harbor` currently focuses on point-cloud-level collaboration, the same design principles can be applied to merge intermediate features or detection results (§3.6).

Limitations of V2V. It has been shown that V2V-based CVP is not scalable [228]. This is because in a V2V-based CVP, vehicles exchange sensor data streams for sharing [163]; depending on the design, either all vehicles receive all the point cloud data from other vehicles and perform collaborative perception, or only one of the vehicles receives data, performs perception, and disseminates the perception results. In either way, a V2V system would require a shared wireless network with sufficient capacity established by vehicles. Traditional WiFi standards (802.11 ac/n) fall short in supporting highly variable wireless channels under high mobility [164]. While WiFi standards like 802.11ad/ay offer Gbps bandwidth [184], they suffer from small coverage and poor mobility performance. Long-distance wireless communication technologies such as DSRC/802.11p [111] can achieve only ∼6 Mbps, inadequate for accommodating a large number of vehicles.

Limitations of V2I. V2I-based CVP leverages a powerful edge server for view merging and perception. Vehicles upload their sensor data to the server without direct communications among them [228]. This allows for more network and compute resources to support large-scale CVP. However, V2I communications are not always reliable: even the most ubiquitous cellular networks still have coverage issues [12]. Based on the LTE coverage map from FCC [12] and the National Highway Traffic Safety Administration's Fatality Analysis Reporting System [9], the LTE coverage on the roads of 5 U.S. states with the largest number of traffic fatalities in 2021 is only 68.5%, 71.6%, and 38.5%, for *AT&T*, *Verizon*, and *T-Mobile*, respectively. Our driving tests in a major U.S. city also confirm that onboard cellular connectivity is fluctuating and highly heterogeneous, as described in §3.1 (Figure 3.1).

Joint Use of V2V and V2I. The limitations of existing V2V-only [163, 62, 118] or V2I-only [228, 138, 144] CVP architecture motivate us to explore a joint use of both. `Harbor` differs from existing work in innovating the network communication paradigm: it adopts a hybrid approach that combines the V2V and V2I communication for CVP application. In `Harbor`, V2V communication is used to bridge vehicles with poor V2I connectivity to participant in collaboration, thereby enhancing CVP performance and robustness across different network conditions.

3.3 Harbor Design

The observations in §4.2 motivate a hybrid CVP architecture that leverages the synergy between V2I and V2V. In this work, we aim to explore the benefits of leveraging V2V communication to enhance the performance of V2I-centric CVP systems. Our primary research question focuses on how the integration of V2V and V2I together can improve CVP tasks, particularly in terms of perception accuracy, detection latency, and system scalability. The

resulting hybrid CVP system, `Harbor`, adopts V2I as its basic underlying framework, and *opportunistically* employs V2V to boost the performance. It enables multiple vehicles to leverage their V2V and/or V2I network access to share 3D sensor data. `Harbor` aims to reduce CVP latency and improve accuracy by combining V2V and V2I communications in an efficient and scalable manner, especially under dynamic wireless network and mobility conditions.

3.3.1 Hybrid System Architecture

Figure 3.4 shows the system architecture of `Harbor`. At any specific time, vehicles fall into two disjoint sets: a helper set and a helpee set. Vehicles in the helper set have good V2I connectivity, whereas vehicles in the helpee set do not. Helpers and helpees establish a V2V network so that helpees can leverage V2V to send their point clouds to the server through the V2I links of helpers, who also upload their own point clouds.

Assumptions made by `Harbor`. There are three assumptions for the designed system `Harbor`: (1) Collaborative vehicular perception is initiated only for vehicles that are in close proximity, therefore vehicles are connected to the same edge server during the collaboration period; (2) Vehicles exchange control plane messages periodically to the edge server for making *helper-helpee assignment* decisions; (3) The V2V network is used only for relaying data back/forth from the edge server to helpee vehicles. These assumptions are typically common and practical in collaborative vehicular perception systems [228, 163, 164] and other vehicular systems [192, 155].

Data Plane Operations. As shown in Figure 3.4, before uploading each point cloud frame, cropping and adaptive encoding [228] are performed to reduce the data size for transmission. On receiving a round of point clouds from different vehicles[3], the server merges these frames based on their locations (described shortly) and performs perception tasks such as drivable space detection [83, 159] and object detection [120, 218, 177] to generate detection results. Finally, the detection result is sent back to the vehicles. Note that in parallel to the above procedure, vehicles still run local detection. This allows a vehicle to fall back to using the local detection if remote results are not available.

Control Plane Message Exchanges. `Harbor` needs to strategically assign helpers to helpees (§3.3.2). A naïve way is to let each helpee find a helper in a distributed manner. However, distributed pairing lacks a holistic view of the network topology and V2I/V2V resources in the system. `Harbor` opts for a centralized method, where the server collects vehicle

[3]Like existing work [228, 164], we assume point cloud capture on different vehicles are synchronized. In reality, point cloud data generation time across vehicles can have slight misalignment, which can be solved by [96, 227].

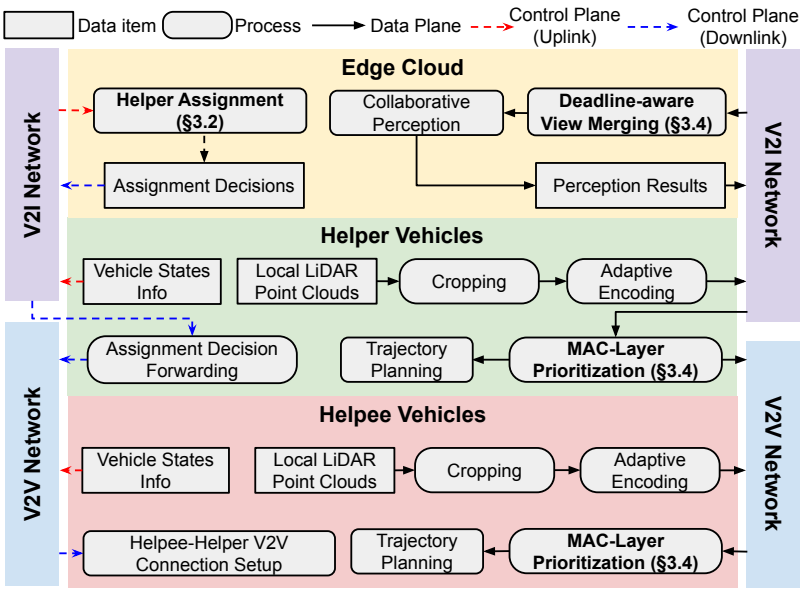


Figure 3.4: System architecture of Harbor. The control plane and data plane run in parallel and do not block each other.

state information and makes assignment decisions accordingly. Dashed lines in Figure 3.4 show the control plane message exchanges in `Harbor`. Helpers periodically send their states to the server, including locations, V2I bandwidth measurements, and V2V network routing tables. Similarly, helpees broadcast their state information periodically using the V2V network and rely on helpers to forward it to the server. Such state information is lightweight in nature and can be delivered to the edge server efficiently, as validated in §3.5.5 and other existing CVP systems [228, 163]. The server then computes the best assignment (§3.3.2) and informs the concerned helpers. Upon receiving the notifications, each helper establishes a V2V connection with the assigned helpee for point cloud forwarding. Due to vehicles' mobility, `Harbor` needs to update helper assignments dynamically. `Harbor` server periodically computes the best assignment every T_p and updates the assignment to vehicles in the system.

Mobility/Handover Management. Inline with previous V2I-based CVP [228], we envision that each edge node serves a pre-defined geographic area (similar to the service area of a cellular base station). Optionally, an edge's service area can be divided into *subareas*. For example, for an edge node deployed at a major intersection, each subarea may correspond to one of the four road segments. Subareas can be overlapped. Vehicles' collaboration and edge-side merging are confined within a subarea. This helps reduce the search space of helper assignments and the edge's merging overhead. When a vehicle (either a helper or helpee) moves across the subarea boundary, its potential helper or helpee(s) will be immediately switched to those in the new subarea. The fast switch is owning to the *stateless* nature of

helper assignment – assignment history is not considered when making assignment decisions. This makes `Harbor` adaptive to the volatile traffic dynamics. Our large-scale evaluation with up to 100 emulated vehicles (§3.5.4) performs such a subarea partition based on the *road_id* information of the map. A handover procedure similar to the above *intra-edge* handover can be applied to *inter-edge* handovers. We leave the detailed inter-edge handover design as our future work.

3.3.2 Modeling Helper Assignment

Efficient assignment of helpers to helpees is a crucial challenge in `Harbor` due to the dynamics of V2V and V2I links as well as the interplay among helpers and helpees. Our goal is to find an assignment that pairs helpees to helpers to maximize the overall system performance, specifically by minimizing all vehicles' point cloud upload time to deliver the sensor data to the edge server. Also, assisting helpees should bring minimal impact on helpers' own sensor data upload. As our goal is to pair helpees (V2V sources) with helpers (V2V destinations), our design is orthogonal to and compatible with the underlying routing scheme (*e.g.,* [57, 68]) used in the V2V network.

`Harbor` performs helper assignment in two steps. First, it determines the *set* of helpers and helpees; second, it finds for each helpee its helper (if available). While ideally the two steps could be combined in a holistic optimization framework, we separate them to reduce the overall complexity of assignment search, which needs to be performed at a high frequency (every $T_p = 100$ ms in our implementation) to tackle the volatile traffic dynamics. In the first step, `Harbor` applies a simple heuristic: regard vehicles whose V2I uplink bandwidth is lower than (at least) a threshold as helpees (helpers). The threshold, which is empirically set to 1 Mbps [12], ensures that a helper has minimal bandwidth to upload its own sensor data before helping others.

Next, given the selected helpers and helpees, we consider how to search for the assignments. A prerequisite here is the V2V bandwidth. As mentioned in §3.1, actively probing every V2V link is not scalable. We thus introduce a novel approach that uses robust analytical models of V2V distance and wireless interference to implicitly approximate the V2V bandwidth, without actually probing it. This leads to a lightweight optimization model (elaborated next). To begin with, we identify three key factors that impact the performance of an end-to-end (E2E) path (which consists of V2I and possibly V2V links).

• **Physical distance.** In a V2V wireless network, the physical distance between a helpee and a helper can affect the throughput of the helpee-to-helper path [224].

• **V2V network interference.** Scenarios like multi-hoping between helper and helpee and multiple helpees transmitting data to the same helper may incur interferences [109, 158] and

Table 3.1: Notations for helper assignment algorithm.

Symbol	Meaning
$E = \{e_1, ..., e_{\|E\|}\}$	Helpee vehicle set: helpee 1 to helpee $\|E\|$
$R = \{r_1, ..., r_{\|R\|}\}$	Helper vehicle set: helper 1 to helper $\|R\|$
$a_k = (e_i, r_j)$	A (helpee, helper) assignment pair
$N(r_j)$	Helping capacity of helper j
$A = \{a_1, a_2, ..., a_m\}$	An assignment is a set of assignment pairs ($m \leq \|E\|$)
$S_{dist}(A)$	Distance score of assignment A
$S_{intf}(A)$	V2V interference score of assignment A
$D(e_i, r_j)$	Physical distance between helpee i and helper j
$P(a_k) = e_i \rightarrow ... \rightarrow r_j$	The network path for a_k (helpee i to helper j)
$IC(v_i, X)$	Interference count produced by vertices in graph X to vertex v_i
$IC(P(a_k), X)$	The sum of interference counts for all vertices in path $P(a_k)$
$\mathcal{G}(A)$	Graph produced by active transmitting and receiving nodes in assignment A

thus hurt the performance of a helpee-helper connection.

• **V2I network bandwidth.** The V2I bandwidth of a helpee's helper can also impact the sensor data upload time, especially when the helper-to-server path becomes the bottleneck of the E2E path. Note the V2I bandwidth of a helper is shared by itself and potentially one or more helpees.

It is important to note that the goal of our analytical model is not to *precisely estimate V2V bandwidth* for each helper-helpee vehicle pair, but to select a helper vehicle that has potentially better V2V performance compared to other helper vehicles. To achieve this, we leverage V2V distance and interference as two primary factors, both of which have been shown to strongly correlate with wireless network bandwidth [224, 158]. Additionally, given that the effective range of CVP systems is generally smaller (around 200m×100m [210]) compared to large-area mesh networks [43], the use of distance and interference as bandwidth estimators is both practical and effective, as validated in our experiments (§3.5.5).

Assuming the total number of helpees is $m = |E|$, an assignment $A = \{a_1, a_2, ..., a_m\}$ contains one or multiple assignment pairs from helpee e_i to its helper r_j (Table 3.1). For an assignment pair a_k, its score is defined as Equation 3.1a. The distance and interference scores quantify how good the assignment pair is. To obtain scores of A, we sum each pair's distance and interference score and then aggregate them using an aggregation function f (Equation 3.1b). While different aggregation functions can be applied, `Harbor` uses the harmonic mean to aggregate scores of different assignment pairs. The use of the harmonic mean helps filter out assignments with low-score pairs to prevent the entire collaboration

system from being slowed down by a single "slow" vehicle, as it is very sensitive to small-value outliers.

$$Score(a_k) = S_{dist}(a_k) + S_{intf}(a_k) \tag{3.1a}$$

$$Score(A) = f(Score(a_1), ..., Score(a_m)) \tag{3.1b}$$

$$\textbf{maximize} \;\; Score(A) \;\; \textbf{s.t.} \;\; C_{bw}(a) \geq 0, \; \forall \; a \in A \tag{3.1c}$$

The objective of finding the best assignment A is defined as Equation 3.1c, and the assignment is subject to the bandwidth constraint C_{bw}. The intuition behind the constraint is that `Harbor` tries to leverage the spare V2I bandwidth from helpers to help helpees upload sensor data without hurting helpers' own performance. In cases where helpers are not capable of helping all helpees, *i.e.,* the constraint of the above optimization cannot be satisfied (*e.g.,* the helpers' V2I bandwidth is limited for helping all the helpees), `Harbor` schedules the assignment so that as many possible helpees are helped as possible.

As `Harbor` updates assignments frequently (§3.3.1), unassigned helpees may be assigned later when there are more helpers or helpers have better V2I bandwidth.

We next describe how to calculate scores and constraints for each individual factor. Since distance and interference have different units and scales, we design each score function (S_{dist}, S_{intf}) to map the corresponding factor to a value within the range of $[0, 1]$ to avoid a single factor dominating the entire score. Table 3.1 summarizes notations used to calculate different scores of an assignment.

• **Distance Score.** Intuitively, the wireless throughput is better when two nodes physically reside closer to each other. Therefore, the assignment should prefer a helper closer to a helpee. The distance score for a pair $a_k = (e_i, r_j)$ is quantified by comparing the physical distance between e_i and r_j with the longest possible distance between e_i and any other helper in the system. Formally, for an assignment pair $a_k = (e_i, r_j)$, its distance score is defined as $S_{dist}(a_k) = 1 - \frac{D(e_i,r_j)}{max\{D(e_i,r_k)\}_{k=1}^{|R|}}$. This equation shows that a smaller physical distance between e_i and r_j leads to a higher score, indicating a nearby node is preferred for better V2V bandwidth. Also, the score value lies between 0 and 1, as desired. With $S_{dist}(a_k)$ for all a_k in A, the distance score for the assignment can be calculated using the aggregation function, as described in Equation 3.1b.

• **Interference Score.** The interference score is designed so that a higher score is generated when the assignment creates less interference. To fulfill this, we propose a graph-based method to quantify the level of interference by leveraging an observation: interference will occur between active senders and from senders to receivers when they are in each other's coverage range [109]. In `Harbor`, an active sender is a node that transmits bulk sensor data

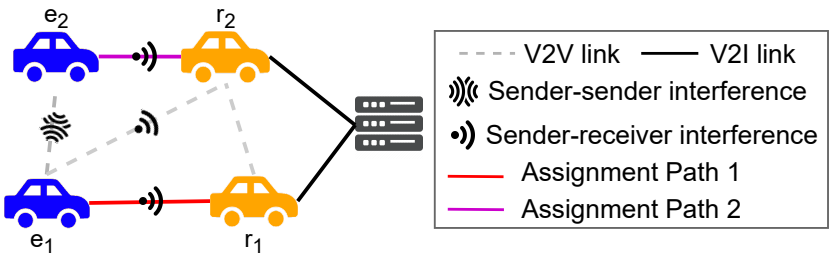


Figure 3.5: An example of graph-based interference score calculation (no link between e_2 and r_1 due to coverage).

to another node through V2V, *i.e.,* helpees that send data to their helpers. For instance, in Figure 3.5, e_2, e_1 are active senders while r_1, r_2 are receivers and use V2I to send received sensor data to the server.

To quantify the interference, we define `Interference Count`, IC. At a high level, $IC(v_i, X)$ measures the number of interfering nodes (*i.e.,* in-coverage active senders) from the network topology X to a node vertex v_i, and $IC(a_k, X)$ measures the interference produced by X to a pair a_k. To calculate IC for a_k, we sum up the interference count for all nodes in the network path $P(a_k)$ of pair a_k, *i.e.,* $IC(a_k, X) = \Sigma_{v_i \in P(a_k)} IC(v_i, X)$. The entire graph can be constructed from control messages received by the server, which include the routing tables on vehicles (§3.3.1). The interference score function maps the interference count for a_k to a value between 0 and 1. The formula to calculate the interference score for a_k is $S_{intf}(a_k) = 1 - \frac{IC(a_k, \mathcal{G}(A)) - IC(a_k, \mathcal{G}(\{a_k\}))}{IC(a_k, G_0) - IC(a_k, \mathcal{G}(\{a_k\}))}$. G_0 represents the graph formed by assuming all nodes are actively generating/receiving network traffic. $IC(a_k, \mathcal{G}(\{a_k\}))$ calculates the interference caused only by the nodes in the network path of a_k. Similarly, $IC(a_k, \mathcal{G}(A))$ is the interference caused by the actual assignment A and $IC(a_k, G_0)$ is the maximum possible interference for the pair a_k. Take Figure 3.5 as an example, there are two connections in the assignment $A = \{a_1, a_2\}$, where $a_1 = (e_1, r_1)$ and $a_2 = (e_2, r_2)$. Their network paths are $p_1 = P(a_1) = e_1 \rightarrow r_1$ and $p_2 = P(a_2) = e_2 \rightarrow r_2$. To determine $IC(a_2, \mathcal{G}(A))$, we sum up the interference sources for all nodes in the network path p_2, *i.e.,* $IC(a_2, \mathcal{G}(A)) = \Sigma_{v_i \in p_2} IC(v_i, \mathcal{G}(A)) = IC(e_2, \mathcal{G}(A)) + IC(r_2, \mathcal{G}(A)) = |\{e_1\}| + |\{e_1, e_2\}| = 1 + 2 = 3$. Note r_1 is not counted when calculating $IC(r_2, A)$ because r_1 uses V2I for sending sensor data to the server and does not generate bulk sensor data transmission in the V2V network. Similarly, we can calculate $IC(a_2, \mathcal{G}(\{a_2\})) = 1$ and $IC(a_2, G_0) = 6$. Finally we have the interference score $S_{intf}(a_2) = 1 - \frac{3-1}{6-1} = 0.6$. Similarly, $IC(a_1, A)$ can be calculated and the interference score $S_{intf}(A)$ can be obtained. With both assignment pairs' scores calculated, the score for A can be obtained from Equation 3.1b.

• **Bandwidth Constraint.** A higher V2I bandwidth is more desired as it can potentially reduce the data upload time. In Equation 3.1c, a V2I bandwidth constraint C_{bw} is reinforced

for each assignment pair. The bandwidth constraint ensures the number of helpees a helper helps does not exceed its "capability". To quantify this, we calculate the V2I helping capacity of each helper $N(r_j)$ based on the average bandwidth required for transmitting the encoded point clouds. The capacity is calculated as $N(r_j) = \lfloor \frac{BW(r_j) - BW_{req}}{BW_{req}} \rfloor$. $N(r_j)$ quantifies how many helpees the helper r_j can help. We set $BW_{req} = 4.8$ Mbps based on the bandwidth requirement for transmitting point cloud at 10 fps after encoding with high resolution [228, 11]. Then we can derive the equation for all the assignment pair a: $C_{bw}(a) = N(r_a) - \sum_{k=1}^{m}(\pi_2(a_k) = r_a) \geq 0$, where r_a is the helper in a and $\pi_2(a_k)$ selects the helper (second element) in the ordered assignment pair a_k.

3.3.3 Efficient Helper Assignment Algorithm

Intuitively, the assignment with the highest score can be obtained from a brute-force search. However, this method is not quite scalable as the number of possible assignments grows exponentially as $|E|$ (the number of helpees) increases, with a complexity of $O(|R|^{|E|})$. In order to handle real-time dynamics for more vehicles, we propose an efficient optimization to solve the best matching in polynomial time.

We transform the helper-helpee matching problem in `Harbor` to a weighted bipartite b-matching (`WBbM`) problem [45]. Consider the helper set and helpee set as the two matching sets of the bipartite graph, the weight of each potential edge between a helper-helpee pair is determined by Equation 3.1a. The objective of `Harbor`'s assignment becomes finding the maximum score from all possible matching. Different from the well-known weighted bipartite matching (`WBM`) problem, where the solution yields a one-to-one matching, the `WBbM` model allows a node to match with multiple nodes. This fits with our system design, where one helper vehicle can help multiple helpees. For `WBbM`, each node has a capacity C, which determines the maximum number of nodes it can match to. In `Harbor`'s context, the helping capacity $N(r_j)$ of helpers is determined by their V2I bandwidth, as mentioned before. For solving the `WBbM` problem, it can be reduced to a `WBM` problem by creating replications of nodes. We adopt this method in `Harbor`'s algorithm optimization. Specifically, for each helper with $N(r_j) > 1$, we create $N(r_j)$ copies of it and connect it to all possible neighbor e_i of the original node r_j with the corresponding edge weight. We then solve the newly formed graph using the existing Hungarian algorithm for `WBM` problems. Since for any possible assignment, each r_j can help at most $N(r_j)$ helpees, we make sure that the V2I bandwidth constraint is satisfied as each node after replication will end with $\leq N(r_j)$ matchings. We summarize the proposed matching algorithm in pseudo-code as Algorithm 1. Given that the complexity of solving `WBM` is $O(|R|^2|E|)$, it is straightforward to prove that the complexity of the algorithm becomes $O(|R|^2|E|^2)$ compared to the $O(|R|^{|E|})$ exhaustive search. We

evaluate the effectiveness of the WBbM optimization in §3.5.5.

Algorithm 1: Harbor's Helper Assignment Algorithm.

Input: helpee set $\mathbf{E} = \{e_1, ..., e_i\}$, helper set $\mathbf{R} = \{r_1, ..., r_j\}$.
Output: A – selected assignment to pair helpees with the helpers.

```
/* Define V2I bandwidth, helping capacity, score map, and helper set after
   replication                                                          */
```
$BW, N, S, \mathbf{R}' \leftarrow$ HashMap, HashMap, HashMap, HashSet;
for $r_j \in \mathbf{R}$ **do**
  $BW_j \leftarrow$ getV2IBW(r_j);
  $N_j \leftarrow$ getHelpingCap(BW_j);
  for $k \in \{1, 2, \ldots, N_j\}$ **do**
```
    /* Replicate a virtual node for compute matching                    */
```
    $\mathbf{R}'$.Insert(r_j^k);
for $e_i \in \mathbf{E}$ **do**
  for $r_j \in \mathbf{R}'$ **do**
```
    /* Compute score for each helper-helpee pair                        */
```
    $S_i^j \leftarrow$ ComputeDistScore(e_i, r_j) + ComputeIntfScore(e_i, r_j);
```
/* Calculate the maximized weighted b-matching results.                 */
```
$A =$ Hungarian_WBM(E, R, S)

3.3.4 Timely Delivery of Detection Results

As vehicles need to digest the point cloud data in real-time, it is vital to deliver the remote detection results back to each vehicle in time. Outdated detection results may not be useful because the driving environment can change drastically within seconds. To achieve timely delivery of the detection results, Harbor harnesses both application-level deadline awareness and MAC-layer prioritization.

3.3.4.1 Deadline Awareness

To cope with the fast-changing network conditions, vehicles in Harbor perform point cloud encoding adaptation. Harbor uses an existing bandwidth estimation technique [228] to perform upload bandwidth estimation, based on which the encoding level is adjusted accordingly.

However, only applying encoding adaptation is inadequate. If the server waits for frames from all vehicles (generated in the same round) to arrive before performing computation, the E2E latency would be inflated by stragglers whose frame uploading is too slow. Based on our measurement, frame upload time can differ by over 0.75s across 6 vehicles due to the above reason. Harbor addresses this latency heterogeneity by incorporating a deadline by which frame merging and detection must start, even if not all frames from the current round have been received.

Specifically, Harbor's server determines this deadline in the following way. Each vehicle has a fixed E2E latency requirement T_{e2e} (§4.4) for each point cloud frame since the frame

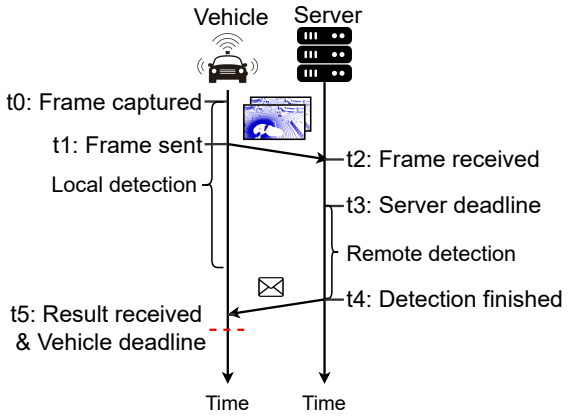


Figure 3.6: Timeline of `Harbor`'s vehicle-side and server-side data processing.

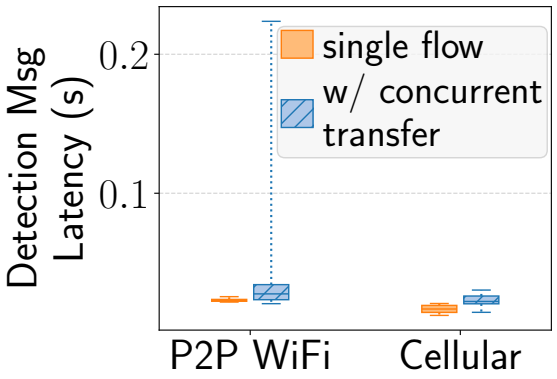


Figure 3.7: Impact of network types on the detection result delivery latency.

is captured from the sensor. Recall from §3.3.1 that in the design of `Harbor`, vehicles still run local detection in parallel and use the local detection if remote results are not received after T_{e2e}. Following the timelines in Figure 3.6, the server can calculate the vehicle-side deadline $t_5 = t_0 + T_{e2e}$. All the vehicle-side timestamps are embedded into control messages sent to the server (§3.3.1). The server then estimates the downlink one-way delay T_{owd} to get $t_4 = t_5 - T_{owd}$. Finally, based on its detection time T_{remote}, the server computes the deadline timestamp $t_3 = t_4 - T_{comp}$ at which it must start merging all the received sensor data and running detection. Since there are multiple vehicles and their time parameters can differ, the server uses the earliest t_3 as the estimated deadline. In `Harbor`, vehicles use NTP to synchronize time within ten milliseconds [76], which is sufficient for our application.

Local Merging. When a detection result arrives at a vehicle, the vehicle will first check if the result arrives too late to be used – specifically, whether the result is delivered beyond T_{e2e} since the upload of the sensor data. Any late-arrival result will be discarded. The vehicle maintains a local deadline (different from the edge-side deadline); when the local deadline is missed, the vehicle will fall back to using its local detection result. Next, the vehicle will check if the edge-returned result uses the view uploaded by the vehicle itself. The result is accompanied by a small bitmap to indicate which vehicles' views are processed by the edge and henceforth used in the result. If a vehicle's corresponding bit is missing (due to missing the edge-side merging deadline), the vehicle will perform *local merging* by merging its local detection with the edge-returned result. The merging is local in that the merged view is only used locally, and the vehicle (as a helper) will not further distribute it downstream; instead, a helper always passes the original result from (and signed by) the edge. This helps reduce the attack surface and prevent the erroneous situation where a single vehicle's detection failure pollutes the edge's merged view. If the edge's view passes both checks, it will be used by the vehicle.

3.3.4.2 MAC-layer Prioritization

Different from previous work [228] where only a cellular V2I downlink is used to transmit result messages, in a hybrid CVP involving both V2V and V2I links, sensor data (uplink) and detection result (downlink) compete for transmission in opposite directions over the same V2V wireless medium. Also, their bandwidth consumption is highly imbalanced, with the uplink data dominating the bidirectional traffic.

We find such imbalance can significantly inflate the end-to-end detection: when bidirectional traffic is present, overwhelmed by the uplink traffic, the downlink traffic is allocated with little network resources (Figure 3.7).

This thus requires prioritizing the downlink traffic that contains the detection results. We find that performing traffic prioritization at the *application* layer on a helper vehicle does not work. This is because when the application layer of a helper (as a relay) perceives the uplink traffic from the helpee (the origin), the traffic has already saturated the wireless medium and thus starved the downlink traffic.

To address the above challenge, we propose a MAC-layer message prioritization design that prioritizes different message types using priority queues. For queues with higher priorities, we adjust the MAC-layer parameters (§4.4) to enable a higher probability of accessing the medium. This design allows the lightweight but latency-sensitive detection result messages to be prioritized over data-intensive sensor data transmission, achieving lower E2E latency while maintaining the network throughput for sensor data transmission. Note that the MAC-layer prioritization module reduces the downlink delay for delivering the detection results and is orthogonal to the helper assignment algorithm which optimizes the collaborative data uploading. We validate the effectiveness of MAC-layer message prioritization in §3.5.5.

3.4 System Implementation

Server. The server performs helper assignment, sensor data merging, and downstream detection. `Harbor`'s drivable space detection uses the RANSAC [82] algorithm to extract the road plane and mark the points on the road plane as "drivable" (other points are marked as objects). Then it converts the road plane into an occupancy grid of $1m \times 1m$ and labels each grid as either "drivable", "occupied" or "unknown". `Harbor`'s object detection adopts PointPillars [120] as the backbone object detection neural network. We use the pre-trained models provided by OpenCOOD [171] for detecting objects.

Vehicle. V2V control message exchanges are implemented over UDP due to the broadcast requirement. V2V sensor data exchanges and V2I communications are implemented

over TCP. We use the OLSR routing Protocol [68] in V2V network and Draco [11] to encode/decode point cloud data. We apply distance-based cropping: each vehicle crops the point cloud data within a fixed threshold of 50 m. Applying more sophisticated data partition algorithms [228] is easy. We implement the MAC-layer prioritization in Linux kernel [13]. We create a higher priority queue for delivering detection results in addition to the regular MAC-layer frame processing queue. We adjust the contention window ($CW_{min} = 2$) and arbitrated inter-frame spacing ($AIFS = 1$) to increase the probability of successfully transmitting the detection result packets. We derive a 500 ms latency threshold T_{e2e} by considering several safety-critical scenarios from previous work [228, 163]. This latency threshold is on par with that defined in a previous V2I-based CVP (EMP [228]) plus the edge inference and result delivery time not considered by EMP.

3.5 Evaluation

We evaluate `Harbor` and compare it with several state-of-the-art solutions under realistic traffic scenarios and network conditions. We quantify improvements in end-to-end latency and accuracy using trace-driven emulations (§3.5.2). We show that CVP empowered by `Harbor` can improve perception accuracy and driving safety in real-world driving scenarios (§3.5.3). We demonstrate the system scalability and robustness under large number of vehicles and fast-changing topologies in §3.5.4. Lastly, we present a system overhead study in §3.5.5.

3.5.1 Experimental Setup and Methodology

We consider the following evaluation metrics.

• **Object detection accuracy**: It quantifies the difference between predicted object bounding boxes and the ground truth. We use Average Precision (AP) [2] as the metric to evaluate object detection. It is calculated with a threshold of Intersection over Union (IoU) [6] to evaluate the accuracy of object detection. The AP metric counts prediction bounding boxes with IoU higher than the threshold as true positives. We set the IoU threshold as 0.5, a value commonly used for evaluating point cloud object detection [120, 177, 210].

• **Drivable space detection accuracy**: It quantifies the difference between the actual detection result and the ground truth. We compare the predicted grid labels with ground-truth labels. Detection accuracy is defined as $Accuracy = \frac{TP+TN}{\text{Total number of grids}}$. TP (True Positives) is the number of grids whose prediction and ground-truth are both drivable, while TN (True Negatives) is the number of grids whose prediction and ground truth are both occupied or unknown.

Table 3.2: Summary of experiment settings.

Experiments	# of CAVs	Speed (km/h)	V2I BW (Mbps)	Traffic Scenes	Point cloud data type
Emulation	2 - 100	0 - 70.0	5.4 - 55.9	roundabout, road segments, intersections, entrance ramp	CARLA
Mcity Testbed	3	0 - 32.3	3.3 - 35.7	crossroads, T-intersections, road segments	Real-world

• **Detection latency**: It measures the time elapsed from when a point cloud is captured on a vehicle to when the perception result (*e.g.,* drivable space detection and object detection) is ready to use by downstream applications.

We compare `Harbor` with the following state-of-the-art solutions. **EMP** [228]: a V2I-based cooperative perception system where each vehicle uses V2I to upload LiDAR data and vehicles without V2I connection cannot join the collaboration. **AVR** [163]: a system that uses V2V to send 3D point clouds (same data as `Harbor`) from other vehicles to a randomly chosen leader vehicle for sharing.

CarSpeak [118]: a V2V system where each vehicle broadcasts sensor data to all other vehicles for perception. We also implemented its loss-resilient compression technique. For all schemes, the vehicles use detection results described in §3.3.4.1.

Trace-driven emulation. We conduct large-scale trace-driven emulation using traffic and network traces. Our traffic traces consist of vehicle trajectories and LiDAR point clouds collected from the CARLA simulator [26] in various driving scenarios and traffic topologies (Table 3.2). We create nodes in Mininet-WiFi [14] and replay the trajectories to emulate the V2V network. Our V2I network traces are LTE and 5G uplink throughput traces collected from real-world driving [152, 228]. The V2I traces are assigned to each vehicle based on a uniform distribution. We use a Linux machine with a 16-core CPU and 32GB memory to run Mininet-WiFi, which creates various numbers of wireless nodes and a server node. Each wireless node runs a `Harbor` vehicle instance and has two network interfaces: an 802.11g WLAN for communication between vehicles as it is widely used by previous work [194, 175], and an Ethernet interface for connection with the server. We use Linux *tc* to replay our network traces over the Ethernet interface to emulate the V2I network. The avg (stddev) bandwidth of V2V links is measured to be 12.35 (5.68 Mbps) under the emulated 802.11g setup. The point cloud capture rate is set to 10 fps in emulations and real-world experiments.

Real-world CAV testing. We also deploy 3 Lincoln MKZ vehicles as CAVs on four realistic driving scenarios at an autonomous driving testbed, *Mcity* [27]. Our testbed is a

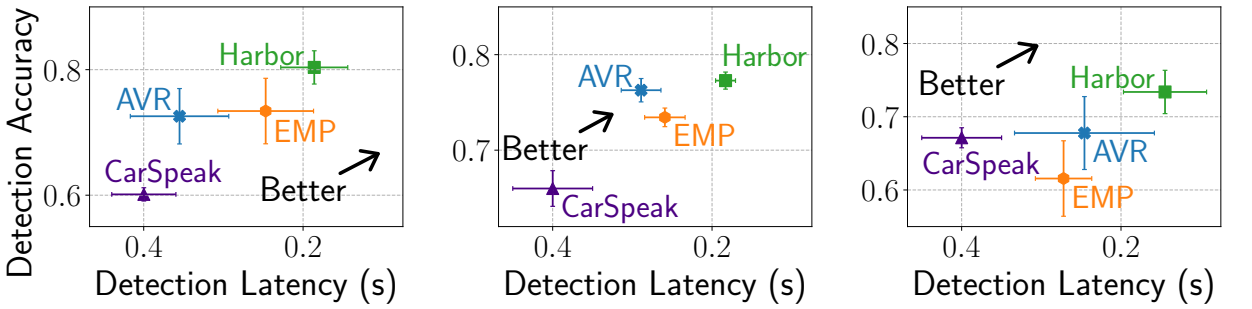


Figure 3.8: Emulation experiment results for E2E drivable space detection performance of `Harbor` and baselines.

real-world mock city for testing CAV applications, which consists of road segments, traffic lights/signs, and testing vehicles. Each vehicle is equipped with OxTS RT3000v3 GPS [34], Velodyne VLP-32C LiDAR [29], and Cohda MK6C OBU [23] for V2V+V2I communication. The same machine in emulation is used as the edge server. We also deploy several other non-connected vehicles as targets for detection. We run experiments on 4 different driving scenarios and run our evaluation on a total of 1602 LiDAR frames.

Real-world vehicle driving test. To examine Harbor under realistic V2V and V2I network conditions on a larger scale, we run real-world field tests by driving 6 cars on the road. Each vehicle is equipped with a laptop that runs a Harbor vehicle instance to replay sensor data from CARLA. In this case, although vehicles still use synthetic sensor data, they experience realistic V2V and V2I network conditions as real-world driving. We use the same server machine in emulation experiments to run the Harbor server instance. For each laptop, we tether a smartphone to it. The smartphones have GPS and each runs an app [16] to fetch locations. The cellular data plans used include AT&T [10], Verizon [19] and T-Mobile [17]. The laptops form an ad-hoc network using their WiFi interfaces for V2V communications. While technologies like DSRC [111] are designed for V2V communication, they often require specialized hardware and programming interfaces. Therefore, we choose to use ad hoc WiFi for V2V because of its compatibility with existing Linux OS as previous work [118]. In real-world CAV tests and vehicle driving tests, the bandwidth statistics (average/standard deviation) of the V2I (V2V) networks are 19.98 $\pm$ 9.08 (10.41 $\pm$ 1.48) Mbps based on our measurements.

3.5.2 Trace-driven Emulation Results

We first show the latency and accuracy performance of `Harbor` by running emulation experiments on the CARLA dataset with 2 - 20 CAVs under various traffic densities and network

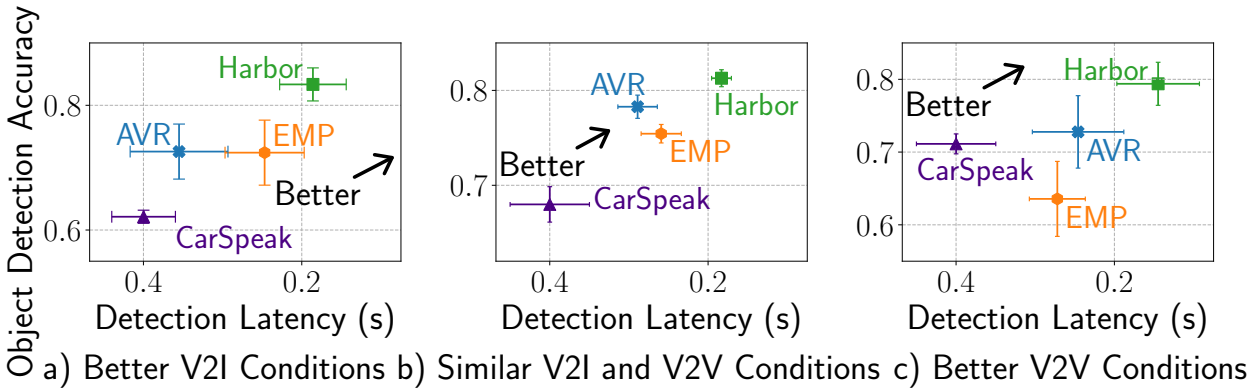


Figure 3.9: Emulation experiment results for E2E object detection performance of `Harbor` and baselines.

conditions (Table 3.2). We measure both the detection accuracy and latency overhead for each scheme.

We vary the number of vehicles, mobility patterns, and V2I network conditions, which create **100** different settings in total. We further categorize these settings into 3 classes, based on average V2I bandwidth, V2V distances and the duration of helpee vehicles: (1) better V2I conditions, (2) similar V2I and V2V conditions, and (3) better V2V conditions.

Figure 3.8 and Figure 3.9 show the End-to-end (E2E) performance of `Harbor` and baseline schemes using drivable space detection and object detection as the perception applications. In the aggregated results, we show the mean latency on the x-axis and mean detection accuracy on the y-axis. The error bar shows the standard deviation of each scheme under various settings. As shown, `Harbor` dramatically improves the detection latency. The mean latency is improved by 39.3% (38.5%) and 36.7% (34.7%) compared to AVR and EMP on drivable space detection (object detection), respectively. The improvements over CarSpeak are even higher: 57.1% (54.2%). `Harbor` also improves the detection accuracy compared to the baselines. Specifically, on average, it improves the accuracy by 3.5% (7.1%), 7.8% (7.6%), and 12.1% (11.6%), compared to AVR, EMP, and CarSpeak on the two apps. The reasons for the benefits are two-fold. First, Harbor tries to bridge more vehicles together despite their V2I/V2V disconnections by jointly using their V2V and V2I links, leading to a more complete view. Second, Harbor reduces the E2E latency of remote detection results delivery, making the results more likely to be delivered in time and hence utilized. It is expected that the latency improvements of the two apps are similar as the only difference affecting the E2E latency between the two is the computation time difference.

Table 3.3: Collaborative perception accuracy under different collaboration schemes.

Traffic Scene	**Object Detection Acc./Drivable Space Detection Acc.**			
	Local-only	EMP	AVR	`Harbor`
Testbed-Overall	42.67/40.77%	69.84/57.89%	69.90/52.53%	**82.08/70.67**%
- Right turn	28.91/33.28%	59.60/53.84%	42.66/49.59%	**77.93/71.44**%
- Left turn A	34.60/44.57%	81.16/59.79%	79.49/57.25%	**83.55/63.87**%
- Left turn B	35.94/44.48%	79.50/58.12%	61.53/55.86%	**81.30/72.89**%
- Lane merge	71.25/40.78%	72.52/59.81%	80.46/47.43%	**87.57/74.51**%

3.5.3 Performance on Real-world Testbed

In our real-world CAV testbed *Mcity*, we create four traffic scenes (Figure 3.11) that are challenging based on the NHTSA pre-crash scenario typology [151]: (1) an unprotected right turn in a crossroad; (2) an unprotected left turn in a crossroad (Left turn A); (3) an unprotected left turn in a T-intersection (Left turn B); (4) a lane merging from a parking space.

Under the four real-world driving scenarios, we first evaluate the end-to-end system performance to demonstrate how `Harbor` benefits CVP tasks after data sharing. Table 4.1 summarizes the perception accuracy of both object and drivable space detection tasks of different schemes. Overall, `Harbor` improves the object detection (drivable space detection) accuracy by 2% - 18% (4 - 18%) compared to EMP, 2% - 36% (6 - 27%) compared to AVR, and 16% - 49% (19 - 38%) compared to local-only. Specifically, Figure 3.10 depicts a specific scene to show how `Harbor` outperforms the baselines. In this scenario, the target vehicle and other vehicles in the scene are consistently detected by `Harbor`, while AVR does not detect the target because limited V2V bandwidth of one CAV to the ego vehicle. While EMP detects the target vehicle, it misses another vehicle in the opposite lane because another CAV does not upload its data on time to the edge server due to limited V2I bandwidth. The benefits of `Harbor` in perception accuracy come from the fact that `Harbor` bridges vehicles with limited V2I performance by judiciously using the spare V2I resources from other nearby CAVs, resulting in a larger coverage of 3D data of the environment.

End-to-end driving outcome. To better understand the driving outcome of each scenario, we deploy the latest Baidu Apollo Driving Software [22] to analyze the driving decision by applying `Harbor` (Table 3.4). As depicted in Figure 3.11, EMP fails in 2 of the 4 challenging cases (1 collision and 1 near-collision). The target vehicle is occluded hence undetected by the ego vehicle. Furthermore, the pure V2I scheme also misses the detection from the server side because other CAVs do not upload their sensor data in time due to poor V2I conditions. In contrast, under all 4 scenarios, `Harbor` can make prompt decisions, including reducing speed and stopping to yield, to avoid the potential conflict of

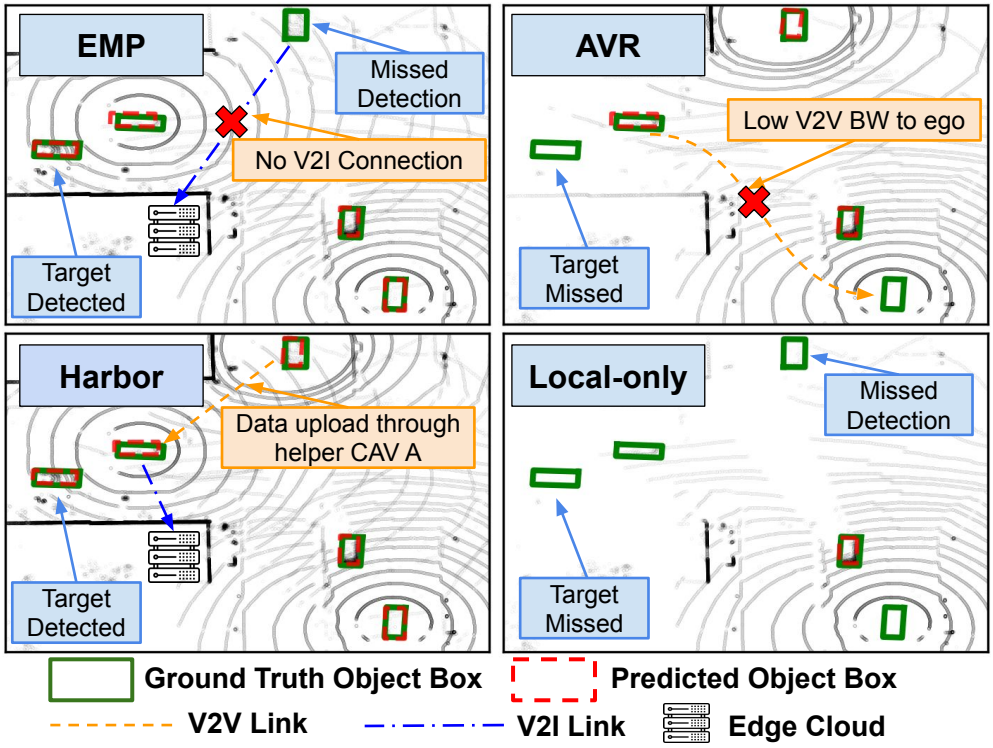


Figure 3.10: Object detection results of various schemes at *Mcity*. The scenario photo is shown in Figure 3.11.

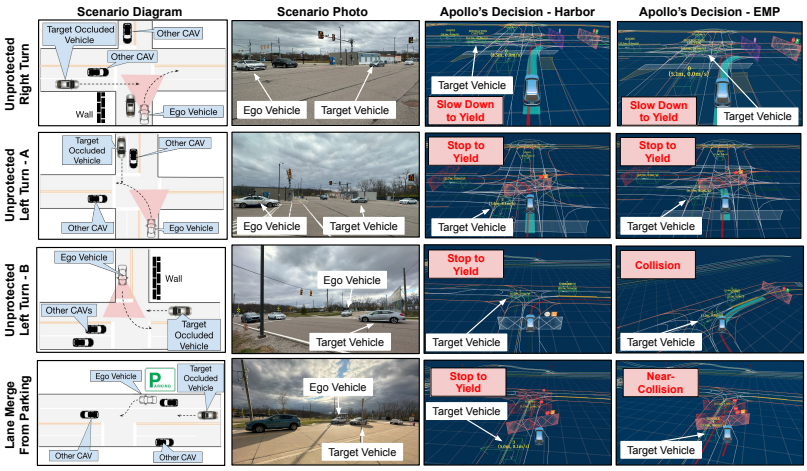


Figure 3.11: End-to-end driving outcome using Baidu Apollo on real-world scenarios.

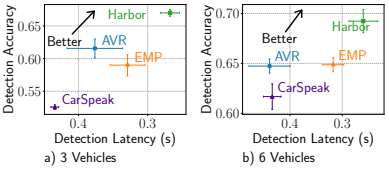


Figure 3.12: E2E live vehicular experiments.

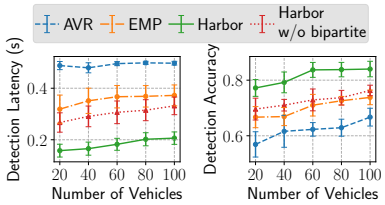


Figure 3.13: Performance of `Harbor` w/ large number of CAVs.

its planned driving trajectory with the target occluded vehicle thanks to the collaboration to detect the target vehicle earlier in the scene. We also perform analysis on the additional reaction time (*i.e.*, how much earlier the ego vehicle detects the target) for the ego AV to make driving decisions compared to local-only perception (Table 3.4). While existing

Table 3.4: Additional driving reaction time (compared to Local-only) and outcome of different collaboration schemes.

Traffic Scene	**Additional Reaction Time (s)/Driving Outcome**		
	EMP	AVR	`Harbor`
Testbed-Overall	+ 0.76/1 crash	+ 0.58 /1 crash	**+ 1.26/0 crash**
- Right turn	+ 0.69/safe-pass	+ 0.41/crash	**+ 1.60/safe-pass**
- Left turn A	+ 1.05/safe-pass	+ 0.20/safe-pass	**+ 1.23/safe-pass**
- Left turn B	+ 0.73/crash	+ 1.13/safe-pass	**+ 1.42/safe-pass**
- Lane merge	+ 0.57/near-miss	+ 0.57/near-miss	**+ 0.79/safe-pass**

Notes: A near-miss occurs when the ego and target vehicle pass within 3 m of each other.

collaboration methods also increase the reaction time for the ego by detecting the target earlier, `Harbor` results in an average of 0.5 - 0.7 seconds (65%) more reaction time compared to the baselines.

Field driving test with more vehicles. To showcase the real-world impact of `Harbor` for more vehicles, Figure 3.12 plots the field test results: Harbor outperforms EMP, AVR, and Carspeak by reducing 18.6%, 29.9% and 37.7% of detection latency and improving 8.0%, 5.78% and 11.0% on detection accuracy. As the number of vehicles increased from 3 to 6, Harbor's performance remains consistently the best, showing better scalability. In real-world driving tests, we have observed that the RTT between commercial cellular network to a server is 97.4 $\pm$ 1.49ms, much longer than a real vehicle-to-edge communication latency (about 20 ms [231]). Compared to emulation experiments, Harbor's improvements in latency are slightly lower due to higher base V2I latency.

3.5.4 Scaling to Large-scale Deployment

We evaluate how `Harbor` scales to environments with a larger number of vehicles. Following the vehicle density and speed information from 2 public vehicle trajectory datasets [15, 241], we place up to 100 vehicles in the CARLA simulator and use trace-driven emulation to replay their trajectories and data for evaluation. Note that as the total number of vehicles increases, both the number of helpers and helpees increase.

Figure 3.13 shows how the performance of `Harbor` changes with an increasing number of vehicles. Harbor outperforms EMP (AVR) by 37.1% (51.3%) in average detection latency, and 11.3% (19.4%) in detection accuracy, respectively. We observe that applying a V2I-based scheme is indeed more scalable than a pure V2V scheme, as shown before [228]. `Harbor` achieves better scalability than V2I by using the V2V medium for helpees and applying bipartite optimization to adapt to network dynamics. Note that `Harbor` not only improves the detection latency but also accuracy compared to `Harbor` without bipartite optimization. This is because the bipartite optimization enables faster computation of assignments and

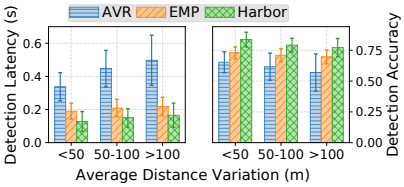


Figure 3.14: Performance of `Harbor` under various levels of helper topology changes.

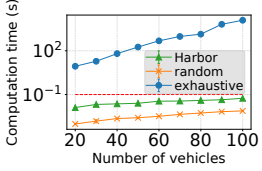


Figure 3.15: `Harbor` assignment overhead.

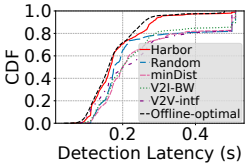


Figure 3.16: Benefits of `Harbor`'s assignment strategy.

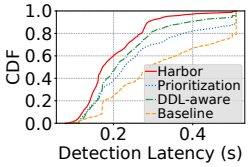


Figure 3.17: Benefits of detection result delivery strategy.

henceforth reacts faster to network changes, resulting in more data being delivered to the edge server and used for perception.

Robustness of `Harbor` under fast changing topologies. Next, we analyze the robustness of `Harbor` under variable traffic topologies. To quantify the degree of topology change, we use a metric calculated from vehicle trajectories: the average relative distance variation. A higher variation of the relative distance indicates that the relative position of ego-CAV varies more compared to others, making the topology more fast-changing. The average relative distance variation is derived as follows. (1) For each CAV, we calculate the standard deviation of its relative distance to all the other CAVs over the entire trajectory every t seconds (we set $t = 0.1$). (2) We then average the standard deviations from all CAVs by the number of vehicles. We conduct this experiment using 50 emulated vehicles. Figure 3.14 shows the performance of `Harbor` under various levels of topology changes. Under slower changing traffic topologies (average relative distance variation ¡50m), `Harbor` outperforms AVR (EMP) by 32.6% (61.2%) in latency and 10.7% (18.3%) in accuracy. Even under the fastest-changing traffic conditions (average relative distance variation ¿100m), `Harbor`'s improvements over AVR and EMP only reduce slightly, despite the excessive mobility's impact on V2V network performance, thanks to Harbor's robust helper assignment algorithm.

3.5.5 System Overhead & Microbenchmarks

We study the overhead and benefits of design decisions made by `Harbor` from three perspectives: (1) the control-plane and data plane message overhead, which measures control/data message size, its transmission latency and the corresponding bandwidth requirements; (2) the runtime performance of `Harbor`'s helper assignment algorithm and its benefits to the E2E detection latency; (3) the latency improvements made by timely delivery of detection results strategy. For ease of presentation, we use drivable space detection as the example application for the remaining part of the evaluation.

Table 3.5: Control-plane and data-plane message overhead for each point cloud frame per vehicle(mean±std deviation).

Metrics	AVR	EMP	`Harbor` (Helper/Helpee)
Control data (Bytes)	24	24	67±5.5 / 67±5.5
Transmission Latency (ms)	28.1±24.6	27.6±18.6	27.9±17.8 / 32.4±22.0
Bandwidth Req. (Mbps)	<0.1	<0.1	<0.1 / <0.1
Sensor data (KB)	42.9±5.07	46.2±3.5	45.9±3.2 / 40.6±7.1
Transmission Latency (ms)	158.3±31.2	91.5±19.9	36.3±19.1 / 72.1±22.2
Bandwidth Req. (Mbps)	3.4±0.5	3.7±0.3	3.6±0.3 / 3.2±0.6

Notes: *Bandwidth Req.* - Bandwidth Requirement at 10 FPS.

3.5.5.1 System Overhead Analysis.

Control-plane and data-plane message overhead. Table 6.4 compares the control plane and data plane message overhead between `Harbor` and other existing V2I/V2V-only CVP solutions. Apart from the vehicle location data also used in EMP and AVR, `Harbor`'s control message further includes V2I bandwidth measurements and V2V routing tables. The additional data size in the control plane is negligible (fewer than 50 bytes per point cloud frame). Therefore, `Harbor`'s control-plane message overhead is similar to EMP and AVR, as mentioned in §3.3.1. Also, as the last row in Table 6.4 suggests, helpee vehicles in `Harbor` have slightly higher transmission latency for control messages. This is attributed to its hybrid (V2I+V2V) network topology. For data messages, all three schemes upload similar size of point cloud data (after encoding), while `Harbor`' transmission latency is significantly lower than EMP and AVR because `Harbor` judiciously allows helpee vehicles to use the spare V2I bandwidth from helpers. Vehicles with poor V2I conditions perform poorly on EMP as they use V2I-only communication. AVR suffers from higher data-plane latency than EMP due to less scalable V2V-only communication, as shown in [228]. For bandwidth requirements, `Harbor` requires the same level of uplink bandwidth as V2I-based systems, while achieving better latency and scalability.

Overhead of `Harbor`'s assignment algorithm. We evaluate the runtime performance of `Harbor`'s helper-helpee assignment algorithm by comparing it with two baselines: (1) **Random**: randomly assign helpers to helpees. (2) **Exhaustive**: perform brute-force search to find the optimal assignment based on `Harbor`'s formulation. We consider two metrics: the assignment execution latency (Figure 3.15) and the end-to-end detection latency (Figure 3.16). As shown in Figure 3.15, the scheduling time of the exhaustive search grows quickly as the number of vehicles increases – taking more than 10s to compute the assignment for 20 vehicles. On the other hand, as shown in Figure 3.16, while the random scheme provides decisions within 100ms (the LiDAR generation cycle), its end-to-end detection latency is

high, due to its selected suboptimal paths. In contrast, Harbor with its built-in bipartite optimization can efficiently perform helper assignment within 100ms (Figure 3.15). This, together with the resulting judiciously determined paths, leads to an average end-to-end detection latency of 175ms even for 100 vehicles, outperforming all the other baseline schemes (Figure 3.16).

3.5.5.2 Microbenchmarks.

Strategic Helper Assignment. We examine the benefits of one of Harbor's strategic helper assignment through trace-driven emulation. To make a fair comparison, we modify only the assignment scheduling logic in Harbor to have five different baseline assignment schemes: (1) **Random:** randomly assign helpers to helpees; (2) **Min-distance:** assign helpers by minimizing the total distance between each helper-helpee pair; (3) **V2I-BW:** assign helpers by selecting the helpers with higher V2I bandwidths; (4) **V2V-intf:** assign helpers by maximizing the interference score on the V2V network; (5) **Offline-optimal:** an offline-generated solution by examining all possible assignments at each scheduling interval with the best performance.

Figure 3.16 shows the benefits of Harbor's strategic helper assignment leveraging different information sources. The 90^{th} and 95^{th} percentile latency have improved by 43.8% and 38.8% compared to the baselines. As the baseline assignment algorithms often optimize a single factor, they fail to perform well in all settings. By considering all different factors that affect V2I and V2V data transmission, Harbor becomes more robust under various driving and network conditions, thus greatly improving the tail latency. Furthermore, Harbor achieves 90^{th} and 95^{th} percentile tail latency that is 91.8% to 92.8% close to the offline-optimal scheme.

Timely Detection Result Delivery. We compare Harbor's fast detection result strategy with three baseline strategies: (1) **Baseline**: Harbor without the entire fast detection result delivery design (§3.3.4); (2) **Prioritization**, which adds the MAC layer prioritization on top of the baseline; (3) Deadline-aware (**DDL-aware**), which incorporates server-side deadline-awareness in frame merging to the baseline. Figure 3.17 shows the effectiveness of Harbor's decisions in fast detection result delivery. Both MAC-layer result message prioritization and server-side deadline awareness reduce the average detection latency by over 19.0%. Harbor achieves a higher latency improvement of 38.3% by leveraging both techniques. Harbor also improves 90^{th} percentile latency by 25.6% (42.6%), compared to Prioritization (DDL-aware). While the MAC-layer result message prioritization module necessitates modifications to the lower layers of the operating system's network stack, its demonstrated effectiveness suggests that future integration with existing CAV hardware platforms (e.g., NVIDIA DRIVE AGX [32]) could enable its practical deployment in current autonomous

vehicle systems. Such integration would allow this optimization to enhance real-world CVP solutions by improving data transmission efficiency and reducing latency in various situations.

3.6 Discussion and Related Work

Security and Privacy of Collaborative Perception Systems. While beyond the scope of this work, security problems might arise in collaborative vehicular perception systems. Malicious vehicles can send fake/modified data to affect the collaborative detection results [188]. Under the context of `Harbor`, both infrastructure and vehicles can take actions to improve system security: 1) Edge server can leverage public key infrastructures (PKI) [190] to issue certificates to authorized vehicles and revoke access whenever anomaly behavior is detected. 2) On vehicles, Trusted Execution Environment (TEE) [99] can be deployed to help prevent data spoofing and modification from attackers. At the application level, `Harbor` can leverage existing multi-vehicle collaboration defense method [226] to check sensor data consistency.

Hybrid V2V+V2I Vehicular Communications. [155] leverages V2V network to speed up V2I file uploading to the roadside unit for a single source vehicle. [192] proposes the V2X protocol for optimizing network message propagation by allowing protocol switching between V2V and V2I. Different from previous work in this space, CVP imposes new challenges in that data uploading from different vehicles needs to be jointly optimized, *i.e.,* collaboration of multiple data traffic, to reduce perception latency. Our system `Harbor` focuses on assisting 3D CVP tasks by strategically utilizing V2V and V2I network resources. The design of `Harbor` optimizes the overall delay for all nodes participating in the collaboration and jointly handles bandwidth-intensive uplink and latency-sensitive downlink transmissions.

Cooperative Vehicular Perception. Various efforts have been made on cooperative vehicular perception. Existing data sharing systems either transfer sensor data or processed features between vehicles [163, 62, 118, 198, 164] or directly send each vehicle's data to a server [228, 117, 138]. Our work advocates using both V2V and V2I to better adapt to different network connectivity and bandwidth conditions. We develop solutions for the dynamic establishment of V2V and V2I channels, and algorithms that efficiently assign relay vehicles considering the wireless network and physical properties. As mentioned in §4.2, while we focus on CVP data-sharing at the raw-data level, the underlying principle of `Harbor` can be applied to collaborative perception designs that share intermediate features. We left the integration of intermediate feature-based sharing with `Harbor` as future work.

Collaborative Mobile Systems. `Harbor` is partially inspired by mobile systems involving network-interface-level or device-level collaborations [157, 238, 46, 110, 108]. They focus on

various applications such as smartphone file download and video streaming. In contrast, our work targets the CVP domain, and thus faces unique challenges as described earlier (§3.1).

Peer-to-peer Wireless Multicast Systems. Collaborative vehicular perception tasks involve sharing data among multiple vehicles or agents in a dynamic wireless network. While `Harbor` focuses on combining V2V and V2I for better CVP performance, peer-to-peer (P2P) multicast technologies can potentially further enhance the efficiency of such data exchanges. Innovations in wireless multicast protocols [79, 205, 124] demonstrate that multicast can improve bandwidth utilization and reduce communication overhead in scenarios with multiple participants. These technologies are particularly relevant for CVP systems like `Harbor`, where multiple vehicles need to access the same control data and perception results. Incorporating peer-to-peer multicast strategies could further reduce the latency of data dissemination, especially in environments with dense vehicular traffic. Future extensions of `Harbor` could leverage these technologies to optimize data sharing between helper and helpee vehicles, further enhancing the system's scalability and efficiency.

3.6.1 Summary

`Harbor` bears several limitations. First, due to its opportunistic nature, `Harbor` is best-effort, without any service quality guarantee or bound. Second, our design and evaluation focus on a single edge node's service area (with multiple subareas supported though, see §3.3.1). How to scale up `Harbor` to the city level requires further research. Third, our current implementation only incorporates basic security measures such as end-to-end encryption for point cloud upload's confidentiality and edge's signature for edge-returned results' integrity. More efforts (*e.g.,* multi-vehicle collaboration defense [226]) are needed to defend various attacks in hybrid CVP.

Despite the above limitations, as a first-of-its-kind hybrid CVP system, `Harbor` considerably outperforms existing V2V- and V2I-only schemes, as backed up by extensive field tests and large-scale emulation. We envision deep collaborations among CAVs, edge/cloud, smart road infrastructure, and even unmanned aerial vehicles (UAVs) to be a key enabler of next-generation transportation systems. Our work makes an important step towards this ambitious goal.

CHAPTER 4

Robust Real-time Multi-vehicle Collaboration on Asynchronous Sensors

4.1 Introduction

Connected and autonomous vehicles (CAVs) rely on a variety of advanced on-board sensors (*e.g.,* GNSS, camera, Radar, and LiDAR) to localize their ego pose in relation to local road elements (*e.g.,* lanes, traffic signs), as well as to detect other traffic participants. However, the effective range of sensors can be limited. For example, off-the-shelf LiDAR sensors have an effective range of about 80 meters [29], beyond which the resolution of sensor data (*i.e.,* point cloud) decreases drastically. To this end, collaboration among CAVs as well as roadside infrastructure can enhance the perception capability of individual CAVs by extending the perception range and covering occluded areas. Through shared sensor data, each vehicle is able to broaden its perception of the surrounding environments, make more accurate predictions, and thus plan its driving behaviors more optimally.

Cooperative perception [113, 62, 198, 228, 164, 69, 210, 209, 101] is gaining traction in the research community in recent years. One challenge is system scalability. The bandwidth usage of sharing raw sensor data is not sustainable, even with state-of-the-art wireless communication technologies. If naively sharing full frames of high-fidelity data, the estimated bandwidth usage exceeds 300 Mbps [228], which is far beyond the capacity of current vehicular networks (*e.g.,* C-V2X at tens of Mbps) [211, 193]. To tackle this problem, one could first partition the raw point cloud into regions and then only transmit the most relevant regions. We call such approach *Partition and Selective Sharing*. For instance, EMP [228] shares close-range points while AutoCast [164] shares the points of objects with low visibility and high relevance to planning.

Though successful in addressing scalability, this selective sharing approach faces two fundamental challenges: (1) During data fusion, a vehicle's local sensor data and data from remote vehicles (or infrastructure) are asynchronous, or captured at different times, due to

different sensor configurations on both sides and unpredictable wireless transmission delay. The time gap between two frames can be larger than 100 ms [21, 178], leading to inaccurate data merging and damaged performance of downstream perception tasks. (2) The indirect estimation of occluded areas can be inaccurate. Particularly, instead of utilizing a vehicle's direct and more accurate insight of its own occluded areas, both EMP and AutoCast obtain location estimates by either inquiring others or using simple heuristics. Without knowing the exact locations and shapes of obstacles blocking the views, such practices may fail to share critical occluded areas.

In this paper, we focus on improving cooperative perception by tackling the challenges caused by asynchronous sensor data and inaccurate occlusion localization. To achieve this, we develop *RAO*, a Real-time Asynchronous Occlusion-aware multi-vehicle cooperative perception system, in which the merged sensor data consistently and accurately cover the occluded areas (*e.g.,* blind spots of vehicles), without incurring additional processing latency. While we use LiDAR as an illustrative example to represent generic 3D geometric features in surrounding environments, *RAO* can also be applied to other sensors like stereo camera or Radar.

We develop event-driven, asynchronous protocols for LiDAR sensors, enabling four separate point cloud processing and sharing tasks to be conducted in parallel without introducing additional delay: (1) When a vehicle gets a local LiDAR image, it generates and shares an occupancy map, which is a 2D map that labels free areas and occupied areas on the road. (2) Simultaneously, in each LiDAR scanning cycle, each vehicle sends data requests to ask remote vehicles or infrastructure for LiDAR data. The data requests are optimized to maximize the quality of the merged point cloud. (3) Upon receiving the data requests, the data is prepared and sent back to the corresponding vehicles. (4) Once a vehicle receives a local LiDAR sensor image, it merges the image with requested sensor data and then conducts downstream perception tasks, such as object detection and tracking.

To address the challenge of merging asynchronous sensor data, we design a novel motion-compensated occupancy flow prediction method, suitable for challenging automotive scenarios featured with high vehicle mobility as well as asynchronous and noisy data. Our prediction method provides sufficient resilience against asynchronous data in multi-vehicle perception collaboration by leveraging motion information calculated from previous frames. Moreover, to tackle the challenge caused by incomplete occlusion estimation, we take an ego-centric approach that allows a vehicle to determine its own occluded areas because the vehicle has direct and more accurate knowledge of these areas. Compared to prior art that relies on remote vehicles' indirect estimation, our solution enables the merged sensor data to cover more occluded areas with improved perception accuracy.

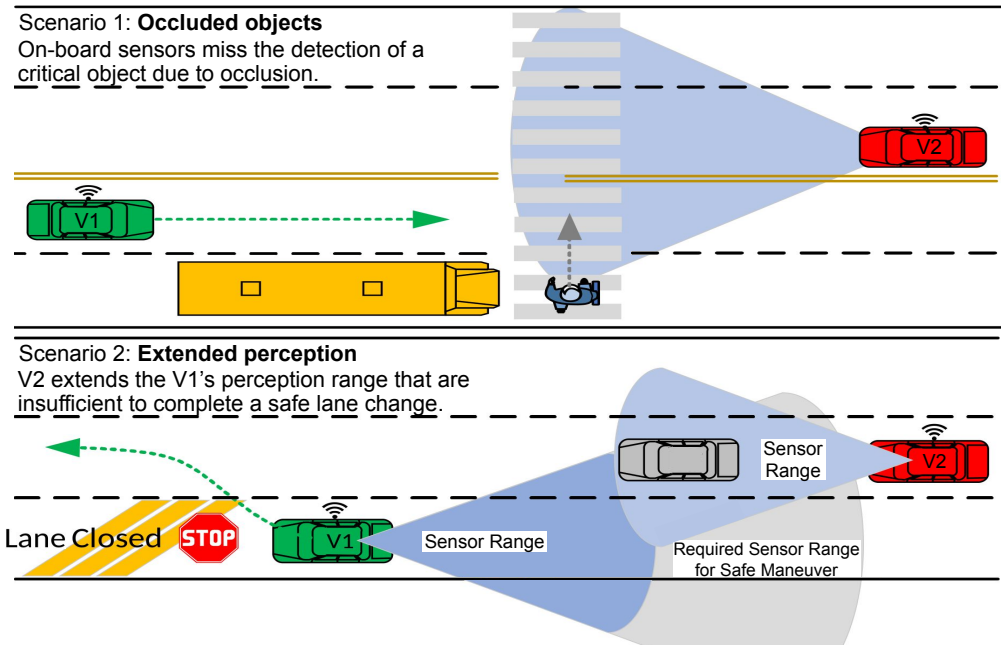


Figure 4.1: Two scenarios where cooperative perception helps.

To evaluate *RAO*, we implement a prototype and run the system in a realistic setup where vehicles transmit and process LiDAR data asynchronously. Utilizing three datasets collected from both real-world and photo-realistic synthesized driving scenarios, we show that *RAO* yields a 34% and 14% improvement over state-of-the-art solutions in perception coverage and accuracy, respectively. *RAO* is lightweight; it only introduces an extra latency of 8.5ms to the perception process on the consumer side and generates a data volume of 10.9 KB per frame, on average. Extensive experiments demonstrate that *RAO* performs well across a rich variety of map topologies and driving scenarios.

The key contributions of this work are as follows:

• We identify the challenges of synchronization and inaccurate occlusion localization in practical cooperative sensor sharing and analyze their impacts on cooperative perception.

• We design and implement *RAO*, a real-time multi-vehicle cooperative perception framework, with robustness against asynchronous data and efficient data sharing scheduling.

• We propose innovative designs of occupancy flow prediction and on-demand data sharing that can efficiently synchronize LiDAR point clouds captured at different timestamps, enabling a seamless multi-vehicle data alignment and fusion.

• Through systematic experiments on two large-scale datasets and one field test, we demonstrate that *RAO* brings significant performance improvements to perception accuracy and coverage with minor system overhead.

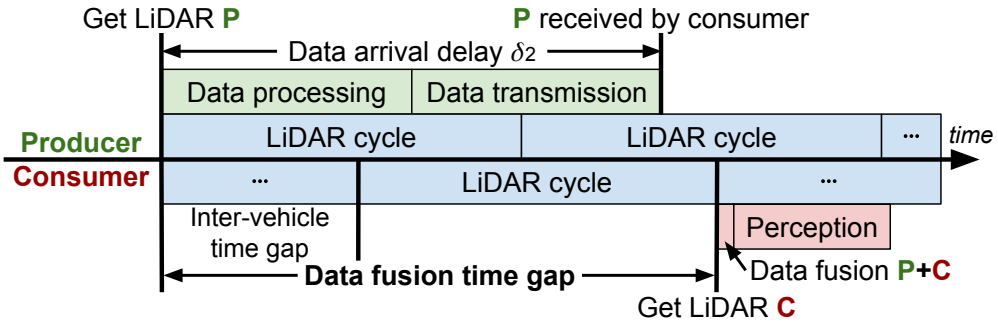


Figure 4.2: Synchronization problem in protocol diagram of cooperative perception.

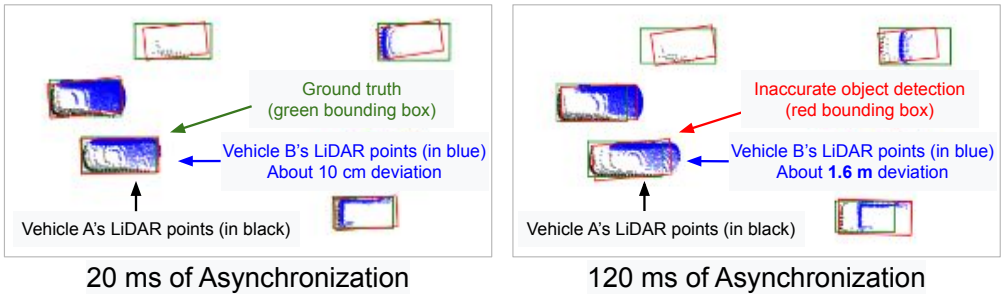


Figure 4.3: Merged point clouds from two vehicles (A and B) at various levels of synchronization (Bird-eye view).

4.2 Motivation

In this work, we focus on early-fusion cooperative perception [62, 198, 164, 210], where vehicles share LiDAR data via a wireless network and process data locally. Operating on raw sensor data enables flexible use of merged data. Efficient data sharing is made possible with mature network infrastructure and proper data partitioning. However, existing system proposals have severe limitations, including inaccurate data fusion due to the asynchronous nature of multi-vehicle data sharing and insufficient shared data due to incomplete occlusion detection. We discuss these issues in detail in §4.2.1 and §4.2.2, and address them by achieving the design goals itemized in §4.2.3.

Terms. We refer to a vehicle that receives sensor data as a *consumer*. Others, such as vehicles and roadside infrastructure, that share sensor data with consumers are *producers*. Note that a vehicle can play both roles simultaneously.

4.2.1 Asynchronous Sensor Data

Existing early-fusion cooperative perception proposals merge asynchronous sensor data, but few address the resulting inaccuracy problems. To this end, we break down the synchroniza-

tion problem and analyze the temporal order of events in cooperative perception depicted in Figure 4.2.

Inter-vehicle time gap. Vehicular LiDAR sensors producing data periodically (the cycle is 100 ms for most commercial LiDAR sensors [29]). However, the capture time of LiDAR images on different LiDARs are not synchronized, leading to inevitable gaps between timestamps of LiDAR images from every two vehicles. For example, in Figure 4.2, the producer gets a LiDAR image from its local sensor at timestamp 0 ms, while the consumer's next image is generated between 0 ms and 100 ms. For each pair of a producer and a consumer, we define the *inter-vehicle time gap* as the time difference between the generation of the producer's LiDAR image and the generation of the consumer's next LiDAR image. Note that this time gap is primarily attributed to the configuration of LiDAR devices thus it cannot be mitigated by traditional clock synchronization methods such as NTP [148].

Data arrival delay. Even though a producer sends its LiDAR data, the data does not arrive at the consumer immediately. The data undergoes pre-processing (*e.g.,* partitioning, compression), transmission, and post-processing (*e.g.,* decompression) before the consumer can use it. As presented in previous works [62, 198, 228], the entire processing duration rarely falls below 100 ms, and thus cannot be ignored. We define the *data arrival delay* as the interval between a producer generating local LiDAR data and a consumer receiving processed, ready-to-use data.

Due to the above factors, a consumer may not have the producers' latest LiDAR data when generating its own local data. However, to minimize the latency of downstream perception tasks, most existing cooperative perception solutions opt to immediately perform data fusion and perception upon receiving their local LiDAR data. In this way, since LiDAR data fusion is a simple process of merging 3D points, these systems introduce little additional latency to perception tasks compared with single-vehicle perception.

The question is how dated the producer's data can be in such systems. Formally, given a producer and a consumer, we denote the LiDAR cycle time (*e.g.,* 100 ms) by T, the *inter-vehicle time gap* by δ_1, and the *data arrival delay* by δ_2. By definition, δ_2 inevitably contributes to the data fusion's time gap. There is also a time gap between data arrival and the consumer generating its next LiDAR image, $(\delta_1 - \delta_2 \bmod T)$. Therefore, the total time gap of data fusion is:

$$\delta = \delta_2 + (\delta_1 - \delta_2 \bmod T), 0 \leq \delta_1 < T, \delta_2 > 0. \tag{4.1}$$

Ideally, the producer's data arrives at the moment of the consumer's data fusion. In this case, $\delta_1 - \delta_2 \bmod T$ is zero, and δ is equal to δ_2. In contrast, in the worst case, the producer's data arrives right after the data fusion. δ goes up to $\delta_2 + T$.

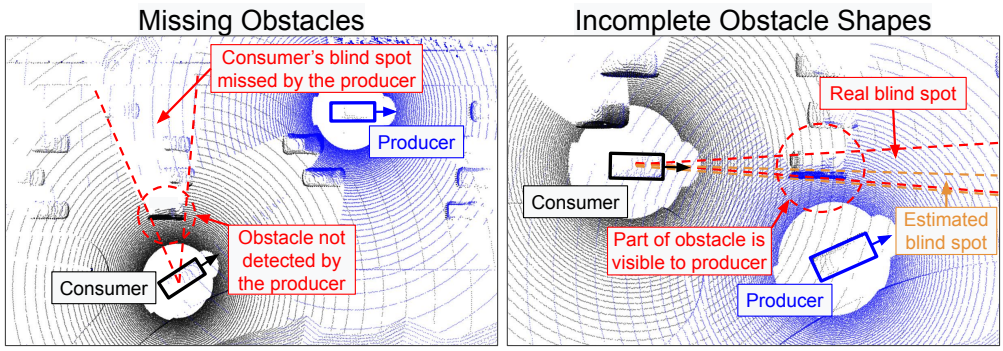


Figure 4.4: Inaccurate blind spot estimation (the consumer's/producer's points and heading in black/blue).

Such synchronization problems are not explicitly discussed in existing cooperative perception proposals. False tolerance mechanisms in conventional robotic perception may not be ideal for CAVs, primarily due to the intricate driving scenarios and the strict latency requirements. For instance, registration algorithms (*e.g.,* ICP) are used in multi-robot SLAM [77] to calibrate generated maps. This approach spatially aligns maps of stationary objects but cannot address asynchronous time frames. Alternatively, other approaches [168, 90] employ motion models (*e.g.,* kinematics models) to forecast the behavior of moving objects. However, obtaining such motion models can be challenging for remote unidentified vehicles on the road. For connected vehicle applications, AutoCast [164] reports that both its data processing and transmission are barely below 100 ms. As a result, when the inter-vehicle time gap is close to 100 ms, the worst-case data fusion time gap is almost 300 ms according to Equation 4.1, during which a 60 km/h vehicle can shift its position by 5 meters. As shown in Figure 4.3, objects can hardly be recognized in the merged point clouds without proper synchronization. V2VNet [198], a deep learning model for intermediate-fusion cooperative perception, uses neural networks to compensate for the time gaps. It states that synchronization on raw point clouds is nontrivial, and therefore is not implemented in its evaluation. VIPS [178] tolerates the asynchronization by aligning bounding boxes from different CAVs, which is effective only in object-level sharing schemes. Autonomous vehicles [22, 21] commonly use high-definition (HD) maps for accurate positioning but maps alone cannot address the asynchronization problem as the errors come from inconsistent timestamps instead of inaccurate localization. To the best of our knowledge, there is no existing work addressing the early-fusion synchronization challenge.

4.2.2 Incomplete Occlusion Estimation

To reduce transmission latency, early-fusion cooperative perception schemes usually shrink the size of shared LiDAR data. Besides compression and sampling, previous works [163, 164]

propose sharing only critical regions, *i.e.,* blind spots[1] of others. This is based on the insight that improving sensor data quality in well-observed regions does not greatly benefit downstream tasks. For instance, in EMP, vehicles share LiDAR data in close areas, based on Voronoi Diagram [48]. In AutoCast, producers estimate the blind spots of consumers based on the consumers' views. Specifically, producers in AutoCast acknowledge the location of consumers, localize on-road objects, and finally perform simple ray casting to "guess" consumers' blind spots occluded by these objects. Thanks to the blind spot estimation, AutoCast can cover more areas after data fusion than EMP. However, the estimated occluded areas can significantly differ from the ground truth. We analyze the sources of inaccuracies as follows.

First, each producer's perception range is limited and thus may not observe all obstacles blocking consumers' view. Without knowing the specific obstacle location, the provider can hardly identify the area behind that obstacle; in contrast, the consumer has accurate knowledge of its own occluded areas. As shown in Figure 4.4, one occluded area of the consumer, over 50 m^2, is missed by the producer because the key obstacle causing the occlusion is unknown to the producer.

Second, producers cannot recognize the complete shape of obstacles, and thus the ray-casting method is sometimes inaccurate in identifying occluded areas. In Figure 4.4, the producer observes a key obstacle that may block the consumer's view but can only see one side of the obstacle without knowing the depth. As a result, when the producer uses the partial obstacle for ray casting, it implicitly assumes that the consumer can "see-through" the rear part, causing the estimated occluded areas to be much narrower than the ground truth.

Due to these inaccuracies, producer-centric occlusion estimation often misses actual occluded areas of consumers, degrading the benefits of data sharing.

4.2.3 Design Goals

In building *RAO*, we aim to solve the above problems while achieving real-time robust cooperative perception.

• **Robustness against asynchronous data.** Under realistic driving scenarios where vehicles share asynchronous sensor data, *RAO* should produce accurate perception results.

• **High occlusion coverage.** A large proportion of occluded areas should be filled with sensor data from remote vehicles.

[1]We define blind spots as areas that cannot be directly perceived by a vehicle itself, caused by occlusion or long distance. We also use the terms *blind spots* and *occlusion* interchangeably.

• **Real-time protocol.** The execution of *RAO*'s protocol should meet the real-time requirements of real-world vehicle perception systems. To be specific, each perception cycle should be finished within 100 ms [135].
• **Negligible additional latency for perception.** Compared to single-vehicle perception, *RAO* should not cause any observable delay for vehicles to get perception results.

4.3 System Design

We describe how *RAO* satisfies each design goal outlined in §4.2.3. After shedding light on how to overcome the limitations of prior work in §4.3.1, we present key components and the general workflow of *RAO* in §4.3.2. We then elaborate the detailed design of three critical components: occupancy map generation, occupancy flow prediction, and data scheduling, in §4.3.3, §4.3.4, and §4.3.5, respectively.

4.3.1 System Overview

To tackle the inaccuracy problem caused by asynchronous data (§4.2.1) and incomplete blind spots (§4.2.2), we propose two novel solutions, *occupancy flow prediction* and *on-demand data scheduling*, in our design of *RAO*.

Occupancy flow prediction. Although sensor data items arriving at consumers are inherently asynchronous, one can still manage to synchronize the received point clouds from different producers using an intelligent prediction mechanism. While there is a rich body of literature on scene flow prediction that predicts the subsequent frame of a LiDAR image based on previous frames [122, 128, 87, 72], these algorithms are not suitable for our targeted applications. They fail to generate results sufficiently fast for real-time perception, with most requiring near or over 100 ms. Also, we aim to synchronize point clouds to any given future timestamp to compensate for the arbitrary time gaps between LiDAR images, while existing algorithms are frame-wise and can only synchronize point clouds at fixed intervals.

To achieve real-time prediction, we simplify the prediction problem by considering the uniqueness of CAV scenarios: the data asynchronization is mainly reflected on objects (*e.g.,* vehicles) moving on a surface (*e.g.,* ground). Therefore, unlike conventional approaches that focus on independent LiDAR points, we propose an efficient prediction algorithm that only tracks non-ground clusters of points. Using our mechanism, upon receiving a local LiDAR image in each frame, the vehicle clusters the LiDAR points to localize on-road objects and estimates the motion of clustered objects by comparing the corresponding clusters across consecutive frames. In this way, whenever synchronization is needed, even if the received data frame is missing or delayed, the consumer can leverage prediction to compensate for

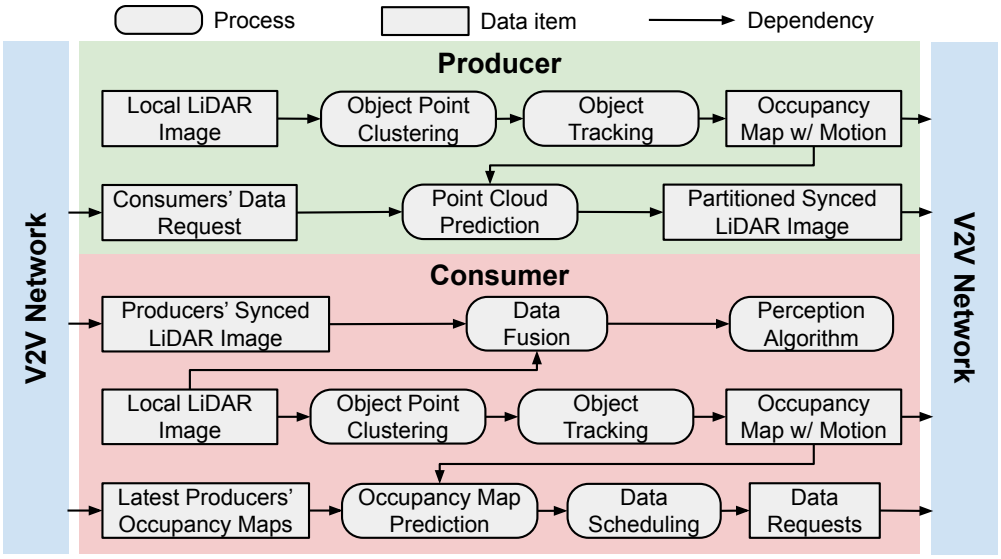


Figure 4.5: System Architecture of *RAO*.

arbitrary time gaps.

On-demand data scheduling. To detect blind spots, we argue that consumers themselves have the most accurate and reliable information because they can readily recognize areas with few or even no LiDAR points and mark these areas as occluded areas. Inspired by this insight, we advocate that consumers should proactively request data on specific areas from producers, which is a departure from current mainstream approaches such as EMP [228] and AutoCast [164] where producers have to "guess" consumers' blind spots. The advantages of this consumer-centric approach are twofold: (1) It improves the accuracy of blind spot estimation since the consumer knows the best; and (2) proactive data requests by consumers prevent duplicate data shared by multiple producers, thus eliminating unnecessary bandwidth usage.

4.3.2 System Components

In *RAO*, the producers and consumers are connected by vehicular wireless networks that allow for the exchanges of sensor data and control messages. *RAO*'s protocol is event-driven, meaning that protocol tasks are executed by producers and consumers asynchronously. As shown in Figure 4.5, the execution of protocol tasks is triggered by certain events, either a new data frame being generated or a prior process being completed. The workflow of *RAO* can be divided into four separate tasks: occupancy map generation, data request generation, data response generation, and data fusion.

Occupancy map generation is responsible for producing occupancy maps, in which

on-road objects, free-to-drive areas, and invisible areas are labeled in the 2D space. For each vehicle, after its LiDAR sensor generates a local LiDAR image, this component immediately starts to generate an occupancy map and broadcasts the map once finished. The occupancy map generation process involves several steps: (1) *RAO* runs LiDAR segmentation to separate LiDAR points into ground points and object points. (2) Then our clustering algorithm assigns the object points into separate groups of objects, and the object tracking algorithm applies correlation on clusters across consecutive frames. (3) To estimate the object motions, which are later used for occupancy flow prediction, *RAO* leverages 3D point cloud registration to obtain accurate transformation of each cluster over two consecutive frames, from which motion parameters (*e.g.,* velocity) are extracted. (4) Finally, the occupancy maps along with their corresponding tracks and motion parameters, are bundled together and broadcast over the V2V network.

Data request generation is executed on consumers to determine what data requests should be sent to producers. Once the consumer generates its local occupancy map, this process is triggered. An essential part of this process is *data scheduling*, which assigns needed LiDAR data that covers the identified blind spot areas to available producers. To achieve high-quality fusion of LiDAR data, the producers' LiDAR data should have a reasonably high resolution (*i.e.,* density of points). To achieve this goal, a consumer first (1) synchronizes different occupancy maps on both the consumer and producer sides to the timestamp of its next LiDAR cycle using our *occupancy flow prediction* method; (2) Then the consumer solves an optimization problem to assign each needed blind spot area to the producer who can provide the best quality of sensor data on that area. (3) The consumer sends out a sequence of requests to corresponding producers.

Data response generation is executed on producers and is triggered by receiving consumers' data requests. For each received data request, the producer fetches its latest locally generated LiDAR image and partitions it into the interested areas per request. Then the producer synchronizes the partitioned image into the timestamp requested by the consumer and sends it back to the consumer. This synchronization step, again, leverages our *occupancy flow prediction* method.

Data fusion is the final process that merges available LiDAR images from different vehicles, before feeding the generated outcome to downstream perception modules. Once a consumer vehicle gets its local LiDAR image, it immediately collects LiDAR data shared by other producers, transforms all LiDAR images to the same coordinate system, and then combines them. No additional data processing is needed since the producers have already executed partitioning and synchronization tasks, delivering the transformed data. Thanks to its simplicity, *RAO* introduce no observable latency onto downstream modules.

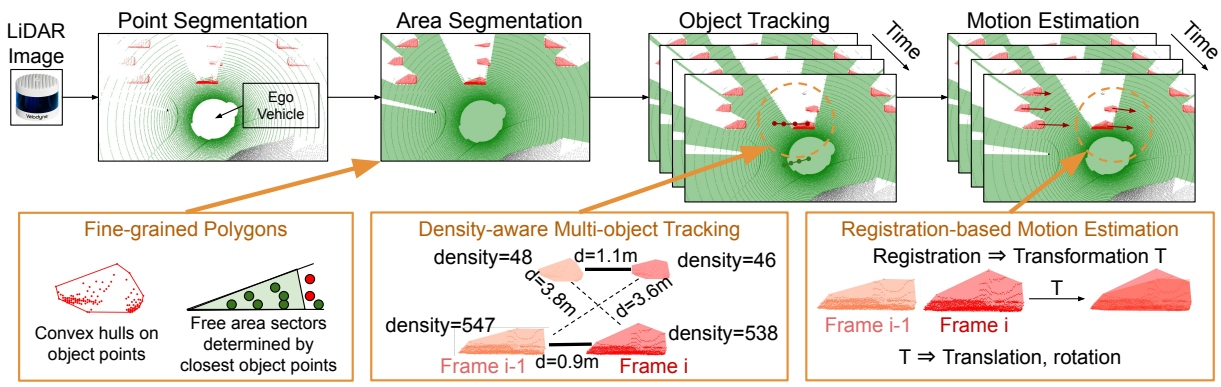


Figure 4.6: Demonstration of the generation of motion-aware occupancy maps.

Note that *RAO* is a framework built on the fundamental CAV architecture. We assume the CAVs can actively discover nearby CAVs and establish bi-directional V2V connections via C-V2X or DSRC. The CAVs also communicate their poses and locations (GPS/IMU), which are required for transforming LiDAR data to the same coordination system.

In the remainder of this section, we elaborate on our designs of three major mechanistic components in *RAO*.

4.3.3 Motion-aware Occupancy Map

This component generates an occupancy map, which not only labels the location of on-road objects but also tracks their mobility trajectory and their motion parameters. Existing deep learning-based perception solutions are not suitable for generating occupancy maps because they require significant computing resources and serve different goals. In *RAO*, the occupancy map acts as metadata frequently shared among CAVs, in order to guide efficient scheduling of overall data sharing. To this end, occupancy maps need to be lightweight in computation and communication overhead. In contrast, perception algorithms based on deep learning, such as 3D object detection and tracking, are performed on the merged sensor data after the data fusion. Occupancy maps indirectly benefit those expensive perception algorithms by improving the quality of data fusion. We introduce the generation of occupancy maps as follows.

LiDAR point segmentation. We first remove less useful background points that fall outside of the road regions, leveraging HD maps provided by autonomous driving systems [22, 21]. Then, we use existing ground detection algorithms [82, 159] to detect the ground plane and remove LiDAR points on the ground. By clustering the remaining points, we can identify all the non-ground objects on the road, with each cluster representing a unique on-road object. This method has been proven to be effective in prior art [75, 204, 223].

Area segmentation. After identifying on-road objects, we generate a fine-grained representation of 2D space occupancy, which classifies the 2D space into three categories: *occupied*, *free*, and *occluded*. (1) Occupied areas are generated by computing the convex hulls [49] of the object clusters. (2) Free areas represent the region containing only ground points. As demonstrated in Figure 4.6, we split the 2D space evenly into sectors whose vertex is the LiDAR location (the number of sectors is configurable for various granularity). In each sector, we measure the distance from the LiDAR location to the closest non-ground point and label the area within the distance as a free area. (3) Occluded areas are our key interests in cooperative perception. They constitute the areas that are neither free nor occupied and can be readily derived by subtracting the occupied and free areas from the entire area. Figure 4.6 illustrates the 2D area segmentation on LiDAR point clouds obtained in real-world environments.

Unlike conventional grid-based occupancy maps [112, 164, 125], our occupancy map splits areas using polygon representation. This approach offers two significant benefits over grid representation: (1) polygons can represent arbitrary shapes with better precision; (2) by adjusting the outline smoothing factor, polygon representation is more flexible to strike the right balance between reconstruction accuracy and data size. For instance, LiDAR points are too large, bounding boxes cannot represent arbitrary shapes, and grids are not data-efficient as a large empty area could involve lots of grids.

Motion estimation. In addition to on-road object detection, motion state is also needed in *RAO*. The traditional motion modeling-based approaches [217, 181] such as kinematic models cannot be used for occupancy maps, since (1) one can hardly obtain accurate motion parameters of on-road objects required by the motion modeling and (2) the shape of occupied areas alters over time due to ever-changing visual angles. To address these challenges, we leverage similarity metrics that are specific to point clusters (*e.g.,* the distribution of points), to estimate the motion of objects over frames.

In *RAO*, we design a density-aware multi-object tracking algorithm for point clusters. Inspired by AB3DMOT [201], a baseline for 3D multi-object tracking, we utilize the affinity matrix[2] for multi-object tracking and re-design the matrix as follows: Given any two objects (*i.e.,* occupied areas), we define the affinity score as a weighted sum of two items, (1) the distance between the mass centers of two clusters' convex hulls, and (2) the difference in point density as the number of points divided by the area of the convex hull. Convex hulls can be directly calculated from point clouds without the need for additional object detection algorithms, minimizing potential sources of inaccuracies. The affinity score of the two clusters

[2]The affinity matrix hosts scores of similarity between possible object pairs across two frames. Similar objects tend to be linked in the same track.

is given by Equation 4.2, where X_1 and X_2 denotes two clusters in consecutive frames and w_1, w_2 are coefficients.

$$f(X_1, X_2) = w_1 \cdot L_2(X_1, X_2)^{-1} + w_2 \cdot |\frac{\textsc{Size}(X_1)}{\textsc{Area}(X_1)} - \frac{\textsc{Size}(X_2)}{\textsc{Area}(X_2)}| \tag{4.2}$$

We then estimate an object's motion based on its point clusters in two consecutive frames, denoted by X_i^t and $X_i^{t'}$ where i is the object ID and t/t' ($t' > t$) are the timestamps of the two frames. We first select the mass center of the earlier occupied area as one reference point (it will act as the "center" of the object in the later perspective transformation process). Both point clusters are transformed in order to use this reference point as their origin. We denote the centering transformation by transformation matrix $T_{i,center}^t$. Formally,

$$Y_i^t = T_{i,center}^t \cdot X_i^t, \quad Y_i^{t'} = T_{i,center}^t \cdot X_i^{t'}, \tag{4.3}$$

where Y_i^t and $Y_i^{t'}$ are clusters centered by the reference point.

We then apply the point cloud registration algorithm [94] to discover a transformation $T_{reg}^{t'}$ that transforms Y_i^t to minimize its squared distance with $Y_i^{t'}$ as follow:

$$T_{i,reg}^{t'} =_T L_2(T \cdot Y_i^t, Y_i^{t'}). \tag{4.4}$$

Leveraging the rich features of 3D point distribution, the registration between Y_i^t and $Y_i^{t'}$ can accurately capture the pose change over two consecutive frames. Next, we normalize the transformation matrix $T_{i,reg}^{t'}$ to represent transformation per time unit. Specifically, we extract a translation vector and a rotation vector from $T_{i,reg}^{t'}$, and divide them by the time difference $(t' - t)$ to obtain translation and rotation per time unit. Then we reconstruct the transformation matrix per time unit as $T_{i,unit}^{t'}$. We refer to this operation as Scale:

$$T_{i,unit}^{t'} = \textsc{Scale}(T_{i,reg}^{t'}, \frac{1}{t' - t}). \tag{4.5}$$

We also update the reference point in timestamp t to t'. Despite on different timestamps, two reference points refer to the same position on the object itself as

$$T_{i,center}^{t'} = T_{i,center}^t \cdot (T_{i,reg}^{t'})^{-1}. \tag{4.6}$$

We store both $T_{i,center}^{t'}$ and $T_{i,unit}^{t'}$ as the estimated motion at timestamp t', which will be used in occupancy flow prediction (§4.3.4). This motion estimation task is executed iteratively when a new frame arrives. In *RAO*, we parallelize the motion estimation process of different objects in order to reduce processing latency.

4.3.4 Occupancy Flow Prediction

Occupancy flow prediction is designed for "synchronizing" data from different vehicles. A consumer can synchronize its available occupancy maps to a future LiDAR scanning cycle during data request generation. Similarly, during data response generation, a producer can synchronize the latest LiDAR image to the timestamp requested by the consumer.

Occupancy map prediction. *RAO* is able to predict the future state of the occupancy map at timestamp t', based on an existing motion-aware occupancy map at timestamp t, where $t' > t$. As discussed in §4.3.3, each occupied area is associated with a motion vector, containing the centering transformation matrix $T^t_{i,center}$ and unit-time transformation matrix $T^t_{i,unit}$, where i indicates the ID of the occupied area. Assuming the vertices of the occupied area i at timestamp t is X^t_i (occupied areas are represented by polygons), we can predict the location of such vertices at timestamp t' as:

$$X^{t'}_i = (T^t_{i,center})^{-1} \cdot \textsc{Scale}(T^t_{i,unit}, t' - t) \cdot T^t_{i,center} \cdot X^t_i. \tag{4.7}$$

We first centralize the occupied area by using the latest reference point as the origin and then apply the unit-time transformation. Finally, the occupied area being calculated is mapped back to the LiDAR coordination. Next, to predict the occluded areas. We calculate the area occluded by newly predicted occupied areas and compare it with the occluded area at timestamp t. We remove the occluded area different from the free area S_F and add the new occlusion into the occluded area S_C. Free areas can be updated similarly.

LiDAR image prediction. *RAO* can also predict a future LiDAR image at timestamp t' based on an existing LiDAR image along with the motion-aware occupancy map at timestamp t. The prediction assumes that the time difference $(t' - t)$ is as small as 100 ms (a LiDAR cycle) so that the LiDAR point clouds at two timestamps have a highly similar point distribution on any object. To achieve this, we first extract LiDAR points associated with each occupied area. Following the same approach outlined in Equation 4.7, we transform the points to the predicted locations and replace the original points in the LiDAR image with the predicted points.

4.3.5 Data Scheduling

Data scheduling assigns the occluded areas of a consumer to different producers capable of providing sensor data, which determines the efficiency of data sharing. We formulate it as an optimization problem with a goal to maximize the overall data quality for data fusion. The metrics of quality are customized by the consumer since different downstream tasks may value data quality from different perspectives.

Algorithm 2: Consumer data scheduling.

Input: Synchronized occupancy maps $M^{(i)} = (S_O^{(i)}, S_F^{(i)}, S_C^{(i)})$, $i \in \{0, 1, \ldots, N\}$, including occupied areas S_O, free areas S_F, and occluded areas S_C; $M^{(0)}$ is the consumer's map while others are producers' maps.

Output: A mapping assigns areas to producers: A.

$H \leftarrow$ MaxHeap;
$A \leftarrow$ HashMap;
for *Producer* $i \in \{1, 2, \ldots, N\}$ **do**
  for $S_{O,j}^{(i)} \in S_O^{(i)} S$ **do**
    $c_j^{(i)} \leftarrow \text{Score}(S_{O,j}^{(i)}, \cdot)$;
    MaxHeapAdd(H, $c_j^{(i)}$, $S_{O,j}^{(i)}$);
  end
end
while $H \neq \emptyset$ **do**
  $S_{O,j}^{(i)} \leftarrow$ MaxHeapGet(H);
  $S_C^{(0)} \leftarrow S_C^{(0)} - S_{O,j}^{(i)}$;
  HashMapAdd(A, $S_C^{(0)} \cap S_{O,j}^{(i)}$, i);
end

We propose to solve the optimization problem through a lightweight greedy search algorithm. As defined in Algorithm 2, data scheduling relies on occupancy maps from both the consumer and available producers. Prior to algorithm execution, all occupancy maps need to be (1) synchronized to the timestamp of the next LiDAR cycle using occupancy flow prediction (§4.3.4), and (2) transformed into the same coordination system using perspective transform (§4.3.2).

The scheduling algorithm is executed by the consumer and consists of the following steps: (1) For each occupied area in producers' occupancy maps, we calculate a customizable priority score and sort all producers' occupied areas in descending order of priority scores. (2) We process producers' occupied areas one by one, from higher scores to lower scores. For each occupied area, we calculate its intersection with the consumer's occluded areas. If an empty intersection is present, this occupied area is discarded; we then assign all the non-empty intersection areas to the producer. (3) To prevent sharing duplicate data from multiple producers, we remove the assigned area from the list of the consumer's occluded areas. We repeat steps (2) and (3) on all producers' occupied areas until the data assignment process is finished. Note that producers do not share ground points, which are in large size but make little contribution to perception.

We define the priority score as the distance between the producer's LiDAR and the occupied area. A shorter distance leads to a higher LiDAR point density, bringing richer features to downstream tasks. Note that the score definition is customizable for various purposes (*e.g.,* producer reputation).

4.4 Implementation

We build a prototype of *RAO* for the purpose of evaluation.

Emulator. We implement an emulator of asynchronous multi-vehicle cooperative perception in Python with 1037 lines of code (LOC). The emulator accepts pre-recorded sensor traces of multiple vehicles as input, including LiDAR point clouds and 3D poses of LiDAR sensors over consecutive frames. Note that the emulator is an event-driven system that executes data processing and data transmission asynchronously on multiple vehicle instances. The latency of data processing is directly measured by executing our algorithm implementations while the transmission latency is measured using recorded network traces.

Algorithms. We implement the core algorithms of *RAO*, including occupancy map generation, occupancy flow prediction, and data scheduling, with 870 LOC in Python and 5,053 LOC in C++. The C++ programs are algorithm implementations optimized to minimize the computation overhead. The Python implementation serves as an interface bridging the emulator and C++ programs. For ground detection, we use RANSAC [82]. For object tracking, we use parameters $w_1 = 1$ and $w_2 = 0.01$ for the affinity score and ICP [94] for point cloud registration. Data scheduling employs a distanced-based priority score as described in §4.3.5.

4.5 Evaluation

We conduct an extensive evaluation of *RAO* to demonstrate its performance under realistic traffic scenarios (including using real-world datasets). We answer the following questions to prove how *RAO* satisfies our design goals listed in §4.2.3: (1) How much performance improvement can *RAO* bring to downstream perception tasks on asynchronous sensor data when compared to state-of-the-art works? (2) What is the coverage of occlusion achieved by *RAO*'s perception sharing? (3) Is *RAO* fast enough to support real-time operations?

We first introduce our experimental setup (§4.5.1). Then we examine the end-to-end performance (§4.5.2) on a suite of large-scale multi-vehicle data sets (including both simulation and empirical traces) to analyze *RAO* (§4.5.3), followed by system overhead analysis (§4.5.5) and ablation study (§4.5.4). Finally, we conduct a case study in a real-world setup to show *RAO*'s advantages over the prior art.

4.5.1 Experimental Setup

Datasets. We evaluate *RAO* using three datasets: (1) a real-world vehicle-infrastructure dataset, DAIR-V2X-C [219], (2) a simulation-based multi-vehicle dataset, CARLA-SUMO,

and (3) a real-world V2V dataset collected at *Mcity* [27].

• *DAIR-V2X-C* is a large-scale dataset consisting of 40k asynchronous frames The data is captured by a vehicle-mounted LiDAR and an infrastructure LiDAR deployed at 28 intersections in Beijing. We further process it by selecting consecutive frames of vehicle and infrastructure data, with each scenario spanning over 10 seconds (*i.e.,* 100 frames).

• *CARLA-SUMO* is an asynchronous multi-vehicle dataset that we collect using OpenCDA [24] based on the co-simulation of CARLA autonomous driving simulator [26] and SUMO traffic simulator [25]. The dataset features diverse road scenarios including straight roads, interactions with two-way single and two lanes, and T-intersections. We set the vehicle speed to up to 50 kph, which is consistent with the average speed limit on urban roads [28]. It contains 2-5 CAVs and 10-50 non-CAVs in each scenario and a total of 6880 frames.

• *Mcity* is a mock city for road testing CAV applications in real-world setup, where we deploy 3 Lincoln MKZ vehicles (CAVs) equipped with OxTS RT3000v3 GPS, Velodyne VLP-32C LiDAR, and Cohda MK6C OBU as a C-V2X receiver. We create 8 challenging driving scenarios that involve occluded or far-away objects. For each scenario, we collect LiDAR, GPS, and C-V2X traces from all CAVs for 15 seconds.

Perception models. We adopt Pointpillar [120] as the backbone object detection neural network for perception-related tasks. For DAIR-V2X-C and CARLA-SUMO, we use pre-trained models provided by DAIR-V2X [219] and OpenCOOD [210], respectively. For *Mcity* data, we fine-tune the OpenCOOD model for 10 epochs to adapt it to *Mcity* sensors.

Baselines. We evaluate *RAO* against the traditional single-vehicle perception using local sensors (*Local-only*) and two state-of-the-art early-fusion cooperative perception solutions: (1) *EMP* [228], a V2I-based cooperative perception system where CAVs selectively upload LiDAR data to an edge node which executes perception algorithms. Based on the algorithm of Voronoi diagrams, each CAV shares LiDAR data in close range. (2) *AutoCast* [164], a V2V-based approach that optimally determines shared data areas and schedules data exchanges. Its data-sharing protocol is discussed in §4.2.

High-fidelity Emulation. We feed the sensor data into our *RAO* prototype (§4.4) and emulate the network conditions using real cellular network traces [152]. Specifically, the traces are collected from LTE uplink while driving on urban roads. The bandwidth statistics of the traces are 15.67 ± 10.38 Mbps. We conduct the experiments on a Linux machine equipped with a 32-core Intel Xeon CPU and an Nvidia 2080Ti GPU.

Table 4.1: Object detection accuracy under different multi-vehicle collaboration schemes and datasets.

Traffic Scene	**Perception AP@0.5/AP@0.7**			
	Local-only	EMP	AutoCast	***RAO***
DAIR-V2X-C	48.99/40.78%	48.82/40.68%	50.36/41.18%	**53.11/43.49%**
CARLA-SUMO	48.63/37.17%	64.08/54.26%	64.91/51.50%	**74.79/62.01%**
- Town05	40.68/30.18%	48.63/38.25%	63.61/39.88%	**69.81/58.72%**
- Town06	65.46/48.30%	73.22/53.22%	67.55/58.47%	**81.72/65.19%**
- Town10HD	40.12/32.58%	64.34/57.18%	69.90/52.50%	**78.53/65.25%**
Mcity	51.51/41.13%	64.88/50.50%	65.76/48.32%	**69.13/51.25%**

Table 4.2: Coverage of data sharing.

Overhead Metric	**Collaboration scheme**		
	EMP	AutoCast	***RAO***
Density	0.2132	0.1857	**0.2756**
Coverage	0.3352	0.2601	**0.3585**

4.5.2 End-to-end System Performance

Using a combination of simulation data and empirically collected traces, we first evaluate the end-to-end system performance of *RAO* to demonstrate how *RAO* benefits downstream perception tasks after data sharing.

Perception accuracy. We run 3D object detection (*i.e.,* perception) on the merged data produced by *RAO* and baseline schemes. We calculate Average Precision (AP) [2] with a threshold of Intersection over Union (IoU) to evaluate the accuracy of perception. IoU measures the overlap between predicted bounding boxes and ground-truth bounding boxes. A higher IoU value indicates more accurate object detection. The AP metric counts prediction bounding boxes with IoU higher than the threshold as true positives. We adopt IoU thresholds 0.5 and 0.7, which are commonly used in evaluating CAV perception. Table 4.1 summarizes the overall average precision supported by each scheme.

As shown in Table 4.1, *RAO* is able to achieve significant improvements in perception accuracy on all three datasets, compared to other schemes. Overall, *RAO* improves the AP at IoU over 0.5 (AP@0.5) by 4% – 30% compared to Local-only, 5% – 19% compared to EMP, and 4% – 14% compared to AutoCast. Among the three datasets, CARLA-SUMO demonstrates the best improvements brought by *RAO*. The goal of creating CARLA-SUMO was to complement DAIR-V2X-C's two-LiDAR setup with a larger number of CAVs and non-CAVs on the road in more diverse driving scenarios (§4.5.1), which is a realistic setting in large-scale deployment of cooperative perception. It explicitly exposes the challenges of synchronization and blind spot detection. We mainly focus on CARLA-SUMO for later analysis in §4.5.3, §4.5.4, and §4.5.5.

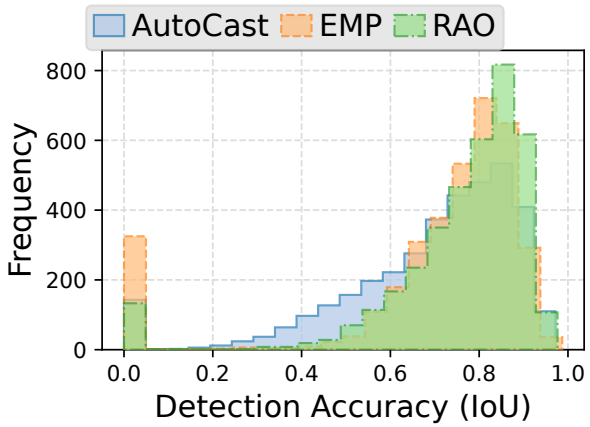


Figure 4.7: Distribution of perception IoU (accuracy).

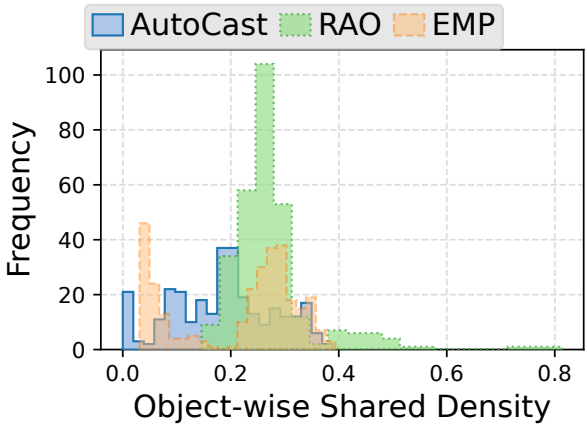


Figure 4.8: Distribution of object density in merged data.

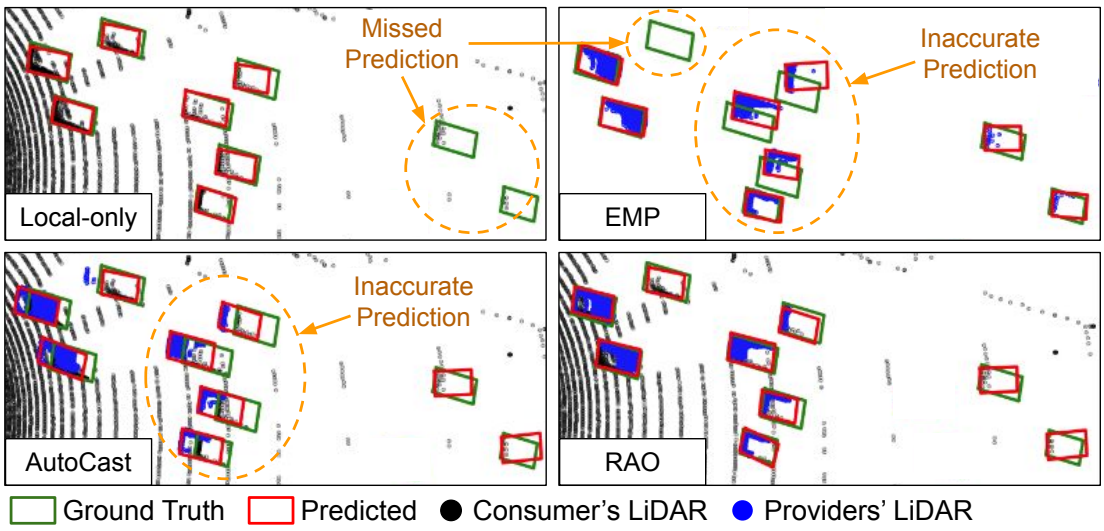


Figure 4.9: Synchronization of LiDAR data in various schemes.

To better understand how accurate the object detection is in *RAO*, we further plot the distribution of IoU values in Figure 4.7. In general, IoU is higher in *RAO*. Compared to AutoCast, *RAO* has significantly more accurate predictions (*e.g.,* IoU 0.7-0.9) and fewer inaccuracy predictions (*e.g.,* IoU 0.2-0.5). This is attributed to the following fact: when a less synchronized point cloud gets fused to the ego vehicle's data, it is likely to cause the predicted bounding boxes of an object to shift away from its ground-truth position. Such a shift significantly lowers the IoU and also the AP of object detection. Note that, the IoU values that fall in the leftmost boxes, are equal to zero for all schemes. These are caused by out-of-range objects.

Coverage of data sharing. *RAO*'s data scheduling is expected to cover blind spots more efficiently. To quantify the efficiency of data sharing, we utilize two metrics: the

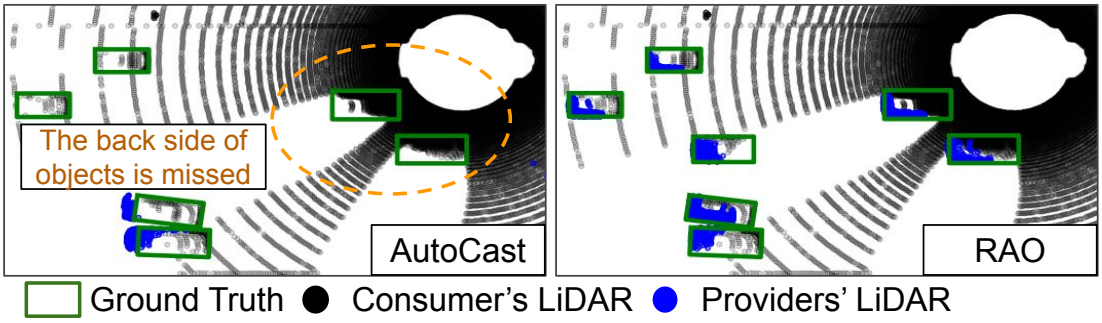


Figure 4.10: Coverage of blind spots in *RAO* and AutoCast.

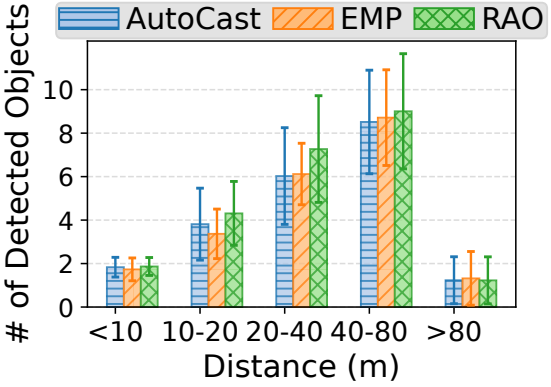


Figure 4.11: Number of objects detected based on distance.

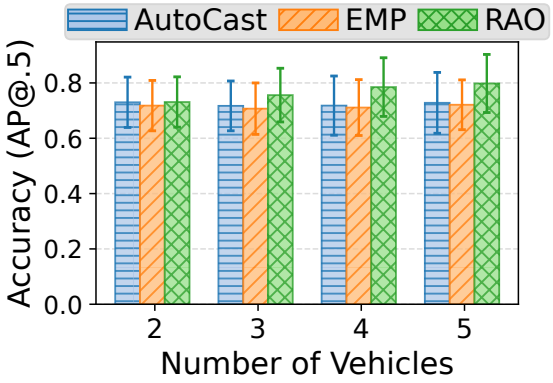


Figure 4.12: Perception performance *w.r.t.* CAV density.

density of objects and the coverage of objects. A higher point density potentially leads to a higher confidence score in object detection, improving the true positive rate. A higher object coverage, on the other hand, reduces the number of undetected objects, which means a lower false negative rate. To obtain such metrics, we associate each LiDAR point in each frame to a specific object using the ground truth labels (non-object points have no associated objects). Assuming a set of LiDAR images X are generated from different vehicles and they are finally merged to one single LiDAR image Y via the cooperative perception, for each object in the scenario, we can calculate the number of associated LiDAR points in X and Y as N_X and N_Y respectively. We define the metrics as follows, where M is the total number of objects.

$$Density = \frac{1}{M}\sum_{i}^{M} N_Y^{(i)}/N_X^{(i)} \tag{4.8}$$

$$Coverage = \frac{1}{M}|\{i \in \mathbb{N}, 1 \leq i \leq M \mid N_Y^{(i)} > 0\}| \tag{4.9}$$

Table 4.2 lists the results of the above metrics. Compared to AutoCast, *RAO* achieves a 50% higher object density score and a 34% higher object coverage. The density and coverage are also 29% and 7% higher than those of the edge-based EMP. In addition, Figure 4.8 sum-

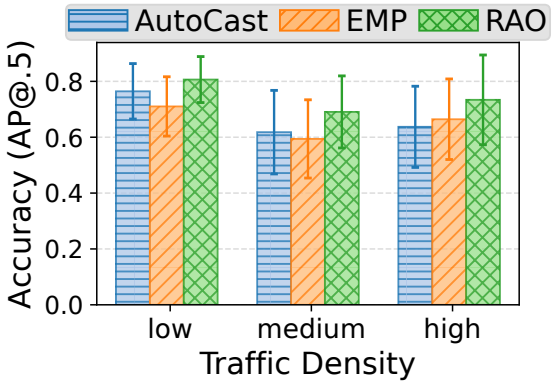


Figure 4.13: Perception performance *w.r.t.* traffic density.

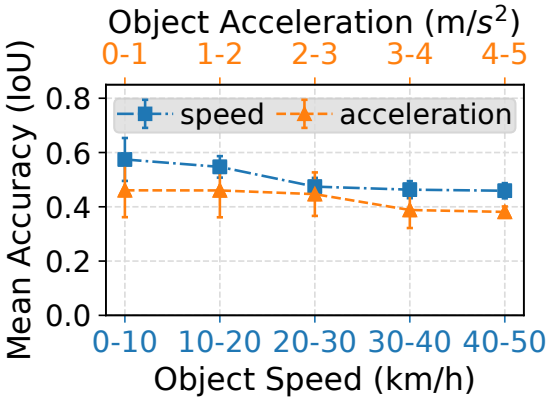


Figure 4.14: Prediction accuracy *w.r.t.* vehicle motion.

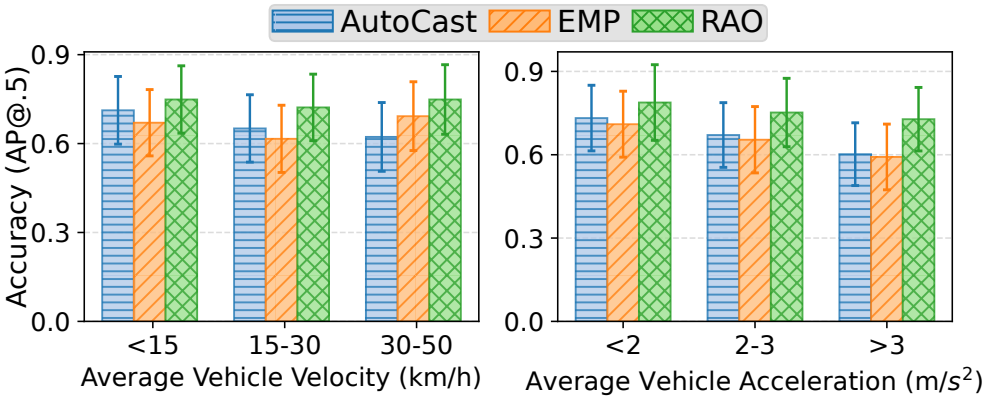


Figure 4.15: Perception performance *w.r.t.* vehicle motion.

marizes the distribution of point density per object ($N_Y^{(i)}/N_X^{(i)}$). Over 90% objects in *RAO* have a density score around 0.2-0.4 and very few objects host a high density of points. The result indicates that *RAO* distributes shared LiDAR points on more objects instead of augmenting a few objects that are already clearly visible. Data scheduling in *RAO* successfully covers most objects in the scenarios in an efficient fashion.

Examples. We next show visualized examples to illustrate how *RAO* tackles the problems of asynchronous sensor data (§4.2.1) and incomplete occlusion estimation (§4.2.2). As shown in Figure 4.9, in the baseline schemes without synchronization, the shared points from other CAVs (blue) will fall out of the ground-truth bounding boxes (green) of the objects, resulting in an inaccurate prediction (red). Instead, the LiDAR points from different CAVs are well aligned in *RAO*, thanks to prediction-based synchronization. Figure 4.10 demonstrates a higher coverage of blind spots in *RAO*. The blue points represent the producer's object points shared with the ego vehicle (consumer). With the consumer's knowledge, fine-grained occluded areas are accurately identified.

4.5.3 Impacting Factors

We analyze the following impacting factors to understand how *RAO* performs in various driving scenarios.

Distances to the ego CAV. Figure 4.11 presents the number of correctly detected objects (IoU¿0.5) based on the object's distance to the ego vehicle. As shown, the most perception benefits come from the mid-range distances (20-40m and 40-80m range), where *RAO* outperforms AutoCast and EMP with 7.04% to 24.03% (0.62 – 1.40) and 3.45% to 18.62% (0.30 – 1.14) more objects detected through collaboration.

Number of CAVs. We can learn from Figure 4.12 that, as the number of CAVs participating in the cooperative perception increases, the perception accuracy gains of *RAO* become larger. This is due to the fact that, with more vehicles collaborating, the impact of synchronization grows higher since more vehicles fall out of sync with each other. For instance, when there are 4-5 CAVs, *RAO* brings 6.95% and 7.63% improvements in perception accuracy over AutoCast and EMP, respectively. This is 2× more than the 2-3 CAV collaboration scenarios. More CAVs also increase system overhead but *RAO* is scalable, as analyzed in §4.5.5.

Traffic density. Next, we consider the overall traffic density. We group the scenarios into three density levels based on the number of all vehicles in each frame. In Figure 4.13, *RAO*'s performance remains the best across all traffic density settings, with 4.20% to 9.66% improvements.

Vehicle motion. CAV motion affects the accuracy of occupancy flow prediction (§4.3.4). We group perception targets by their velocity and acceleration, and use IoU between the predicted occupied regions and the ground truth to evaluate prediction accuracy on each group. As shown in Figure 4.14, prediction accuracy drops gradually as velocity and acceleration increase, but still achieves 0.4 IoU on average in the worst case (0.5 IoU is considered very accurate in perception [2]). Figure 4.15 further illustrates the impact of vehicle motion on end-to-end perception accuracy. Although minor inaccuracy of prediction exists, *RAO* consistently outperforms AutoCast and EMP in all situations by 3.65% to 13.60%.

4.5.4 Ablation Study

We do an ablation study to showcase how the two novel designs, *occupancy flow prediction* and *on-demand data sharing* (§4.3.1), improve the performance of cooperative perception.

Benefits of occupancy flow prediction. We create two variants of *RAO*: (1) one disables prediction algorithms, meaning that providers' LiDAR images are not synchronized with the consumer's data; (2) the other enables ground-truth synchronization, which directly

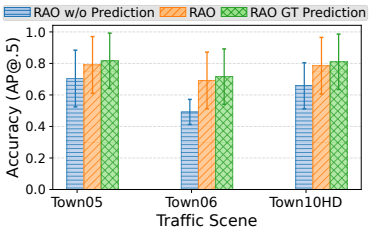


Figure 4.16: Ablation study on occupancy flow prediction.

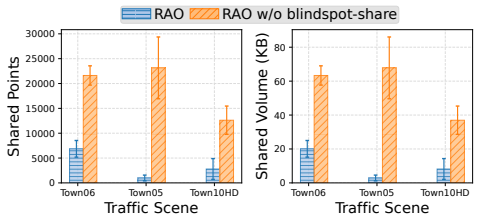


Figure 4.17: Ablation study on the on-demand consumer-centric data scheduling.

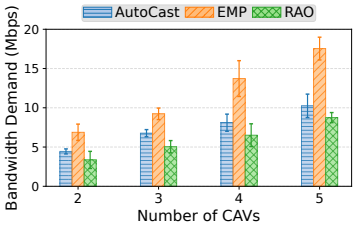


Figure 4.18: Network bandwidth requirements and scalability study.

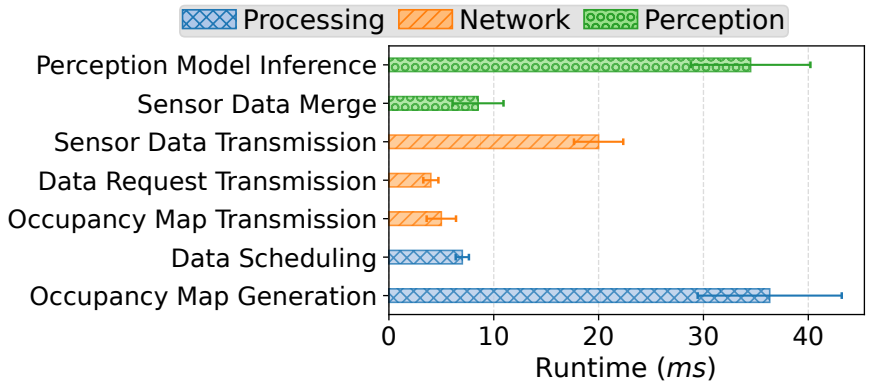


Figure 4.19: Latency breakdown of *RAO*.

moves objects points according to ground-truth labels and perfectly synchronizes all LiDAR images in data fusion. Figure 4.16 demonstrates the difference between our proposed *RAO* and the two altered versions. The average precision achieved by employing ground-truth prediction is only 1.97% higher than *RAO*. This suggests that the synchronization module of *RAO* achieves a performance close to the optimal. When disabling prediction, the accuracy of perception drops significantly by 13%, which also highlights the importance of synchronization.

Benefits of on-demand data scheduling. We disable data scheduling in *RAO* (*i.e.,* producers directly share LiDAR images after ground removal) and compare *RAO* with the downgraded version. As shown in Figure 4.17, in all three traffic scenarios, selectively sharing data in the occlusion region reduces the data volume of sharing by over 68.45% (up to 95.8%). *RAO* can achieve high perception accuracy (Table 4.1) with such a limited amount of data sharing, validating *RAO*'s accurate identification of critical areas for perception.

4.5.5 System Overhead

The latency of *RAO* consists of latencies from data processing, network transmission, and perception, as discussed in §4.2.1. Figure 4.19 provides a breakdown of different latency types.

Table 4.3: Data sharing overhead for each provider-consumer pair per frame (mean±standard deviation).

Metrics	EMP	AutoCast	***RAO***
LiDAR Points	8320±3228	3140±2171	3110±2501
Control data (KB)	<0.1	<0.1	1.77±0.50
Total Volume (KB)	24.37±9.46	9.17±6.36	10.90±7.32

Data processing involves occupancy map generation and data scheduling. The occupancy map generation (clustering, motion estimation, *etc.*) is the most compute-intensive in *RAO*, which takes an average of 36.3 ms. Data scheduling is lightweight and can be finished in 7.0 ms. The network transmission latency depends on the size of shared data. A comparison of data sizes is shown in Table 4.3. For each connected pair of provider and consumer, in each frame, *RAO* shares one LiDAR image containing 3110 points and one occupancy map with 211 polygon vertices on average. Along with tiny metadata such as LiDAR sensor pose and motion information, the total shared data size is around 10.9 KB. The size is similar to that of AutoCast's data (9.17 KB) and is only 44% of EMP's (¿24 KB). Note that we remove background points (§4.3.3) in EMP and AutoCast as well for a fair comparison. The perception latency includes data fusion and model inference. The data fusion module simply concatenates point clouds, which introduces a minor latency of 8.5 ms. The model inference time depends on the specific perception model, and it is 34.5 ms in our implementation.

For each LiDAR image, the sum of computation latency and network transmission latency (*i.e.,* δ_2 in Equation 4.1) is 80.8 ms, indicating that the asynchronization between LiDAR images is around 80 – 180 ms. Thanks to the occupancy flow prediction, the asynchronization issue is effectively addressed. Compared with single-vehicle perception, the only latency that *RAO* introduces is the data fusion latency (8.5 ms) because each consumer starts perception immediately upon the generation of its local LiDAR image, with other data processing and network transmission tasks running simultaneously in the background.

RAO is scalable and works well under a large number of participating CAVs. The overall bandwidth overhead grows linearly with the number of connected CAVs (Figure 4.18). Vehicular communication based on C-V2X can provide a bandwidth of up to 100 Mbps [147, 86], far beyond sufficient for our system. The overhead of occupancy map generation is independent with the number of CAVs as each CAV processes its own LiDAR images locally.

4.5.6 Case study: *RAO*'s Real-world Benefits

In addition to the experiments on large-scale datasets, we conduct another empirical case study using data collected at *Mcity* to demonstrate *RAO*'s superior performance in the real

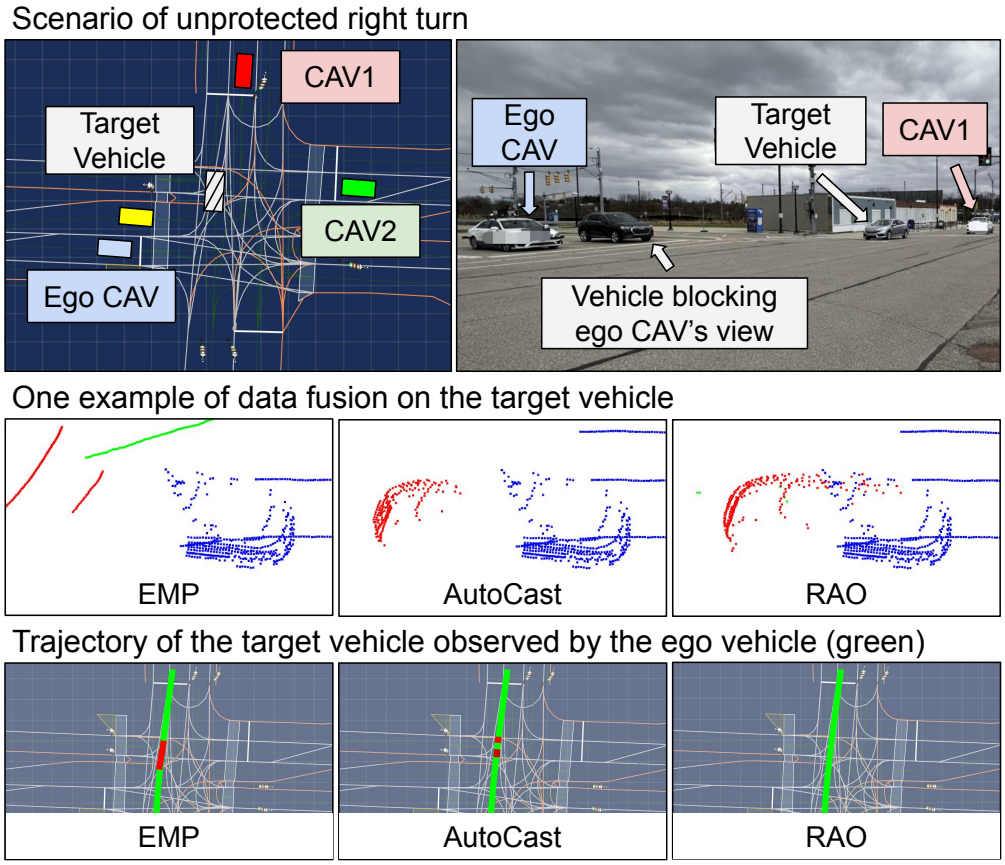


Figure 4.20: Case study of multiple cooperative perception schemes on real-world drive scenes at *Mcity*.

world. As shown in Figure 4.20, the ego CAV is turning right on red and should yield to a vehicle (target) coming from the left. The ego CAV's view toward the target, however, is blocked by obstacles, making it necessary to rely on the other two CAVs to localize the target.

We execute *RAO*, EMP, and AutoCast and find that, (1) EMP is not aware of occlusion and its data scheduling solely depends on distances. Since the LiDAR data of the target area should be shared by the CAV closest to it, at specific locations (red section of the trajectory), other CAVs will not share data for the target even though the target is in the blind spot of the ego vehicle, leading to a failed detection. (2) AutoCast can recognize the blind spot and ask the red CAV to share LiDAR data, but it suffers from a serious synchronization issue (§4.2.1) that makes object detection unstable in certain frames. (3) In contrast, *RAO* addresses both blind spot detection and synchronization issues simultaneously, producing well-aligned augmented point clouds.

Over the 25 examined frames (2.5 s) in which the target is in the blind spot, *RAO* consis-

tently identifies the target, while EMP and AutoCast fail on 12 and 5 frames, respectively. Notably, detected objects in AutoCast have a relatively higher deviation from the ground truth due to the asynchronous data. In this case, *RAO*'s robust perception earns more time for the ego CAV to react to the incoming traffic.

4.6 Summary

In this chapter, we design, implement, and evaluate *RAO*, a real-time occlusion-aware multi-vehicle collaboration system running on asynchronous sensors. With the sophisticated integration of motion-aware occupancy map, occupancy flow prediction, and data scheduling, *RAO* intelligently shares sensor data filling blind spots of other vehicles with minimized network bandwidth consumption and accurately aligns data collected under real-world asynchronous sensor settings through prediction. We believe that our approach to tackling the synchronization issue in sensor sharing can spur a new wave of autonomous driving applications based on multi-vehicle collaboration, and ultimately accelerate their real-world deployment.

CHAPTER 5

SCORPION: Robust Spatial-Temporal Collaborative Perception Model on Lossy Wireless Network

5.1 Introduction

Connected and autonomous vehicles (CAV) rely heavily on their onboard sensors to perceive and navigate in complex driving environments. However, single-vehicle perception systems often struggle with occlusions and limited sensing ranges, potentially leading to severe consequences [229]. Collaborative Perception [101, 143, 227, 210, 198, 237, 209, 61, 154, 235] addresses the limitation of single-agent sensing by allowing multiple agents (vehicles/infrastructures) to share sensor data via vehicular networks so that each vehicle gains an extended sensing range and a better understanding of spatial semantics, particularly in occluded areas. Recent research also highlights that incorporating temporal consistency enhances the perception of dynamically moving objects [227, 216, 98]. Thus, robust and accurate perception relies on comprehending both spatial and temporal information for a complete contextual understanding.

However, it's important to note that collaborative perception relies on networks and hardware devices that are subject to real-world constraints, significantly impacting both temporal and spatial information. Realizing collaborative perception thus requires a deep understanding of these underlying systems. For instance, packet drops in Vehicle-to-Everything (V2X) communication result in lost spatial semantics, leading to missing crucial information needed for decoding the final perception output, while network delays can cause temporal misalignment during data fusion. Similarly, hardware devices such as GPS+RTK [34] and onboard clock devices often produce noisy measurements: localization errors in vehicle pose can lead to spatial misalignment [189], and clock deviation [227] can introduce temporal misalignment. Fig. 5.1 and 5.2 illustrate the impact of these issues, highlighting the need for AI solutions that account for the fundamental limitations of underlying stacks.

Historically, efforts to enhance network reliability [160, 146] and vision application per-

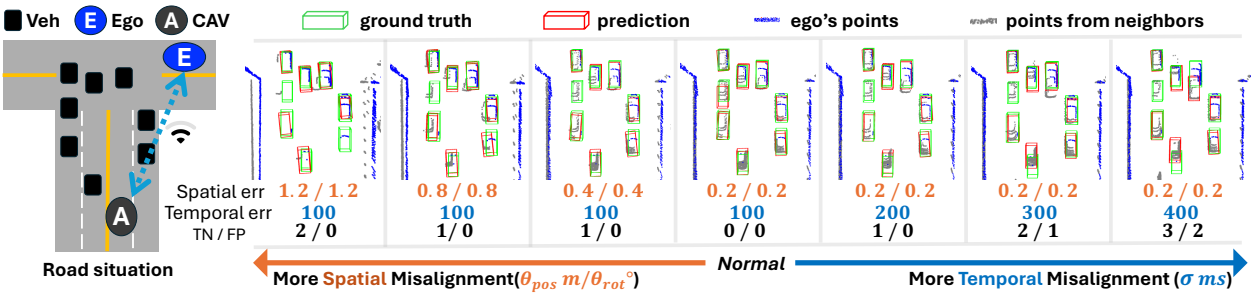


Figure 5.1: Motivating examples of various degrees of spatial and temporal misalignments. Spatial misalignments occur due to sensing errors ($\star$) or dropped network packets, while temporal misalignments arise from sensor asynchronization ($\star$) and network delays. For simplicity and clarity, we focus on the cases marked with ($\star$) in this figure. This scenario is generated using the CoBEVT [208] model on the V2XSet Dataset [209]. TN and FP represent the True Negative and False Positive detections made by the model.

formance [120, 218] were often pursued independently, reflecting a longstanding division between network and application-layer advancements. Previous work on collaborative perception application (Table 5.1) often focus on improving perception performance by designing advanced fusion mechanisms [210, 208] and strategies to reduce network bandwidth usage [101]. All these solutions assume that the communication and perception environments are ideal. A few recent work started to look at real-world noises such as pose noise (localization noise) [209, 143, 216], synchronization noise [227], and lossy communication [126]. However, *none* of these approaches take a holistic view of the entire system stack, including fundamental constraints such as packet encoding in vehicular networks or the interplay between hardware noise and network impairments. This gap significantly limits the practical deployment and scalability of collaborative perception in real-world applications.

To bridge the gap, in this work, we present `SCORPION`, a robust **S**patial-temporal **CO**llabo**R**ative **P**erceptI**o**n model on l**O**ssy **N**etwork. `SCORPION` is an end-to-end approach that embodies the nascent philosophy of AI co-design across the application and lower system layers (including lossy network and noises of hardware devices) to directly address and compensate for the inherent imperfections from these system layers. At the core of `SCORPION` is a bird's-eye view (BEV)-based intermediate fusion: BEV effectively captures the spatial layout with a consistent top-down view, facilitating effective fusion from different viewpoints; meanwhile, intermediate fusion strikes a balance between network resource demands and accuracy, making it a practical approach in collaborative perception [210, 209]. Based on the BEV feature map, our three main components enhance spatial and temporal understanding: (1) For spatial semantics, our lost BEV feature reconstruction module incorporates realistic network packet encoding and recovering strategies that can recover spatial

semantics lost during wireless V2X communication; (2) for temporal consistency, temporal alignment module align synchronization errors by leveraging historical frame information; and (3) for the correlation between temporal features and spatial features, a deformable spatial cross-attention module facilitates flexible mapping between ego vehicle's historical and neighbor's features. By integrating these components into a single transformer [191] architecture, `SCORPION` demonstrates notable improvements over state-of-the-art (SOTA) methods on the OPV2V, V2XSet, DAIR-V2X datasets [210, 209, 219] in both standard and challenging scenarios, achieving robust collaborative perception.

The main contributions of this work are as follows:

- Real-world networking and hardware challenges, like lossy communication and sensor noise, are typically addressed within domain-specific boundaries [227, 228, 143]. `SCORPION` demonstrates the new potentials of transformer-based AI to tackle these persistent issues, expanding its scope to enable co-design across the entire hardware-network-AI software stack.
- `SCORPION` achieves robust collaborative perception by introducing a novel feature masking and reconstruction strategy specifically designed for vehicular networks (lost feature reconstruction), integrated with a deformable spatial cross-attention module and a temporal alignment module, allowing simultaneous tolerance to dynamic noisy environments with lossy V2X communication, localization errors, and synchronization errors.
- `SCORPION` outperforms SOTA methods in both standard and challenging scenarios, including extreme sensing errors and congested network conditions. For example, on the V2XSet dataset [209], `SCORPION` achieves 13.2%, 7.5%, and 5.9% higher accuracy than the SOTA approaches in situations involving lossy networks, spatial misalignment, and temporal misalignment.

5.2 Motivation

Collaborative Perception. Collaborative perception solutions can be divided into three categories based on message-sharing strategy: (1) Early fusion, (2) Intermediate fusion, and (3) Late fusion. Late fusion integrates decisions from independent models at the object level [167] and tends to yield suboptimal accuracy due to the lack of cross-sensor feature enrichment [65]. Early fusion shares raw sensor data, which is network-resource intensive. To reduce bandwidth cost, existing methods [228, 227, 237] use partitioning, where data from each vehicle's nearby region is transmitted and merged to create an extended view of

Table 5.1: Comparison of Different Works Handling system imperfections.

Work	Sensing Errors	Sensor Asynchronization	Lossy V2X Network	Fusion
EMP [228]	✗	✗	✗	Early
Harbor [237]	✗	✗	✗	Early
Where2comm [101]	✗	✗	✗	Intermediate
CoBEVT [208]	✗	✗	✗	Intermediate
RAO [227]	✗	✓	✗	Early
V2X-ViT [209]	✓	✓	✗	Intermediate
CoAlign [143]	✓	✗	✗	Intermediate
LCRN [126]	✗	✗	✓	Intermediate
SCOPE [216]	✓	✓	✗	Intermediate
VIPS [178]	✗	✓	✗	Late
SCORPION (Ours)	✓	✓	✓	Intermediate

the environment.

Intermediate fusion shares encoded features, reducing the network resources needed compared to raw data. For effective feature fusion, AttFuse [210] applies self-attention to analyze features at the same spatial locations across agents. CoBEVT [208] shares Bird's Eye View (BEV) features to efficiently represent multi-view data and uses transformers to fuse BEV features. Where2Comm [101] optimizes bandwidth by identifying and sharing only essential information. Beyond accuracy and bandwidth efficiency, some works propose robust models to improve resilience in error-prone scenarios. V2X-ViT [209] introduces an advanced attention model to handle asynchronous error in V2X communication. LCRN [126] achieves

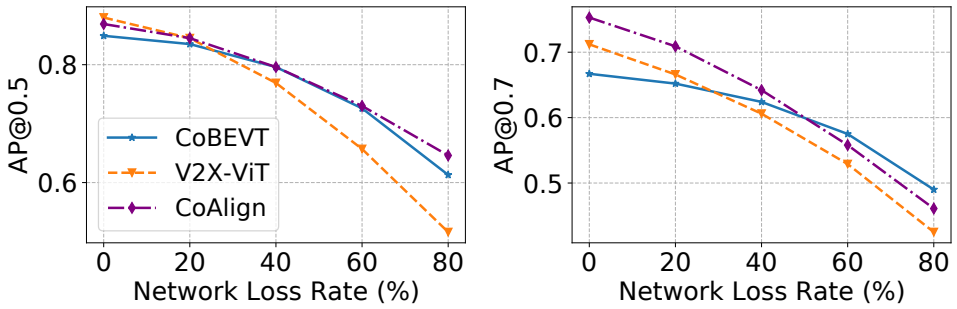


Figure 5.2: Illustration of detection performance drop suffering from lossy V2X wireless communication based on the V2XSet dataset [209]. The three baseline methods, CoBEVT [208], V2X-ViT [209], and CoAlign [143] are trained with no network packet loss and tested at various level of loss.

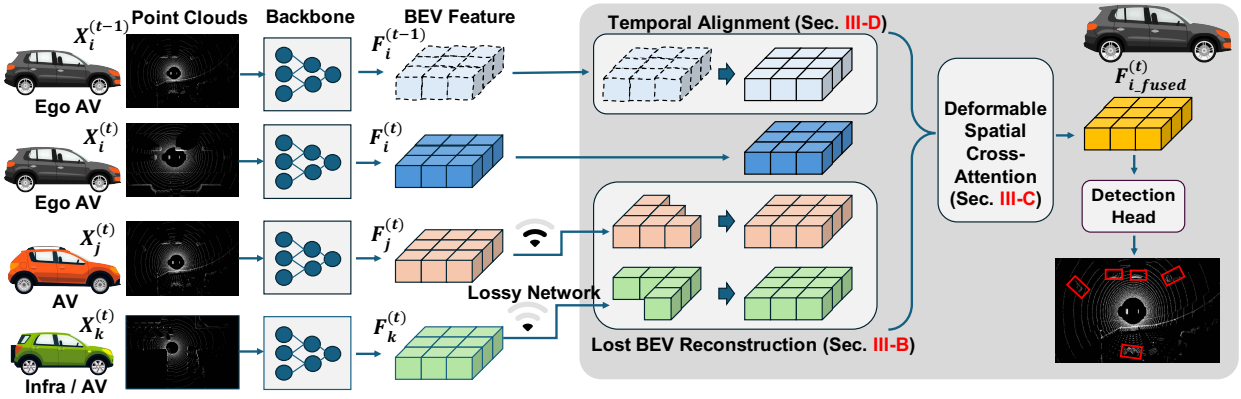


Figure 5.3: The overall architecture of SCORPION. The framework consists of three major components: a lost BEV feature reconstruction (L-BEV-R) module, a deformable spatial cross-attention (DSCA) module, and a temporal alignment (TA) module.

robust vehicle-to-vehicle communication under lossy communication by employing a repair network. CoAlign [143] addresses pose estimation errors through agent-object pose graph modeling to ensure pose consistency across agents. While temporal and spatial misalignment and lossy communication are common issues in reality, no current work addresses all of these challenges comprehensively (see Table 5.1). This motivates us to develop a more robust collaborative perception network for practical scenarios.

Vehicle-to-Everything (V2X) Network. Collaborative perception relies on transmitting sensing information between agents. Technologies like DSRC [111] and C-V2X [193] are designed for V2X communication. However, these wireless networks are inherently unreliable. Communication on vehicular networks often relies on UDP [162] as the underlying transport protocol [51, 52, 5], which can have a high loss rate under dynamic and fast-changing vehicle topologies. Prior work on collaborative perception [209, 210, 101, 208, 143, 227, 154] assumes that the transmission of feature map or raw data will not get lost, which is not realistic on a real-world V2V/V2X setup [5, 52]. The only prior work explicitly addressing lossy communication [126] oversimplifies the issue by simulating network loss with random noise while *ignoring the structured packet encoding of shared feature maps*, making it impractical for real deployment. Therefore, we argue that tolerance to practical lossy V2X communication is a vital characteristic that collaborative perception models must address, as incomplete shared intermediate features from lossy communication can significantly compromise effectiveness (Fig. 5.2).

5.3 Method

In this paper, we focus on enhancing the robustness and accuracy of collaborative perception, emphasizing temporal and spatial understanding in the face of inherent system imperfections. We propose a novel BEV-based intermediate fusion framework `SCORPION`. Our framework consists of three key components (Fig. 5.3): Lost BEV Feature Reconstruction (L-BEV-R), Deformable Spatial Cross-Attention (DSCA), and Temporal Alignment (TA).

5.3.1 Problem Formulation

At time t, N agents participate in a collaborative perception task, establishing a communication graph. We designate the i-th agents as the ego vehicle, with other agents within the communication range acting as collaborators. These collaborators share perceptual information and metadata (*e.g.,* pose, timestamps) with the ego vehicle for collaboration. Each agent $j \in \{1, ..., N\}$ encodes the point clouds into Bird's Eye View (BEV) features to extract local visual features. Given the point cloud $X_j^{(t)}$ of j-th agent at timestamp t, the extracted feature is $F_j^{(t)} = f_{enc}(X_j^{(t)}) \in \mathbb{R}^{C \times H \times W}$, where $f_{enc}(\cdot)$ is the feature encoder shared among all agents, and C, H, W stand for the channel, height, and width. The collaborator agents share the processed feature maps with the ego agent for fusion and perception. In a realistic setup, feature maps go through lossy V2X communication, and poses or timestamps of agents can contain noise. We show how `SCORPION` handles the lost spatial features in §5.3.2, how `SCORPION` addresses the misaligned spatial information in §5.3.3, and how to maintain temporal consistency in §5.3.4.

5.3.2 Lost BEV Feature Reconstruction

Reconstructing the original signal from partial observation of images has been a challenging task in computer vision. In the context of realistic collaborative perception, our goal is to address the critical issue of data loss in V2X communications, a gap not effectively bridged by current methods [209, 208, 143]. Inspired by the masked-autoencoder (MAE) [92], we design a customized Lost-BEV-Reconstruction (L-BEV-R) network for intermediate feature recovery from other CAVs. We treat the intermediate BEV feature map after the lossy communication as partial images, where the locations with features that are received as *visible patches* and treat the locations of the features that get lost during V2X network transmission as *masked patches* that require recovering.

The framework of the L-BEV-R network is shown in Fig. 5.4, which is an encoder-decoder architecture based on Vision Transformers [73]. We next elaborate on the details.

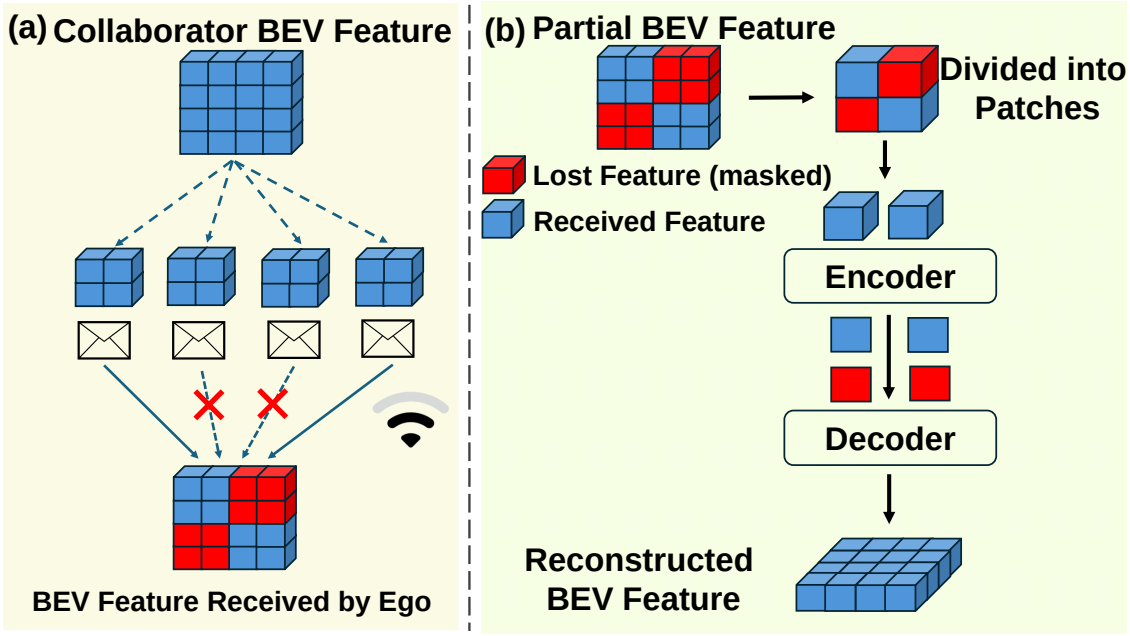


Figure 5.4: SCORPION's Lost BEV Feature Reconstruction architecture.

V2X Network Packet Encoding and Partitioning for Feature Maps. Previous work [118] on the system and network community has proposed methods to encode raw point cloud data into self-contained network packets. To the best of our knowledge, no prior work has discussed how to encode and partition the intermediate feature maps into network packets. In SCORPION, we detail the process of packing the Bird's-Eye-View (BEV) features into multiple *self-contained network packets*. In the context of intermediate-fusion collaborative perception, network packet loss impacts by dropping feature map values. Assuming a BEV feature map of size $\mathbb{R}^{C\times H\times W}$, the actual transmission of the feature will break the feature map into multiple network packets (Fig. 5.4(a)), where the size of each packet is limited by Maximum Segment Size (MSS). For V2X communication, MSS is in the range of 50 bytes to 300 bytes [30, 118]. For example, assuming a feature map of shape $(C', H, W) = (16, 4, 4)$ where C' is the channel size after compression [210] and an MSS of 280 bytes (Fig 5.5), a packet can hold information on $\lfloor 280/((16+1)\times 4)\rfloor = 4$ cells in the BEV map, where "+1" accounts for additional metadata of feature indices.

Fundamental Similarity of Network Packet Loss and Feature Masking. As a network packet can cover a small range of geometric locations in the BEV feature map, we can see that packet losses are fundamentally similar to masking input images in the MAE architecture. In this case, a packet loss during the V2X communication indicates that the ego vehicle will lose the 4 cells in the received feature map. Based on this, our masking strategy in the L-BEV-R module is straightforward: we place patches on the feature maps, treat the patches that contain lost indices as masked patches, and train an encoder-decoder

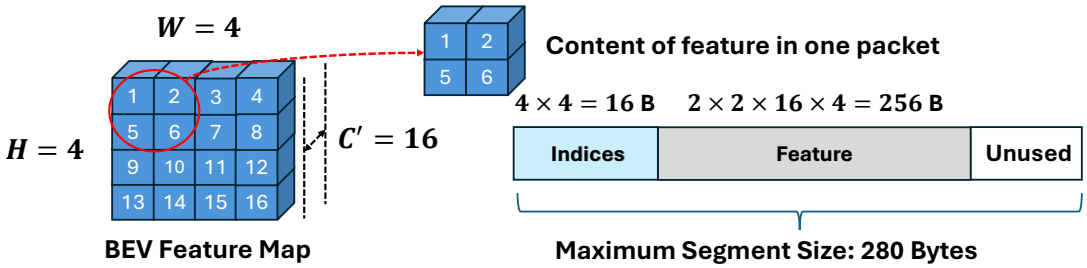


Figure 5.5: An example of an encoded feature map into a network packet.

network to recover them.

L-BEV-R Encoder. Our encoder is a variation of the ViT [73] by extending the input space from 3-channel RGB images to C channel BEV feature maps. Similar to a standard ViT, we divide the feature map into patches of size $P \times P$ and add positional encoding to the patches generated from a linear projection layer. A high portion (*e.g.,* 75%) of random patches will be masked during the training and the encoder operates on the visible patches through a series of Transformer blocks.

L-BEV-R Decoder. The input to the decoder is the full set of patch-tokens consisting of both encoded feature patches and masked patches representing the features that are lost during V2X communication. The decoder has another series of Transformer blocks to reconstruct a feature map of (C, H, W). The reconstruction target is the pixel value of the patch and is linearly projected to the original feature map channel C. During the training of the L-BEV-R model, we use the processed BEV feature maps generated from the backbone network [210] as the ground truth and use the mean squared error (MSE) as the loss function and compute the loss only on masked patches. Later, we fixed the parameters of the trained L-BEV-R model in the training of the entire end-to-end collaborative perception model. During testing and inference, the L-BEV-R module processes the partial BEV feature from collaborators by reconstructing the original feature map before fusing them with the ego feature.

5.3.3 Deformable Spatial Cross-Attention

After reconstructing the partial feature map shared by collaborators, we proceed to fuse the multi-agent feature to enhance the perception capability over single-agent perception. Existing works [209, 210] propose to use attention-based mechanisms to fuse multi-agent feature maps and achieve SOTA performance in ideal environments. However, the assumption that feature interaction only happens at the same spatial location (*i.e.,* the same indices in the feature map in the ego agent's coordinate system) makes these methods vulnerable to localization and synchronization errors, as these errors may create misalignment of percep-

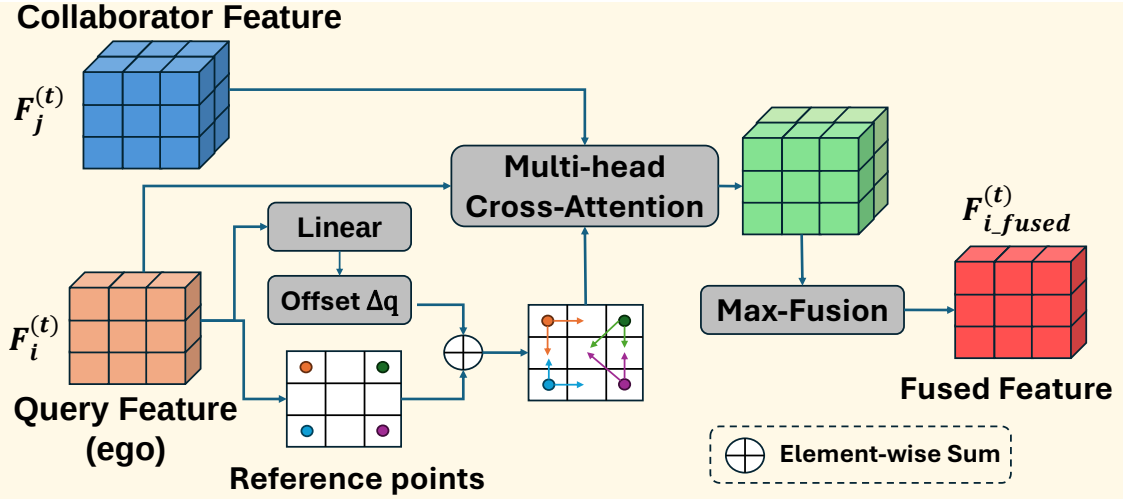


Figure 5.6: The architecture of the proposed deformable spatial cross-attention (DSCA) module.

tual critical areas. To mitigate such effects, we draw inspiration from [239] and implement a simple yet effective Deformable Spatial Cross-Attention (DSCA) component. The DSCA module consists of deformable attention along with a max fusion aggregation of multi-agent features (Fig. 5.6). Different from prior work [216] that employs deformable attention *locally on only high confidence regions*, DSCA in `SCORPION` models the interaction of features *globally from the entire feature map* that contains localization and synchronization error and applies a max-pooling layer to aggregate most salient features, resulting in better perception performance at higher noise levels (see Sec. 5.4.2).

Deformable Spatial Cross-Attention Module. As depicted in Fig. 5.6, we treat the ego feature $F_i^{(t)}$ as the query feature Q and obtain a default reference point sets (same as the naive attention in [209]). To resolve the misalignment of BEV feature maps, the DSCA performs multi-head cross-attention on a set of sampled offset points. The *deformable* concept helps in utilizing and learning the spatially spread-out features caused by aforementioned misalignments. For each reference query point q (*i.e.,* each cell in the BEV feature grid), we select X points based on a learned offset and extract their features as the attending features. A linear layer is used to learn the 2D spatial offset Δq of the reference point q for sampling positional offset at $q + \Delta q$. The output at each query 2D position from ego feature $F_i^{(t)}$ to a collaborator feature $F_j^{(t)}$ is shown in Equation 5.1.

$$DSCA(q)_{i,j} = \sum_{a=1}^{A} W_a \left[\sum_{x=1}^{X} \phi(W_b F_i^{(t)}(q)) F_j^{(t)}(q + \Delta q_x) \right] \tag{5.1}$$

where A is the total number of attention heads and a indexes the attention head, $W_{a/b}$ are learnable weights for each ego-collaborator pair, and $\phi(\cdot)$ is the softmax function.

Max Fusion. For each pair of the ego BEV feature and a collaborator BEV feature, we generate a deformed cross-attention map at each spatial location q. To aggregate the attention features, we further apply a max-fusion at each query point to extract the most salient feature. The final output from the DSCA module at each query position is

$$DSCA(q) = \max(DSCA(q)_{i,1}, DSCA(q)_{i,2}, ..., DSCA(q)_{i,N}) \tag{5.2}$$

5.3.4 Temporal Alignment

Integrating historical observations has the potential to enhance the temporal consistency of collaborative perception. To capitalize on this potential, we treat past BEV feature maps as additional input sources. These historical features are aligned to the current timestamp using the ego vehicle's motion data. This alignment allows us to perform deformable cross-attention alongside the reconstructed feature maps from other agents, effectively leveraging both past and present data to minimize errors and improve the robustness of the perception system.

Historical Feature Alignment. Given the BEV queries Q at current timestamp t and a history BEV feature $F_i^{(t-1)}$ observed at time $t-1$, we first align the historical feature to the current timestamp using ego-agent motion with GPS+IMU pose to make the features at the same BEV grid correspond to the same real-world location. Note that such alignment using ego-motion handles the shifts of static environments but not dynamic objects in the scene. Inspired by [131], we model the temporal correlation between features by using a deformable temporal self-attention (DSA) layer, as follows:

$$DSA(q_p, \{F_i^{(t)}, F_i^{(t-1)}\}) = \sum_{V \in \{F_i^{(t)}, F_i^{(t-1)}\}} \text{DeformAttn}(q_p, V) \tag{5.3}$$

where q_p denotes the BEV query located at $p = (x, y)$ and DeformAttn is the same deformable attention module in the DSCA module. In our implementation, we merge the DSA layer with the DSCA by stacking $F_i^{(t-1)}$ with $F_j^{(t)}$, $j \in \{1, ..., N\}$ as they are conceptually same but with different parameter and meaning (*i.e.,* cross-agent attention *vs.* temporal self-attention). The fused feature map is then obtained from the max fusion of DSCA output $F_{i_fused}^{(t)} \in \mathbb{R}^{C \times H \times W}$ and goes through a series of standard transformer layers including LayerNorm [191] and MLP [84].

5.3.5 Detection Head

We leverage both the regression decoder and classification decoder to decode the final fused feature map $F^{(t)}_{i_fused}$ into the predicted outputs. The regression and classification decoders are MLP layers. The regression output is $\mathbb{R}^{7\times H\times W}$, where the first dimension 7 represents the position (x, y, z), size (h, w, l), and yaw angle θ of the predicted bounding boxes at each location. The classification output is the confidence level of each predefined box to be a target vehicle or background. Following existing work [120], we adopt the smooth L1 loss for regressing the bounding boxes and a focal loss [170] for classification.

5.4 Experiments

5.4.1 Datasets and Evaluation Metrics

Datasets. We validate our proposed method `SCORPION` with LiDAR-based 3D object detection on three datasets: two simulation datasets **V2XSet** [209], **OPV2V** [210] and a real-world vehicle-to-infrastructure dataset **DAIR-V2X** [219].

Implementation Details. We build `SCORPION` based on the Pytorch [161] and OpenCOOD framework [33] and train the models on NVIDIA A100 GPUs [31]. Pointpillar [120] is used as the backbone feature encoder with 0.4m × 0.4m voxel size. We train our L-BEV-R network following the standard MAE network training [92] for 100 epochs and freeze the parameters in the end-to-end model training.

Evaluation Metrics. Following existing work [209, 210], we adopt the Average Precision (AP) at Intersection-over-Union (IoU) thresholds of 0.5 and 0.7 for evaluation.

Benchmark Comparison. We compare with seven state-of-the-art (SOTA) intermediate fusion models: V2VNet [198], F-Cooper [61], AttFuse [210], CoBEVT [208], V2X-ViT [209], CoAlign [143] and SCOPE [216]. In addition, we also compare our approach with single-agent perception (*i.e.,* No Fusion).

5.4.2 Quantitative Results

We compare the performance of 3D object detection with other methods (Table 5.2). Our proposed approach `SCORPION` achieves the highest perception performance compared to the baseline methods on all datasets. It achieves average improvements of 22.6%/25.2%, 23.8%/28.7%, and 11.6%/9.4% on three datasets in AP@0.5/0.7 compared with 'No Fusion', showing the benefits of collaborative perception. Compared to prior SOTA collaborative perception methods, our method improves AP@0.7 by over 2.4% on the V2XSet dataset,

Table 5.2: Performance comparison on the V2XSet, OPV2V, and DAIR-V2X datasets in perfect environments.

Model	V2XSet		OPV2V		DAIR-V2X	
	AP0.5	AP0.7	AP0.5	AP0.7	AP0.5	AP0.7
No Fusion	65.73	52.57	69.38	56.40	63.04	47.39
V2VNet [198]	87.82	74.28	86.76	73.38	65.09	48.18
F-Cooper [61]	82.82	69.38	89.22	79.66	70.54	52.21
AttFuse [210]	81.70	66.24	88.54	72.91	68.02	48.40
CoBEVT [208]	81.00	65.06	88.99	72.80	67.61	55.51
V2X-ViT [209]	82.32	71.21	86.74	75.70	70.87	54.35
CoAlign [143]	86.90	75.31	91.60	82.30	74.02	**56.81**
SCOPE [216]	87.55	75.67	89.60	80.71	74.15	56.52
SCORPION	**88.32**	**77.78**	**93.10**	**85.10**	**74.65**	56.76

Table 5.3: Performance comparison under combined network loss, localization and synchronization noise.

		V2XSet				**OPV2V**				**DAIR-V2X**			
Case	Loss Rate	**SCORPION**	SCOPE	CoAlign	V2X-ViT	**SCORPION**	SCOPE	CoAlign	V2X-ViT	**SCORPION**	SCOPE	CoAlign	V2X-ViT
UF	26%	**84.0/61.2**	78.7/57.3	68.4/39.0	72.9/52.7	**88.6/76.4**	84.6/70.5	74.9/59.0	80.5/63.1	**72.5/54.3**	70.7/52.8	70.3/53.8	68.6/53.3
RF	20%	**84.2/61.9**	78.3/57.5	68.7/38.6	74.3/53.2	**88.9/77.1**	84.8/70.5	76.0/60.0	80.9/63.0	**72.9/54.5**	70.9/53.2	70.1/54.1	69.6/53.7
RR	32%	**82.0/61.0**	77.3/57.0	68.6/38.5	71.3/52.1	**88.4/75.9**	84.5/70.3	73.2/57.6	80.1/62.8	**72.1/54.3**	69.9/53.0	70.5/53.2	68.5/53.3
SR	15%	**85.2/63.1**	78.6/56.5	67.7/38.9	75.5/53.6	**89.3/77.7**	84.9/70.8	77.2/61.0	81.0/63.9	**73.2/55.2**	71.5/54.8	70.1/55.1	69.0/53.5
OF	12%	**84.8/64.9**	78.8/56.6	67.4/39.1	76.0/53.7	**89.6/78.0**	85.1/70.9	78.0/61.5	81.1/63.9	**73.5/55.5**	71.8/54.2	71.1/55.0	69.2/53.5

2.8% on the OPV2V dataset and achieves comparable performance with the best method on DAIR-V2X dataset.

Performance with Network Loss, Localization and Synchronization Noise Together. Using data from V2X network measurement [52], GPS errors [180], we evaluate SCORPION under the *coexistence of packet loss, localization, and synchronization error*. We summarized 5 diverse road settings with a variety of loss rate (12% - 32%): (1) Urban Freeway (UF), (2) Rural Freeway (RF), (3) Rural Road (RR), (4) Suburban Road (SR), (5) Open Field (OF). The network packet loss is simulated by randomly masking out the shared feature maps. We simluate localization error by adding Gaussian noise with a standard deviation of 0.6 m for urban areas and 0.4 m for rural areas and open fields following statistics from [180]. For synchronization latency, we set it to 100ms following existing work [209, 166]. We select the *top 3* performing baseline methods from the perfect environment setup, V2X-ViT, CoAlign and SCOPE for comparison in later experiments. Note that the selected baselines have partial designs on handling the system noises (Table 5.1).

From the results in Table 5.3, we note that SCORPION outperforms existing baselines by over 4%, 2%, and 5% for V2XSet, OPV2V, and DAIR-V2X dataset in absolute AP when

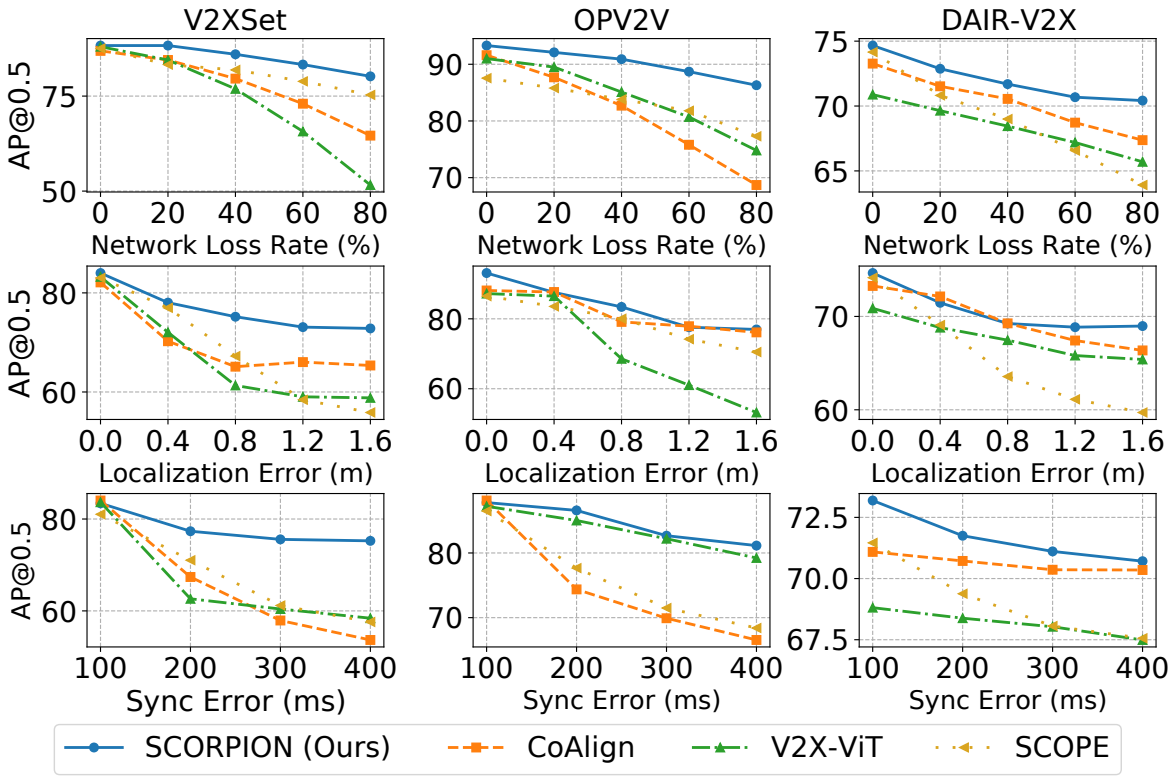


Figure 5.7: Performance comparison across models under various network loss/localization/synchronization errors.

imperfect feature transmission, localization noises, and synchronization errors exist together. These improvements stem from our holistic design, which effectively handles feature loss (via L-BEV-R) and aligns misaligned features caused by localization and synchronization errors (via DSCA and TA).

5.4.3 Robustness to Individual Noise Factor

In this section, we evaluate the robustness of `SCORPION` by analyzing its performance under different levels of individual noise factor: network loss, localization errors, and synchronization errors.

Results under Various Network Loss Rates. Fig. 5.7 reveals that our proposed method `SCORPION` shows superior performance under all levels of network loss rates. An increase in the network loss rates leads to a rapid and sharp decrease in the perception performance for baseline methods, as perceptual information gets lost in the shared feature maps. For instance, with 40% network loss, SOTA methods like V2X-ViT and CoAlign see reductions in AP@0.5 of more than 20% for V2XSet and OPV2V, and over 5% in DAIR-V2X, compared to ideal communication conditions. Our method consistently outperforms previous SOTA models in all levels of network losses, clearly demonstrating the robustness of `SCORPION` to

network loss. Note that for higher network loss, our method maintains higher improvements in accuracy thanks to the L-BEV-R module. When the network loss rate gets as high as 80%, our method outperforms the second-best approach by 5.1%, 7.5% and 3.1% on V2XSet, OPV2V, and DAIR-V2X dataset.

Results under Various Localization Errors. To demonstrate the robustness under localization error, we vary the localization noise from 0 to 1.6 m to the agents' poses in the datasets (2× higher than pervious work [216, 209, 143]). As depicted in Fig. 5.7, our robust perception design surpasses state-of-the-art methods that are pose-error aware, enhancing average perception performance by 6.8%, 1.3%, and 1.0% across the datasets, respectively. Comparing to SCOPE [216], a SOTA baseline that also tackles spatial and temporal misalignments, `SCORPION` shows 5.5% to 16.9% performance under higher noise conditions because of the DSCA models global feature interaction more precisely. Unlike CoAlign [143], which relies on prior training of an uncertainty estimator for bounding boxes, our end-to-end model with the DSCA approach offers a simple yet effective solution for addressing localization errors, thanks to the deformable attention interaction with offset locations in the misaligned feature maps.

Results under Various Synchronization Errors. The third row of Fig. 5.7 illustrates the perception performance under different levels of synchronization errors. Compared to methods like V2X-ViT which have designs to handle asynchronous time delay, our approach can still improve the perception accuracy by 1.1% to 11.6% across datasets. `SCORPION` consistently outperform the baseline approaches from 200 ms to 400 ms of asynchronous data. Compared to methods (*e.g.,* CoBEVT, AttFuse) that do not take asynchrony into account, our method improves AP@0.5 by an average of 11.6%, 11.1%, and 7.4%. The performance boost is due to two key factors: (1) TA module aligns historical feature information to the current ego timestamp, mitigating the impact of asynchronous feature interaction from other agents and allowing feature interaction to focus on perceptually important areas from ego space; (2) DSCA module facilitates perceptually critical information interaction across agents under both spatial-temporal domains, making the perception results more robust.

5.4.4 Ablation Study

Contribution of Major Components. To evaluate the effectiveness of our proposed component, we progressively remove the three key elements in our framework. Table 5.4 details the contribution of each component. The consistent drop in detection accuracy demonstrates the effectiveness of each component: L-BEV-R, DSCA, and TA. By adding DSCA, AP@0.7 is improved by 2% compared to using naive attention mechanism [209]. By further incorporating past ego-agent features and performing temporal alignment, we

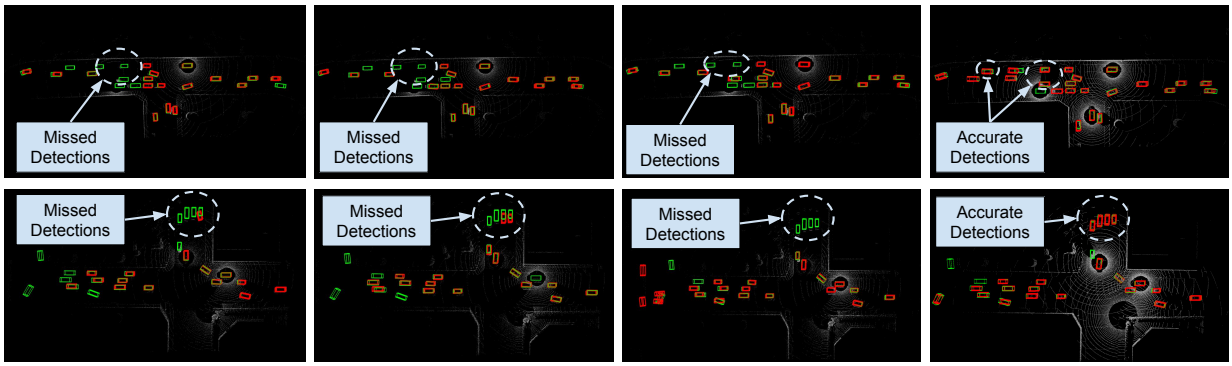


Figure 5.8: Qualitative comparison results on the V2XSet dataset under **localization error, synchronization error, and network loss altogether**. The green and red boxes are ground truth and prediction outputs, respectively.

Table 5.4: Feature comparison and performance metrics on V2XSet. For baseline w/o DSCA, we use attention fusion from [209].

DSCA	TA	L-BEV-R	Lossy Network	AP0.5 / AP0.7
✓	✓		N	**88.32**	**77.78**
✓			N	86.59	75.48
			N	86.09	73.52
✓	✓	✓	Y	**86.01**	**70.83**
✓	✓		Y	83.82	69.83
✓			Y	81.79	69.51
			Y	79.80	64.94

further improve AP@0.7 by another 2.5%, indicating the effectiveness of both the DSCA and TA modules. Furthermore, incorporating the L-BEV-R module enhances `SCORPION`'s robustness under lossy communication conditions [52], improving AP@0.5 by 2.2% compared to baselines that do not include feature reconstruction capabilities.

Impact of Offset Point Count in DSCA. We adjust the deformable offset points from 2 to 16 in DSCA (Tabel 5.5). Noticeably, DSCA with 4 points achieves better detection performance on all three datasets. This result shows that `SCORPION` effectively extracts essential spatial semantics by setting a reasonable offset point number.

Effect of Historical Ego Frame Count. As shown in Table 5.5, we evaluate the impact of incorporating different numbers of historical information from the ego agent. The finding is that using 1 historical frame is already beneficial in achieving good performance gains over all datasets.

Visualization of Detection Results. To illustrate the perception capability of different models, we provide a detection visualization comparison (Fig. 6.12) under 1.3 m of

Table 5.5: Ablation study results of SCORPION (AP@0.5/0.7).

Dataset	OPV2V	V2XSet	DAIR-V2X
Effect of offset point numbers in DSCA			
2	89.25/78.82	91.25/77.30	71.94/53.42
4 (Default)	88.32/77.78	93.10/85.10	74.65/56.76
8	87.68/75.29	92.33/81.12	70.91/50.00
16	88.30/75.61	91.78/82.26	70.33/51.29
Effect of historical ego frame counts in TA			
1 (Default)	88.32/77.78	93.10/85.10	74.65/56.76
2	87.98/77.98	92.76/84.54	73.24/54.72
3	88.30/77.43	92.33/84.27	71.96/54.36

Table 5.6: Table on inference speed of SCORPION on V2XSet [209] dataset. The latency is measured on an A100 GPU and shown with mean and standard deviation in ms. The definition of lossy network is the same as Table 3 in the main text.

DSCA	TA	L-BEV-R	Lossy Network	Latency (ms)
✓	✓	✓	Y	92.45 ± 6.13
✓	✓		N	78.72 ± 3.83
✓			N	64.28 ± 3.54
			N	58.72 ± 3.64

localization error, 100 ms of asynchrony latency, and 35% of network loss based on the existing measurement work on GPS [180] and V2X network loss [52]. We also provide a video demonstration of SCORPION at https://www.youtube.com/watch?v=6zCAlhH7pGw. Baseline methods like SCOPE, V2X-ViT, and CoAlign fail to accurately detect moving objects because of the presence of three types of errors. Our proposed SCORPION handles the network packet loss through feature reconstruction and exploits the spatial and temporal misalignments through the DSCA and TA module, thus improving the detection accuracy.

Visualization of feature reconstruction from L-BEV-R. We visualize the reconstructed BEV feature map from the L-BEV-R component and compare it with the original feature map generated at the collaborator agent in Figure 5.9 to Figure 5.11. As shown, our lost BEV reconstruction module effectively recovers missing spatial feature information from lossy V2X communication, making it close to the original BEV feature generated at the collaborator side. Therefore, with better spatial semantic information, collaborative perception empowered by SCORPION achieves better and more robust perception performance.

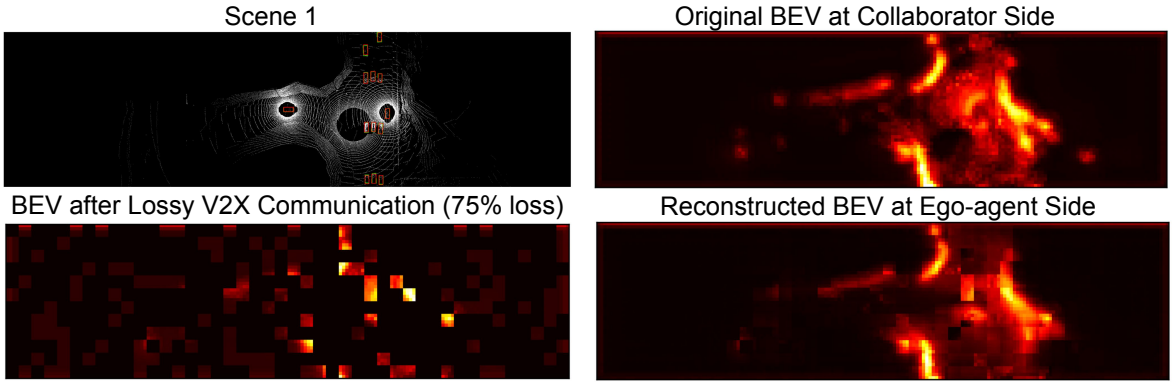


Figure 5.9: Visualization of BEV feature reconstructed at the ego-agent side on an entrance ramp.

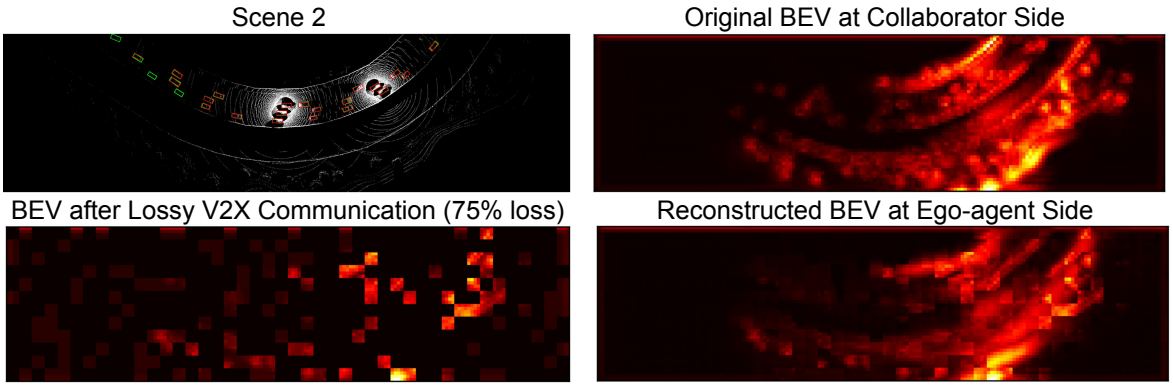


Figure 5.10: Visualization of BEV feature reconstructed at the ego-agent side on a curvy road.

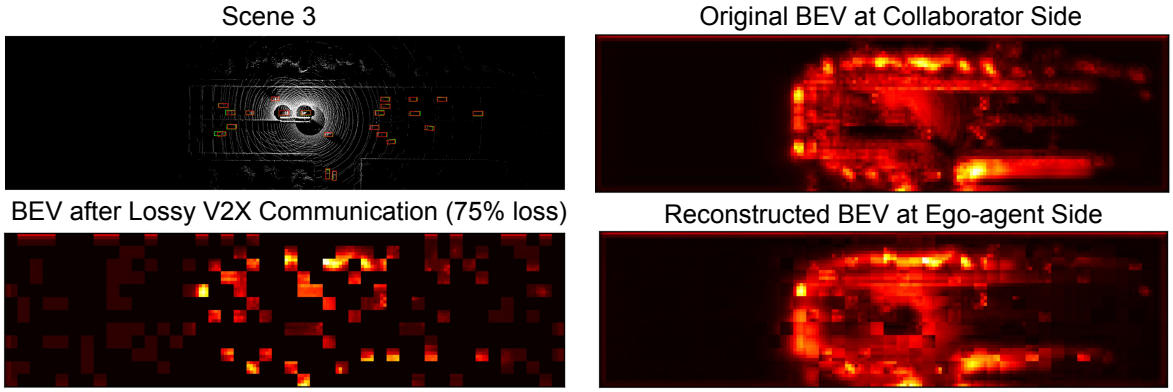


Figure 5.11: Visualization of BEV feature reconstructed at the ego-agent side on a T-intersection.

Table 5.7: Table on GPU memory usage during inference for SCORPION on V2XSet [209] dataset. The average memory usage is shown in MB.

DSCA	TA	L-BEV-R	Lossy Network	Memory (MB)
✓	✓	✓	Y	4680
✓	✓		N	2797
✓			N	2338
			N	2250

5.4.5 System Overhead Analysis

Inference time analysis. To investigate the real-time performance of SCORPION, we measure the inference time of SCORPION on a NVIDIA A100 GPU [31]. As shown in Table 5.6, the mean inference time for SCORPION is 78.7 ms per frame, satisfying the 100 ms end-to-end latency for autonomous driving [135] under *ideal V2X communication* scenarios. Furthermore, even if enabling the L-BEV-R module with a *lossy network environment*, SCORPION achieves an inference latency of 92.5 ms, and still supports real-time inference. Progesstively removing each design component can further save the inference time, supporting higher FPS for real-time operations by sacrificing some detection accuracy as depicted in Table 3 from the main paper.

GPU memory usage. In addition to inference latency, we profile the GPU memory usage of SCORPION with various configurations. As shown in Table 5.7, our model SCORPION with 3 modules activated takes only around 4.5 GB in GPU memory, making it lightweight enough to fit in most GPU devices.

5.5 Summary

This chapter presents SCORPION, which expands AI's role in co-designing the hardware-network-AI software stack by addressing persistent issues like lossy communication and sensor noise in networking and hardware. SCORPION demonstrates improved accuracy and robustness across extensive experiments, including normal and challenging scenarios that expose typical system issues. Future work will extend SCORPION to more complex perception scenarios and incorporate additional sensor modalities, such as monocular and stereo cameras.

CHAPTER 6

GraphMap: Scalable Crowd-Sourced Global Vectorized HD Map Construction via Sparse Visual Graph Fusion

6.1 Introduction

High-definition (HD) maps [41, 38, 70] play a critical role in autonomous driving by providing rich geometric and semantic information, such as lane dividers, boundaries, and pedestrian crossings. Among various formats [37, 36, 208, 97, 58, 127, 141, 107], vectorized HD maps [37, 36], which represent map elements as structured primitives such as polylines and polygons, are becoming the new technical trend of mapping solutions [145, 134, 133] for connected and autonomous vehicles (CAVs). Conventional HD map construction pipelines [173, 174] often rely on collecting dense LiDAR point clouds and require significant human annotation, limiting scalability and update frequency. Recent advances in online HD mapping [133, 134, 221] use deep learning models to generate local HD maps around the ego-vehicle in real time from onboard sensors, significantly reducing the need for offline human intervention. However, individual vehicles suffer from limited sensing range [221], occlusions [208], and incomplete coverage [101], making it difficult to build reliable global maps independently. This motivates a shift toward ***crowdsourced global vectorized*** HD map construction, where a fleet of vehicles (agents) contributes local observations [70] to collaboratively build a unified global map [176, 139, 182, 106, 60].

Previous efforts in online HD mapping and collaborative perception have primarily relied on **dense feature fusion**, where sensor data (*e.g.,* Cameras and LiDARs) are encoded and projected into a shared bird's-eye-view (BEV) representation [209, 61, 227, 143, 208]. These methods aim to align perspective differences among agents and sensors by aggregating features on fixed-size BEV tensors. However, dense fusion methods suffer from several key limitations: First, they are inherently ego-centric, bounded by local fields of view and fixed perception ranges (*e.g.,* 60×30 m), making them poorly suited for constructing global maps; Second, they are sensitive to spatial and temporal misalignment [143, 227], struggling to

robustly fuse spatially misaligned and partially overlapped observations from different agents; Third, increasing the size of dense BEV tensor can incur significant computational [132] and communication overhead [210] due to the high-dimensional, redundant nature of dense feature maps; Fourth, detecting map updates using dense fusion methods [202] requires significant effort in model architecture change and training, making it less scalable. These limitations highlight the need for a more *scalable*, *efficient*, and *robust* approach to global vectorized HD map construction, particularly in large-scale and dynamically changing road environments. To meet these goals, a new approach must: (1) incorporate multi-agent observations and handle spatial-temporal misalignment over extended space and time in a sparse manner; (2) operate efficiently under limited network and computation resources; and (3) provide a lightweight solution to maintain and update the global map upon road reconfiguration.

To address the key challenges of scalable global HD map construction, we propose `GraphMap`, a novel framework that shifts from dense, ego-centric fusion to a graph-based sparse fusion paradigm for collaborative, vectorized global HD map construction. `GraphMap` operates in a two-tier interactive architecture consisting of vehicle-side sensing and cloud-side aggregation and scheduling, with the following key features:

Scalable multi-agent sparse fusion across space and time. To enable scalable fusion across spatially misaligned, partially overlapping observations, `GraphMap` adopts a graph-based representation of local HD maps. Each agent creates a compact map graph from onboard sensor data using online HD map generators. These graphs consist of semantic and geometric attributes (e.g., type, shape, orientation) of map elements and are lightweight for transmission. On the cloud, we introduce the MapGlue algorithm, which incrementally aligns and merges local maps through Type-Aware Structure-Preserving Graph Matching. By modeling intra-region graph structure and solving the NP-hard maximum subgraph matching problem [78] with an efficient greedy approximation, MapGlue robustly handles spatial variation and partial overlaps while preserving topological consistency in map alignment. For map growth beyond local regions, MapGlue performs inter-region graph registration, expanding coverage with new elements and merging consistent observations across space and time.

Communication and computation-efficient map contribution. Rather than uploading every local observation, agents and cloud in `GraphMap` jointly make data contribution decisions via a selective map sensing policy. This policy evaluates the sensing value (SV) of individual local maps by considering three key factors: (1) the trade-off between exploring new areas and reinforcing known regions of the global map, (2) the redundancy of spatially or temporally proximate observations from the same agent, and (3) the differential contri-

bution of heterogeneous agents such as fixed infrastructure and moving vehicles. Solved with an ϵ-greedy Multi-Armed Bandit (MAB) algorithm, this vehicle-cloud feedback loop optimizes that only informative and non-redundant data are transmitted, reducing network and computation overhead while maintaining global map quality.

Lightweight and continual map update detection. To support evolving environments and long-term deployments, `GraphMap` incorporates an HD-Continuous Change Detection and Classification (HD-CCDC) module. By probabilistically validating incoming agent graphs against the stale global map states, HD-CCDC detects changes like road reconfigurations, including element additions and removals, and updates the global map accordingly. The probabilistic approach accumulates evidence across multiple asynchronous observations rather than relying on single detection, making it robust to transient occlusions, sensor noise, and occasional detection errors from individual agents. This enables `GraphMap` to serve not just as a fusion engine, but also as a live HD map infrastructure, adaptable to real-world dynamics and suitable for future applications such as real-time cloud-assisted mapping [95].

Combining all the merits above, the resulting system is an end-to-end global map construction framework to build large vectorized maps through crowd-sourcing with multiple agents while maintaining precision and resource efficiency.

In summary, the main contributions of this work are:

• We propose `GraphMap`, a scalable, graph-based framework for crowdsourced global HD map construction. It achieves robust global stitching and consolidation of map data from heterogeneous agents by demonstrating tolerance to spatial and temporal misalignments. Our intelligent collaborative map construction method, based on sparse graph fusion and selective sensing, enhances accuracy, optimizes wireless network bandwidth usage and computation overhead, improves robustness in dynamic environments, and ultimately enabling a more scalable solution for real-world deployment.

• We introduce the MapGlue algorithm, the core of `GraphMap`'s fusion and maintenance capabilities. It features a novel Type-Aware Structure-Preserving Graph Matching for efficiently fusing sparse map graphs. MapGlue not only robustly matches and merges map elements but also underpins a fully integrated pipeline for dynamic map upkeep. This allows `GraphMap` to build global maps from scratch and continuously detect changes or updates (with HD-CCDC), ensuring an up-to-date map representation.

• We introduce a Sensing Value (SV)-Guided Selective Map Sensing mechanism to address real-world bandwidth and compute constraints for global map construction. This strategy evaluates the sensing value of individual local maps based on their potential to improve the global map while maintaining map consistency. By prioritizing high-value data under limited resources, it enables scalable, efficient, and accurate global mapping.

• Our extensive evaluations demonstrate the scalability, effectiveness, and resilience of `GraphMap` across real-world multi-agent datasets [219, 233] and large-scale simulation town-level scenarios [210] with 30-60 agents. Compared to state-of-the-art methods, `GraphMap` achieves 37.2 mAP improvement in mapping accuracy while reducing computational overhead by >3× and bandwidth usage by 32%. Furthermore, our change detection module successfully identifies map updates with an F1-score improvement of 52.9%.

6.2 Background and Motivation

Online local HD map construction methods. Existing local HD map models, such as HDMapNet [127], VectorMapNet [141], and MapTR [133], are limited by small perception ranges (e.g., 60x30m), with even extensions like StreamMapNet [221] (100x50m) or longer-range efforts like ScalableMap [220] remaining focused on the ego vehicle's immediate vicinity rather than true global coverage. Despite these advancements in local HD map construction, these methods still focus on a relatively small area compared to the requirements for constructing a global HD map. Building an end-to-end global HD map construction framework faces two main challenges: (1) Single-frame or short-sequence data from one vehicle is insufficient for building large regions due to the limited view and occlusions; (2) Scaling transformer-based BEV to global regions is computationally infeasible due to the quadratic complexity. Consequently, global HD map construction necessitates architectural shifts and multi-agent collaboration. `GraphMap` addresses this by enabling collaborative, large-scale HD map creation through sparse graph fusion across diverse road agents and time, thereby overcoming the limitations of ego-centric local detectors.

Limitation of existing global HD map construction methods. Prior work [172, 215, 176, 59] on global HD map construction primarily employs two fusion strategies: (1) *element-fusion:* Methods like PolyMerge [172] and VMA [63] merge vectorized map instances (*e.g.,* polylines) from a single agent with multiple local frames using rule-based heuristics like proximity thresholds. Rule-based coarse-grained merging, however, overlooks finer features across agents or time, often degrading map quality. (2) *temporal intermediate-fusion:* Recent methods [215, 221, 59] incorporate temporal priors to aggregate historical map information from a single vehicle. While accumulating sequential data over time enables global map construction, this approach is restricted to single-agent settings, failing to leverage crowdsourced multi-vehicle data due to significant spatial misalignments [227] from differing viewpoints. In contrast, `GraphMap` employs fine-grained, type-aware fusion across both time and agents, facilitating robust global mapping despite real-world sensing and communication noise.

Prior map-informed map updates detection. Accurate and updated HD maps are vital

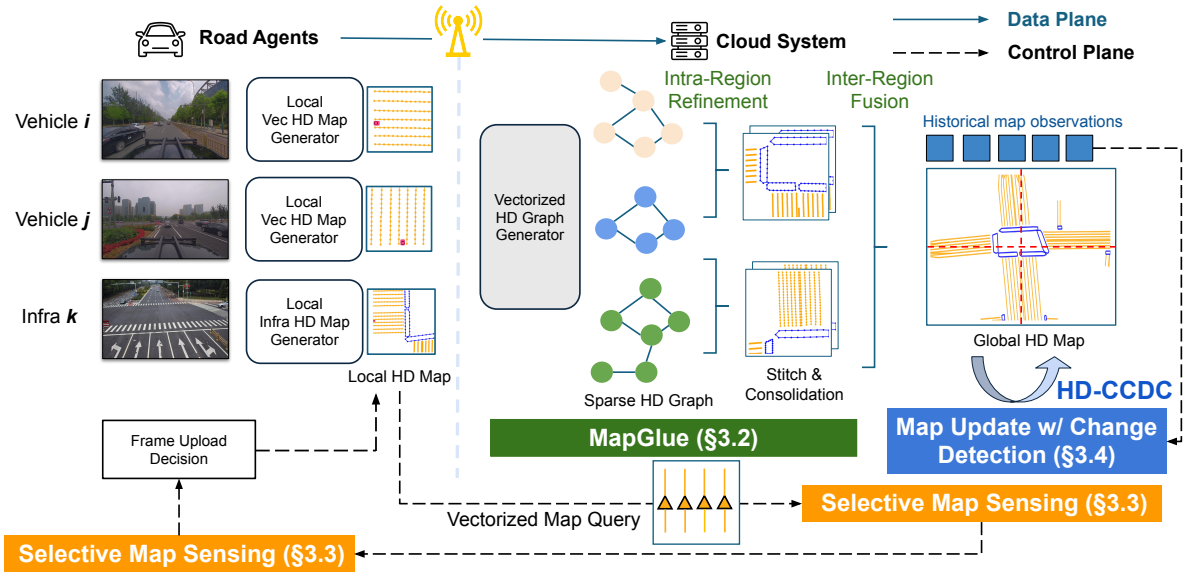


Figure 6.1: System Architecture of `GraphMap`. Local HD map graphs generated by road agents are selectively uploaded to the cloud based on their sensing value. The cloud incrementally fuses these graphs with MapGlue, keeps track of historical map observations, and performs map updates via HD-CCDC.

for safe autonomous navigation, requiring efficient update and change detection mechanisms. Much research leverages existing (or outdated) HD maps as priors to guide this process rather than creating maps anew or detecting changes in isolation. For instance, ExelMap [202] and MapEX [183] integrate current BEV features with historical map data to detect per-lane changes. LDMapNet-U [206] provides a city-scale system that treats lane-level updating as a generation process grounded in historical map data and current BEV images, proven in industrial-scale frequent updates. TbV [119] uses filtering methods to evaluate discrepancies between onboard sensor detections and the prior map, identifying changes in dynamic areas like construction zones. Such methods highlight the existing map's value as a strong baseline for efficiently identifying and localizing real-world changes. Learning from these ideas, `GraphMap` integrates a change detection module. This module cross-validates incoming multi-agent graphs with the current global map in a probabilistic manner to identify updates, enabling both the creation and maintenance of up-to-date maps.

6.3 Design of `GraphMap`

6.3.1 System Architecture

`GraphMap` (Fig. 6.1) is structured around two tightly integrated operational planes: the data plane and the control plane. The data plane governs the transmission of vectorized local

HD maps and sparsely fuses them into a global vectorized map. In parallel, the control plane orchestrates agent-level frame uploading decisions and global map maintenance based on real-time feedback.

Data Plane Operations. The data plane manages the generation, transmission, and fusion of vectorized HD map data across agents and the cloud. Each road agent, *e.g.,* a vehicle (Vehicle i, j) or an intelligent infrastructure unit (Infra k), perceives its environment using onboard sensors such as cameras or LiDARs. These sensor streams are processed by an online local HD map model (*e.g.,* MapTR [133]) to generate vectorized HD map segments of the immediate surroundings. Road agents then selectively upload these compact local map observations to the cloud, along with their current GPS location and IMU pose. Upon receipt, the cloud-side **MapGlue** module (§6.3.2) transforms the local map observations into global coordinates and encodes them into a sparse HD graph. It then performs graph-based sparse fusion of these graphs in two stages: Intra-region Refinement, which consolidates overlapping or closely spaced maps of the same region, and Inter-region Fusion, which stitches distinct geographical map segments into a global map to expand the map's coverage and ensure overall consistency. This results in a progressively expanding, vectorized Global HD Map $\mathcal{G}$ that reflects aggregated observations across space and time.

Control Plane Operations. The control plane of `GraphMap` manages scheduling, adaptation of data uploading from agents, and long-term map maintenance. From the road agent side, the **Selective Map Sensing module (§6.3.3)** determines whether to upload a local map observation via uploading a compact Vectorized Graph Map Query to the cloud. This decision is made using a sensing value policy, which is jointly decided based on the global map feedback and road agents' local characteristics and resources. Instead of continuous data streaming, selective map sensing aims to identify and prioritize the most valuable information for the global map, thereby optimizing network bandwidth and onboard computational resources. At the cloud level, the **HD-CCDC module (§6.3.4)** performs continuous change detection and map updating. By comparing a list of new agent observations against historical global map states, it identifies road reconfigurations, adds or removes elements, and ensures that the global map remains temporally consistent. These updates also help refine future sensing priorities and can be beneficial for future vehicle-cloud applications like real-time cloud-assisted mapping [95].

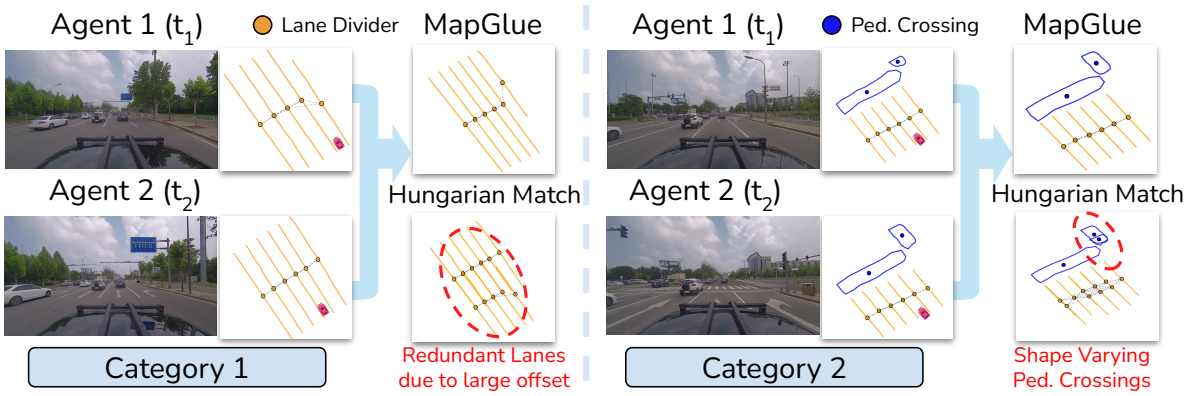


Figure 6.2: Importance of Type-Aware Structure-Preserving Graph Matching: Lane dividers (Category 1) often exhibit larger positional offsets, whereas pedestrian crossings (Category 2) typically appear in consistent locations but vary in shape. Circles in the local observation represent the visual graph node encoding of HD map elements. Hungarian matching [116], a standard distance-based matching algorithm without type and structure-awareness, fails on both cases.

6.3.2 MapGlue: Type-Aware Structure-Preserving Graph Matching

At the heart of `GraphMap` cloud-side fusion pipeline is MapGlue, a Type-Aware Structure-Preserving Graph Matching algorithm designed for robust and scalable vectorized HD map fusion. MapGlue consolidates sparse vectorized map observations from heterogeneous agents, incrementally merging overlapping segments across time and space—even when observations are noisy, spatially misaligned, or temporally disordered.

Previous works on collaborative perception [143, 178, 156] have applied graph-based alignment for multi-agent object detection by matching co-visible objects (e.g., vehicles, pedestrians) across agents. These methods rely on a strong *locality assumption*: the same object will appear at similar positions with similar sizes in different agents' views, making proximity-based graph matching effective. However, this assumption does not hold in the context of global vectorized HD mapping. Based on our analysis, we find two key observations regarding the characteristics of local map elements perceived by diverse agents. The design of MapGlue is informed by key observations regarding two distinct categories of behaviors exhibited by map elements when perceived by agents across varied spatial and temporal contexts.

Category ❶: *Shape-Preserving with Spatial Offsets*

Many map elements, when partially observed by different agents at different times, tend to preserve their intrinsic geometric structure (e.g., **lane dividers**, and **road boundaries**). However, these elements can exhibit significant spatial offsets in their reported absolute positions due to partial occlusions, limited sensing range, or localization drifts.

Category ❷: *Location-Preserving with Shape Variations*

Conversely, other types of map elements typically maintain a relatively consistent location or centroid across multiple observations. However, their perceived geometric structure—such as shape, extent, or completeness—can differ substantially. These variations often arise from differing sensor view angles, the granularity of the detection algorithm, or partial views of the element. **Pedestrian crossings** and **junctions** exemplify this behavior.

These contrasting behaviors, visually exemplified for lane dividers and pedestrian crossings in Fig. 6.2, highlight a critical challenge. Consequently, traditional graph matching approaches that rely on uniform proximity thresholds or fixed structural constraints for all element types are insufficient for reliable and accurate multi-agent HD map fusion. This necessitates the type-aware approach adopted by MapGlue, which adapts its matching and fusion strategies based on the behavioral category of the map elements involved.

Based on these observations, we design MapGlue, a matching strategy that is explicitly type-aware, adapting its criteria based on the semantic class of the map elements being fused. The core idea of MapGlue is to (1) allow Category 1 elements (*e.g.,* lane dividers) to be matched with location shifts but preserve their structure; (2) limit the location shifts of Category 2 elements (*e.g.,* pedestrian crossings) and change their geometric shapes.

MapGlue begins by encoding local HD map observations as sparse semantic graphs. Each graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ comprises a set of nodes $\mathcal{V}$ and edges $\mathcal{E}$ derived from vectorized map elements detected by an individual agent or a state of the map from the cloud. Each node $v \in \mathcal{V}$ represents a map element (e.g., a segment of a lane divider) and stores attributes including its center, orientation, detailed geometry (polyline/polygon coordinates), and semantic label (type). Edges $e \in \mathcal{E}$ capture geometric relationships between nodes. These may include spatial proximity or structural relationships, commonly quantified using the Chamfer distance [127].

The Type-Aware Structure-Preserving Graph Matching algorithm for map fusion then

proceeds as follows. Upon receiving a new map observation, the cloud constructs two graphs: $\mathcal{G}^L = (\mathcal{V}^L, \mathcal{E}^L)$: the local graph derived from the agent's recent observation and $\mathcal{G}^G = (\mathcal{V}^G, \mathcal{E}^G)$: the reference **intra-region graph** extracted from the cloud's current global map state, constrained to a region near the local observation. Let $n = |\mathcal{V}^L|$ and $m = |\mathcal{V}^G|$ denote the number of nodes in the local and reference graphs, respectively. MapGlue computes a type-aware affinity matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ to estimate the potential correspondence between elements in the two graphs. The total affinity $\mathbf{M}$ is composed of two components: Node affinity matrix $\mathbf{M}_{\text{node}}$, which measures candidate matches between nodes; Edge affinity matrix $\mathbf{M}_{\text{edge}}$, which compares structural consistency between node pairs using their edge relationships.

Specifically, MapGlue computes node affinity $\mathbf{M}_{\text{node}}$ and edge affinity $\mathbf{M}_{\text{edge}}$ for each pair of nodes across the local and reference graphs, conditioned on their semantic labels. Let each node $v \in \mathcal{V}$ be represented by a feature vector $f = [x, y, \theta, \ell]$, where: (x, y) is the 2D centroid of the map element, θ is the heading (orientation angle), ℓ is the element length. Let f_i^L and f_j^G denote the features of node i in the local graph $\mathcal{G}^L$ and node j in the global graph $\mathcal{G}^G$ and t_i^L and t_j^G be their semantic labels. The **node affinity** between f_i^L and f_j^G is computed only if the labels match: $t_i^L = t_j^G$ (otherwise set to $-\infty$). For matching labels, the affinity score incorporates the following: Euclidean center distance $d_{ij} = |(x_i, y_i) - (x_j, y_j)|_2$, Heading difference $\Delta\theta_{ij} = |\arctan 2\,(\sin(\theta_i - \theta_j), \cos(\theta_i - \theta_j))|$ and Length difference $\Delta\ell_{ij} = |\ell_i - \ell_j|$. Then, the node affinity score is $\mathbf{M}_{\text{node}}[i, j] = -(d_{ij} + \Delta\theta_{ij} + \Delta\ell_{ij}) + c$, where c is a bias constant. **Edge affinity** favors consistency in local structure by comparing the average intra-class neighbor distances between nodes. Let D^L and D^G denote the pairwise distance matrices for the local and global graphs, respectively, the edge affinity is defined as $\mathbf{M}_{\text{edge}}[i, j] = -\lambda \cdot |\bar{d}_i^L - \bar{d}_j^G|$, where λ is a weighting hyperparameter and $\bar{d}_i^L$ and $\bar{d}_j^G$ are average distance of node i (j) to its close neighbor nodes (within threshold distance τ) in the graph. The full type-aware affinity matrix is obtained by aggregating the node and edge components: $\mathbf{M}[i, j] = \mathbf{M}_{\text{node}}[i, j] \oplus \mathbf{M}_{\text{edge}}[i, j]$ (element-wise sum). This formulation ensures that MapGlue matches elements only when they share semantics and exhibit compatible structure and local geometry.

Upon computing the affinity matrix $\mathbf{M}$, the next step is to identify the largest consistent subgraph between the local graph $\mathcal{G}^L$ and the global graph $\mathcal{G}^G$. This problem, known as Maximum Common Subgraph (MCS) detection, is NP-hard [78], necessitating algorithms that balance precision and computational efficiency. To balance accuracy and efficiency, we leverage the semantic and structural observations introduced earlier (CATEGORY 1 and 2) and design a greedy, type-aware matching algorithm. The core idea is to iteratively select the node pair with the highest affinity while enforcing type-specific constraints on

displacement and geometry alignment. For lane dividers and boundaries, the matching allows for global displacement of lane divider nodes (CATEGORY 1) during graph merging using a displacement variance constraint and a heading angle constraint. For pedestrian crossings (CATEGORY 2), nodes are matched with stronger locality constraints. However, their geometric representations are handled differently during merging; instead of strict shape and orientation matching, the algorithm allows the actual map element (polygon) to grow or reshape by taking the union of the matched observed polygons. Pedestrian crossing nodes in close proximity can also be merged. This matching process proceeds as follows:

1. Find the highest-scoring unmatched node pair (i, j) in $\mathbf{M}$.
2. Check if (i, j) satisfies displacement and heading constraints based on its semantic label t.
3. If valid, record the match, update the displacement buffer, and remove row i and column j from $\mathbf{M}$.
4. Repeat until no valid matches remain. Add remaining nodes to the merged graph $\mathcal{G}^{G'}$.

This greedy approach enables efficient and robust graph matching under real-world conditions, handling misalignments and view inconsistencies while preserving semantic structure. It allows MapGlue to incrementally merge incoming graphs with asynchronous, multi-agent observation.

After obtaining the intra-region merged graph $\mathcal{G}^{G'}$, MapGlue performs an **inter-region alignment** step to integrate the new data into the global map state $\mathcal{G}$. Inspired by ADoPT [66], which demonstrates the utility of point-set registration for capturing temporal consistency, we apply *rigid Coherent Point Drift (CPD)* [150] to align $\mathcal{G}^{G'}$ with the global map. CPD is particularly well-suited for our application due to its robustness in aligning structured 2D primitives like polylines and polygons under small misalignments, as we have already merged the graphs with larger displacements. Unlike prior approaches [172], which often rely on simpler heuristics like proximity for *instance-level* merging, MapGlue employs a sophisticated type-aware, structure-preserving greedy matching algorithm. This method explicitly accounts for the varying displacement consistency observed across different semantic types of map elements. When this intelligent graph matching is synergistically combined with precise *point* (the minimum unit in vectorized HD map) rigid registration [103, 214], `GraphMap` can resolve complex misalignments and viewpoint discrepancies inherent in multi-agent observations with significantly higher accuracy and semantic fidelity, leading to a more consistent and reliable collaborative global HD map construction.

Algorithm 3: Selective Map Sensing with Feedback-Delayed Double-Ended Bandit

Input : Epsilon ϵ, feedback delay d, total frames T, Hyperparamater DataType
Initialize: $Q_{\text{upload}}, Q_{\text{skip}} \leftarrow 0$ `// SV rewards`
$N_{\text{upload}}, N_{\text{skip}} \leftarrow 0$ `// Action counts`
Global raster mask $\mathcal{M}_G \leftarrow \emptyset$
for $t \leftarrow 1$ **to** T **do**
  Extract local map graph $\mathcal{G}_t^L$
  `/* Agent-side: upload decision */`
  if *Random()* $< \epsilon$ **then**
    $a_t \leftarrow$ random choice from {upload, skip} `// Explore`
  else
    $a_t \leftarrow \arg\max(Q_{\text{upload}}, Q_{\text{skip}})$ `// Exploit`
  if $a_t =$ *upload* **then**
    `/* Cloud-side: compute SV */`
    Rasterize $\mathcal{G}_t^L$ to get mask $\mathcal{M}_t$
    `/* Calculate fraction of` $\mathcal{M}_t$ `not in/overlaps with` $\mathcal{M}_G$ `*/`
    $\text{NewArea}, \text{Consistency} \leftarrow \frac{|\mathcal{M}_t \setminus \mathcal{M}_G|}{|\mathcal{M}_t|}, \frac{|\mathcal{M}_t \cap \mathcal{M}_G|}{|\mathcal{M}_t|}$
    $SV_t \leftarrow \alpha\text{NewArea} + \beta\text{Consistency}$; $\mathcal{M}_G \leftarrow \mathcal{M}_G \cup \mathcal{M}_t$
    Schedule SV_t as feedback r_t
  `/* Agent-side: receive feedback */`
  if *feedback* r_t *is available* **then**
    $a_{t-1} \leftarrow$ action taken at time $t-1$; $N_{a_{t-1}} \leftarrow N_{a_{t-1}} + 1$
    $\text{Freshness} \leftarrow \exp(\Delta t) + \exp(\Delta d)$ where Δt is the time since the agent's last upload, Δd is the spatial displacement from the last upload location
    $Q_{a_{t-1}} \leftarrow Q_{a_{t-1}} + \frac{1}{N_{a_{t-1}}}(r_{t-1} + \gamma \cdot \text{Freshness} + \delta \cdot \text{DataType} - Q_{a_{t-1}})$

Algorithm 4: High-Definition Continuous Change Detection and Classification

Input : Global map $\mathcal{G}$, local observations $\{o_t^e\}$, decay constant τ, threshold θ_{add} and θ_{remove}, initial change prior ε (learned from historical calibrated data)
Initialize: $P_0(z^e = \texttt{unchanged}) \leftarrow 1 - \varepsilon$
$report_{\text{added}} \leftarrow \emptyset$
$report_{\text{removed}} \leftarrow \emptyset$
foreach o_t^e *in temporal order* **do**
  `/* Compute observation likelihoods */`
  $P(o_t^e|\texttt{changed})$, $P(o_t^e|\texttt{unchanged})$ based on IoU
  `/* Predict prior from previous step */`
  $P(z_t^e|o_{<t}^e) = \sum_{z_{t-1}^e} P(z_t^e|z_{t-1}^e) P(z_{t-1}^e|o_{<t}^e)$
  `/* Update posterior via Bayesian belief update (Eq. 6.2) */`
  $P(z_t^e = \texttt{changed}|o_{\leq t}^e) \propto P(o_t^e|z_t^e)\, P(z_t^e|o_{<t}^e)$
  `/* Apply temporal weighting */`
  $\omega_t = \exp(-\Delta t/\tau)$
  $P_t(\text{change}) \leftarrow \frac{\omega_t P(z_t^e=\texttt{changed}|o_{\leq t}^e)}{\omega_t P(z_t^e=\texttt{changed}|o_{\leq t}^e) + (1-\omega_t)[1-P(z_t^e=\texttt{changed}|o_{\leq t}^e)]}$
foreach *unmatched local element* l_i **do**
  if $P_t(change) > \theta_{add}$ *and* $l_i \notin \mathcal{G}$ **then**
    $report_{\text{added}}.insert(l_i)$
foreach *unmatched global element* g_j **do**
  if $P_t(change) > \theta_{remove}$ *and* $g_j \in \mathcal{G}$ **then**
    $report_{\text{removed}}.insert(g_j)$
return *Change report: added, removed*

6.3.3 Selective Map Sensing

To efficiently manage network bandwidth and cloud computation in large-scale collaborative HD map construction, `GraphMap` incorporates a Sensing Value (SV)-guided *Selective Map Sensing* (SMS) mechanism. This strategy determines whether an agent should upload its local observation to the cloud. It optimizes data transmission by ensuring only information with significant utility for global map construction is processed, thereby reducing wireless network bandwidth requirements and computation load. The design of our SV function is motivated by three key observations: (1) Global map construction must balance exploration (new area coverage) and exploitation (consistency improvement); (2) Redundant uploads—especially from stationary or slow-moving agents—should be penalized; (3) Agents differ in their mapping value. Static infrastructure sensors initially provide high coverage and consistency. However, their marginal utility for mapping unchanged areas tends to diminish. Vehicles, on the other hand, are crucial for continually contributing to dynamic updates and

capturing new information.

Based on these insights, we define the Sensing Value of an observation $\mathcal{G}^L$ as:

$$\begin{aligned} SV(\mathcal{G}^L) = \alpha \cdot \mathit{NewArea}(\mathcal{G}^L) + \beta \cdot \mathit{Consistency}(\mathcal{G}^L) \\ + \gamma \cdot \mathit{Freshness}(\mathcal{G}^L) + \delta \cdot \mathit{DataType}(\mathcal{G}^L) \end{aligned} \tag{6.1}$$

Each term reflects a distinct utility dimension, and the scalar weights $\alpha + \beta + \gamma + \delta = 1$ allow tuning based on system priorities:

• **NewArea**(G^L) quantifies the degree to which x covers previously unmapped regions, encouraging spatial exploration.

• **Consistency**(G^L) measures how well G^L aligns with the current global map—rewarding observations that maintain map coherence while penalizing those that are perfectly consistent and thus unlikely to contribute new information.

• **Freshness**(G^L) penalizes redundant uploads from slow-moving or static agents, favoring non-repetitive updates.

• **DataType**(G^L) assigns source-dependent weights as hyperparameters (e.g., fixed infra vs. dynamic vehicles), capturing the heterogeneity and time-varying value of agents.

While consistency and freshness are both linked to temporal characteristics, they operate at different layers: Consistency is evaluated at the cloud side by comparing against the current global map, whereas Freshness is computed locally by the agent using movement and timestamps. This separation is intentional—it forms the core of our *double-ended* SV computation design. Importantly, there are scenarios where a "freshness penalty" on the agent side can be overruled by valuable content assessed by the cloud. For example, an agent might have just uploaded a frame, but the next observation may still cover a new area or reveal updated geometry previously occluded (e.g., due to dynamic objects like trucks). In such cases, the combined cloud-side feedback—capturing both new coverage and improved consistency—can outweigh the freshness penalty, leading to a justified upload. This dynamic trade-off is handled seamlessly through our hybrid SV computation.

To schedule uploads across time and agents, `GraphMap` formulates the SMS policy using a *feedback-delayed, double-ended* ϵ-greedy multi-armed bandit (MAB) algorithm [185]. The algorithm is a lightweight Reinforcement Learning (RL) method [80, 115, 50]. While it is efficient to run in real-time, it still captures the core trade-off between exploration (trying potentially valuable uploads) and exploitation (leveraging known high-SV behaviors), which is the key to our case. Each agent (or potential data stream) is treated as an arm, and its reward corresponds to the SV of its observation. With probability $1 - \epsilon$, each agent selects to follow the prediction of reward SV of past feedbacks (exploitation); with probability ϵ, it explores by randomly making decisions to account for uncertainty or unobserved changes

and to prevent the system from getting stuck in local optima.

The *double-ended nature* ensures that SV computation is distributed. Agents can locally compute parts of the SV: DataType (a hyperparameter based on its static/mobile role), Freshness (based on GPS/IMU movement, time since last upload). The cloud, on the other hand, must compute the global-context components: NewArea (by comparing the observation to the global map's current coverage), and Consistency (by rasterizing and aligning the observation with existing fused segments). Detailed formulas are in Algorithm 3.

Moreover, the *delayed feedback* setting reflects the practical reality that the full impact of an upload, *i.e.,* NewArea and Consistency measure, can only be assessed after MapGlue has processed the data. This makes our designed MAB model particularly suitable for asynchronous, multi-agent scenarios. The workflow of the algorithm is presented in Algorithm 3. While MAB is a classical tool for balancing exploration and exploitation, our unique contribution lies in extending it to a *feedback-delayed, double-ended design* tailored for *decentralized, multi-agent HD map construction.* By partitioning reward generation between agent-side and cloud-side in a way that enables asynchronous, resource-aware upload decisions, SMS enables `GraphMap` to build a comprehensive vectorized global HD map while minimizing communication and compute costs through intelligent, adaptive data selection.

6.3.4 Detecting Map Updates with HD-CCDC

Real-world road environments are highly dynamic, with frequent structural or semantic changes caused by construction, maintenance, or temporary events [119, 44, 53]. To ensure the long-term accuracy and reliability of the global HD map, `GraphMap` incorporates a robust mechanism for cross-validating HD Map Changes through a probabilistic method named *High-Definition Continuous Change Detection and Classification* (HD-CCDC). Inspired by the original CCDC paradigm [240] for detecting landscape-level changes on satellite images, our method improves the majority-voting-based CCDC method on vectorized map elements and enables uncertainty-aware map maintenance across multiple asynchronous agents.

The core idea of HD-CCDC is to treat the state of each map element as a latent state variable whose probability evolves over time. Rather than tallying discrete votes from agent observations, each observation contributes probabilistically to the posterior belief that a map element has changed. Specifically, for each map element e_i, we maintain a posterior $P_t(\text{change}|O_{1:t})$ that reflects the confidence of change after t rounds of observations $o^e_{\leq t} = \{o^e_1, o^e_2, \ldots, o^e_t\}$.

Each map element e_i is associated with a latent binary state variable $z^e_t \in \{\texttt{changed}, \texttt{unchanged}\}$ and an observation sequence $o^e_{\leq t} = \{o^e_1, o^e_2, \ldots, o^e_t\}$. At each new

observation, the posterior belief of being changed is updated recursively following:

$$P(z_t^e \mid o_{\leq t}^e) \propto P(o_t^e \mid z_t^e) \sum_{z_{t-1}^e} P(z_t^e \mid z_{t-1}^e) P(z_{t-1}^e \mid o_{<t}^e) \tag{6.2}$$

Here, $P(o_t^e \mid z_t^e)$ represents the observation likelihood — how probable the observed local map evidence is, given that the element is in a changed or unchanged state — while $P(z_t^e \mid z_{t-1}^e)$ captures the temporal transition dynamics. To incorporate temporal relevance, we apply a time-decay weight $\omega_t = \exp(-\Delta t/\tau)$ when integrating evidence. This ensures that recent, high-confidence observations have a stronger influence, while outdated information gradually loses impact. $P(z_t|z_{t-1})$ and prior observation $P(o_t|z_t)$ are learned through Baum-Welch algorithm [213] using IoU geometric similarity [6] from calibrated data (more details in evaluation). After updating all elements, the posterior probabilities are used to classify changes as follows: (1) elements with high posterior but missing from the global map are labeled *added*; (2) existing elements with persistently high posterior but no recent matches are labeled *removed*; These probabilistic change candidates are then integrated through MapGlue (§6.3.2) for incremental updates to $\mathcal{G}$. The pseudo-code of HD-CCDC is shown in Algorithm 4.

By leveraging probabilistic inference and time-decaying weights in change detection, HD-CCDC unifies temporal consistency, observation reliability, and uncertainty propagation into a single module. It mitigates false positives from transient mismatches, adapts to asynchronous sensing, and provides a continuous probabilistic interpretation of map evolution. This enables `GraphMap` to maintain and refine global HD maps with both statistical rigor and practical robustness in dynamic real-world environments.

6.4 System Implementation

We implement `GraphMap` using both Python and C++, and the prototype consists of about 8K lines of code. We implement an emulator of crowd-sourced multi-agent global vectorized map construction. The emulator accepts pre-recorded sensor traces of different modalities, including camera images or LiDAR point clouds and 3D poses of vehicles over consecutive frames. Emulated vehicles run MapTRv2 [134] as their online local HD map detector, and upload the vectorized detections via SMS policy. The emulated cloud runs MapGlue, cloud-side SMS in the foreground, and HD-CCDC in the background. We run all our emulation experiments on an Ubuntu 20.04 server machine with a 64-core CPU and 128 GB of memory.

Network Emulation. To emulate real vehicle-cloud communication, we use real-world cellular and V2X [153, 149] network traces collected during driving and emulate changing

Table 6.1: Performance comparison on global HD map construction on DAIR-V2X. `GraphMap` (C) stands for agents using only camera data, and (C+L) for using both LiDAR and camera for local HD map detection.

DAIR-V2X Maps: 06, 08, 09

Map	Method	AP_{ped}	AP_{div}	mAP
Range: 300×300 m				
06	PolyMerge	14.8	8.5	11.7
	FDC	20.2	2.7	11.4
	HBM	16.7	9.4	13.0
	`GraphMap` (C)	**87.5**	30.3	58.9
	`GraphMap` (C+L)	**87.5**	**32.0**	**59.8**
Range: 250×250 m				
08	PolyMerge	8.0	13.7	7.4
	FDC	33.0	0.2	16.6
	HBM	0.0	22.9	11.5
	`GraphMap` (C)	**100.0**	**50.2**	**75.1**
	`GraphMap` (C+L)	**100.0**	48.1	74.1
Range: 300×300 m				
09	PolyMerge	76.9	5.8	41.3
	FDC	59.4	4.6	32.1
	HBM	7.0	5.0	6.0
	`GraphMap` (C)	**92.3**	7.7	50.0
	`GraphMap` (C+L)	**92.3**	**16.6**	**54.5**

DAIR-V2X Maps: 10, 13, 16

Map	Method	AP_{ped}	AP_{div}	mAP
Range: 300×300 m				
10	PolyMerge	0.1	18.3	9.2
	FDC	31.0	0.3	15.7
	HBM	0.0	11.9	6.0
	`GraphMap` (C)	**85.7**	41.6	63.6
	`GraphMap` (C+L)	**85.7**	**45.7**	**64.7**
Range: 250×250 m				
13	PolyMerge	61.8	8.1	39.9
	FDC	51.6	1.8	26.4
	HBM	57.5	6.3	31.9
	`GraphMap` (C)	**100.0**	8.8	54.4
	`GraphMap` (C+L)	**100.0**	**16.3**	**58.1**
Range: 250×250 m				
16	PolyMerge	49.4	1.7	25.6
	FDC	45.0	1.2	23.1
	HBM	12.5	9.8	11.1
	`GraphMap` (C)	**83.3**	9.9	46.6
	`GraphMap` (C+L)	**83.3**	**14.0**	**52.8**

bandwidth while simultaneously replaying the sensor data streams. The bandwidth statistics of the traces are 13.5 ± 10.4 Mbps, with 16.6 ± 5.4 ms of latency and $4.5 \pm 12.7\%$ of packet loss. In our evaluation, we emulate agents to use UDP for uploading the vectorized local maps. This is because instead of TCP, UDP is the primary underlying transport layer protocol for vehicular applications [52, 149].

Training details. We train a state-of-the-art online local HD model, MapTRv2 [134] with 4 NVIDIA A100 GPUs with a batch size of 8. We adopt ResNet50 [93] as the image backbone and PointPillar [120] as the LiDAR backbone. Other training details are consistent with [133].

6.5 Evaluation of `GraphMap`

6.5.1 Experiment Setup

Motivation for using multi-agent datasets. Unlike prior work that evaluates local HD map models on single-vehicle datasets such as nuScenes [56] or Argoverse [203], our goal is to evaluate the end-to-end performance of collaborative, vectorized global HD map construction across asynchronous and spatially distributed data. Multi-vehicle datasets are essential to capture key properties such as agent heterogeneity, spatial overlap, and asynchronous sensing, which are not presented in single-vehicle benchmarks. Therefore, we use both real-world

and synthetic multi-agent datasets to faithfully emulate the crowdsourced global map construction scenario. While multi-vehicle datasets are extensively studied for object detection tasks [219, 233, 143, 227], to the best of our knowledge, our work is the first work to evaluate **global vectorized maps** using real-world multi-vehicle data.

Datasets and evaluation metrics. We evaluate `GraphMap` on two real-world multi-agent datasets: **DAIR-V2X** [219] and **V2XPnP** [233], and one synthetic multi-vehicle dataset: **OPV2V** [210], using standard HD map detection and evaluation settings [133, 134, 127]. The DAIR-V2X dataset includes one vehicle and one infrastructure agent at six intersections. Each agent is equipped with a front-view RGB camera and the dataset provides global 2D vectorized HD map annotations. Due to the front-only camera view, we restrict the local perception range to the forward x-axis of each agent. For the vehicle, the perception range is $[0, 30]$ meters in the x-direction and $[-15, 15]$ meters laterally; for the infrastructure, the forward range is extended to $[0, 75]$ meters. The V2XPnP dataset involves four agents: two connected vehicles and two infrastructure nodes. Each vehicle is equipped with four RGB cameras, allowing us to extend their local perception range to $[-30, 30]$ meters in the x-direction. Each infrastructure node has two cameras and shares the same perception configuration as in DAIR-V2X. We adopt the mean Average Precision (mAP) metric [133], evaluated over multiple chamfer distance thresholds (1.0, 1.5, 2.0 m), to assess vectorized global map accuracy. Following prior work [133, 134], we evaluate three common types of vectorized map elements: pedestrian crossings, lane dividers, and road boundaries. Due to different available annotations in the dataset maps, we evaluate pedestrian crossings and lane dividers on DAIR-V2X, and lane dividers and road boundaries on V2XPnP. To further assess `GraphMap` under large-scale, highly dynamic multi-agent settings, we conduct additional experiments on the OPV2V dataset [210]. Unlike DAIR-V2X and V2XPnP, which are real-world datasets collected from limited spatial extents, OPV2V provides simulated but photo-realistic cooperative driving scenes across multiple **town-level maps** in CARLA [26], featuring diverse urban layouts, curved lanes, intersections, and heterogeneous traffic participants. We group scenarios in the dataset based on the map number in the original train/validation set for global map construction. Each grouped scene involves 30–60 agents equipped with RGB cameras and LiDAR sensors. Each agent extracts their local HD map predictions within a fixed perception range of $[-30, 30]$ meters in both axes. We evaluate the fused global maps using the same mAP metric as in the real-world experiments, focusing on pedestrian crossings and lane dividers.

Collaborative map update benchmark. To evaluate the system's ability to detect and integrate map changes, we build **DAIR-V2X-Change**, a synthetic benchmark based on DAIR-V2X. In this benchmark, the ground truth map is manually altered [119] while

the original front-view camera image sequence remains unchanged, emulating real-world map update scenarios. We simulate these updates by removing/adding a subset of HD map elements from the original ground truth map. These removals/additions represent reconfigurations like construction or road repaintings. Following standard practice [210], we perform train/validation/test splits, using the training set to calibrate HD-CCDC parameters via Baum-Welch, and validation/test sets to evaluate change detection performance.

Baselines. We compare `GraphMap` against several baselines, which pair the vehicle-side MapTRv2 [134] model with different state-of-the-art (SOTA) cloud-side fusion strategies.

• **PolyMerge** [172]: A SOTA rule-based mechanism merging vector polylines via spatial proximity and geometric heuristics, reimplemented from the paper due to lack of open-sourced code.

• **Fréchet Distance Clustering (FDC)** [54]: It is a strong representative of trajectory similarity-based clustering approaches for grouping and consolidating similar map elements based on their intrinsic shapes, often considered a more sophisticated alternative to simple proximity. These metrics are frequently benchmarked in trajectory analysis [89, 64].

• **Heatmap Buffer Merge (HBM)**: Utilizes an intermediate raster representation, converting vectors to heatmaps, buffering for merging [59], then re-vectorizing [74]. Raster-based fusion followed by vectorization is a technique explored in various map works [59, 176].

We train distinct MapTRv2 detectors for vehicle agents and infrastructure agents (using their specific calibrations and visibility) to handle heterogeneity. To evaluate the impact of sensor modality, we create two variants of `GraphMap`: `GraphMap` (C) using camera-only local HD map detectors, and `GraphMap` (C+L) using both camera and LiDAR sensors for richer environmental perception. All models across baselines and `GraphMap` use identical training supervision and configurations for a fair and rigorous comparison.

6.5.2 End-to-end Global Vectorized Map Results

We first systematically evaluate the end-to-end performance of `GraphMap` on global vectorized HD map construction across multiple real-world intersections, road segments, and simulated town-level maps. The end-to-end evaluation aims to answer the two key questions: (1) How does `GraphMap` compare to baseline methods in terms of overall mapping accuracy? and (2) To what extent does `GraphMap` reduce network bandwidth and system-level computation costs through its design?

Global vectorized HD map accuracy. Table 6.1 and Table 6.2 present the results across multiple global intersections and road segments from the DAIR-V2X and V2XPnP datasets. Across all maps and settings, `GraphMap` significantly outperforms the baseline fusion strategies—PolyMerge [172], FDC [54], and HBM [74]—in both per-class AP and overall

Table 6.2: Performance comparison on global vectorized HD map construction on V2XPnP.

V2XPnP Map: Urban Arterial Road (UAR)

Map	Method	AP_{bound}	AP_{div}	mAP
Range: 300×450 *m*				
UAR	PolyMerge	24.0	13.1	18.5
	FDC	13.9	14.6	14.3
	HBM	14.5	0.8	7.6
	`GraphMap` (C)	**31.2**	**46.7**	**38.9**
	`GraphMap` (C+L)	28.9	45.7	37.3

V2XPnP Map: Urban Expressway (UE)

Map	Method	AP_{bound}	AP_{div}	mAP
Range: 250×600 *m*				
UE	PolyMerge	39.5	12.8	26.2
	FDC	39.0	12.1	25.6
	HBM	13.0	2.8	7.9
	`GraphMap` (C)	**52.8**	**37.6**	**45.2**
	`GraphMap` (C+L)	47.1	35.7	41.4

Table 6.3: Performance comparison on Town-level global vectorized HD map construction on OPV2V.

OPV2V Maps: Town04, Town05

Map	Method	AP_{ped}	AP_{div}	mAP
Number of Agent: 35				
04	PolyMerge	73.1	24.3	48.7
	FDC	32.8	16.7	24.8
	HBM	30.3	7.8	19.1
	`GraphMap` (C)	**92.8**	35.7	64.2
	`GraphMap` (C+L)	**92.8**	**38.0**	**65.4**
Number of Agent: 60				
05	PolyMerge	48.3	25.5	36.9
	FDC	22.6	17.7	20.2
	HBM	42.5	16.1	29.3
	`GraphMap` (C)	**86.8**	33.0	59.9
	`GraphMap` (C+L)	**86.8**	**33.2**	**60.1**

OPV2V Maps: Town06, Town10HD

Map	Method	AP_{ped}	AP_{div}	mAP
Number of Agent: 53				
06	PolyMerge	66.4	40.6	53.5
	FDC	53.2	19.6	36.4
	HBM	75.0	24.8	49.9
	`GraphMap` (C)	78.5	45.5	62.0
	`GraphMap` (C+L)	**85.7**	**45.6**	**65.7**
Number of Agent: 30				
10	PolyMerge	58.0	51.7	54.8
	FDC	35.2	16.9	26.1
	HBM	65.8	40.2	53.0
	`GraphMap` (C)	87.5	46.8	67.2
	`GraphMap` (C+L)	**93.7**	**52.9**	**73.3**

mAP. This validates the effectiveness of our graph-based sparse fusion method, MapGlue, for scalable, accurate global HD map construction. In particular, we observe that: (1) On DAIR-V2X maps, `GraphMap` (C+L) achieves an average mAP improvement of 25–50% over the best-performing baseline. When both camera and LiDAR are used (C+L), performance further improves for lane divider alignment due to better sensor coverage and more sensor features, as agents in DAIR-V2X only have one front camera; (2) For Map-08 and Map-10, our system reaches mAPs of 75.1 and 63.6 respectively—substantially higher than the next-best baseline FDC (16.6 and 15.7 in both cases). On Map-13, `GraphMap` (C) achieves perfect detection for pedestrian crossings, further demonstrating the system's robustness across diverse urban intersections; (3) On the V2XPnP dataset, which features more complex road geometries and larger agent numbers, `GraphMap` again achieves the highest performance, particularly for lane divider predictions. Notably, the gap is even more significant for the Urban Arterial Road scene, where our method leads to a 15–30% mAP gain over PolyMerge and FDC. When both camera and LiDAR are used in V2XPnP, we did not observe a significant improvement in overall performance. This is likely due to that agents in V2XPnP are equipped with multiple cameras already, capturing rich enough information for detecting

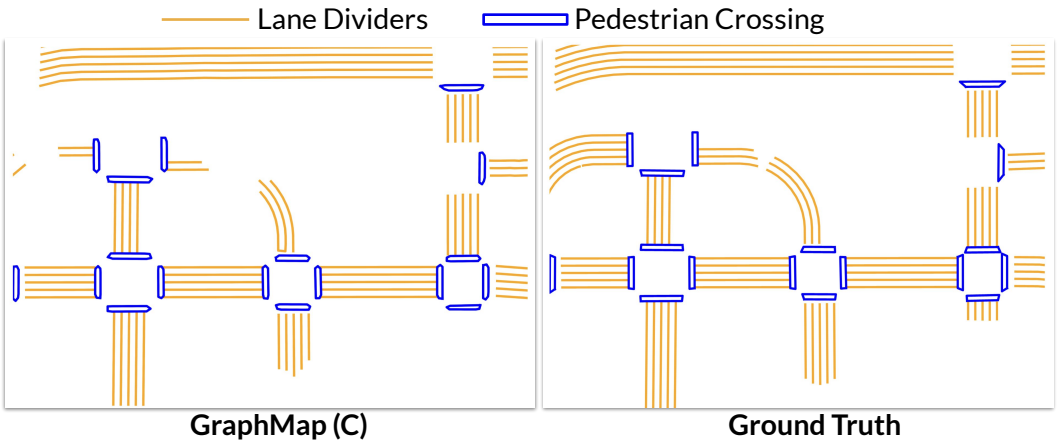
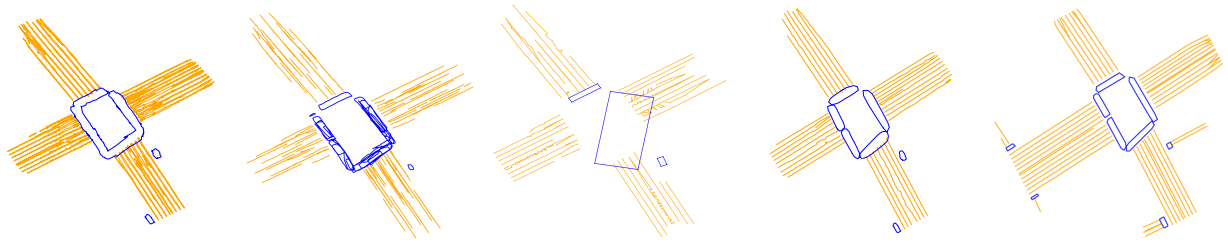


Figure 6.3: Visualization of a multi-intersection map built by `GraphMap` in OPV2V Town05.

HD map elements locally, thus providing enough information for MapGlue to construct the global map.

Large-scale emulation on OPV2V. We further evaluate `GraphMap` on the OPV2V dataset, which provides diverse cooperative driving scenarios across different **town-level maps**, varying in layout complexity, traffic density, and scales more number of participating agents. As summarized in Table 6.3, `GraphMap` consistently outperforms all baseline map fusion methods across all towns. In Town04 with 35 agents, `GraphMap` (C+L) achieves a mAP of 65.4, substantially higher than baselines with 16.7 absolute mAP. Similar improvements are observed in Town05 with 60 agents, where `GraphMap` (C+L) reaches 60.1 mAP, outperforming the best baseline by over 60%. In the more challenging Town06 and Town10HD map, `GraphMap` (C+L) achieves a mAP of 65.7, improving consistently over baseline methods by over 22%. These consistent gains demonstrate that `GraphMap` effectively maintains global map accuracy that scales with a larger number of agents and heterogeneous agent observations. Thanks to the unique design of MapGlue, `GraphMap` maintains structural continuity for map elements spanning across large areas (Fig. 6.3), highlighting the benefit of integrating local updates into the global map fusion process.

Examples of vectorized global map construction results. Fig 6.4 shows an example of global HD map construction results using different fusion methods. As shown, none of the baseline methods produces an accurate global HD map by fusing multi-agent local map observations. For example, we find that PolyMerge [172] is faster than FDC and creates smooth results for most lane dividers, but it can easily be influenced by outliers and partial observations of pedestrian crossings, making the merged pedestrian crossing inaccurate. All three baseline methods fail to generate accurate global maps, indicating the inefficiency of existing map fusion techniques for collaborative global map construction. `GraphMap` significantly



(a) HBM (b) FDC (c) PolyMerge (d) `GraphMap` (C) (e) Ground Truth

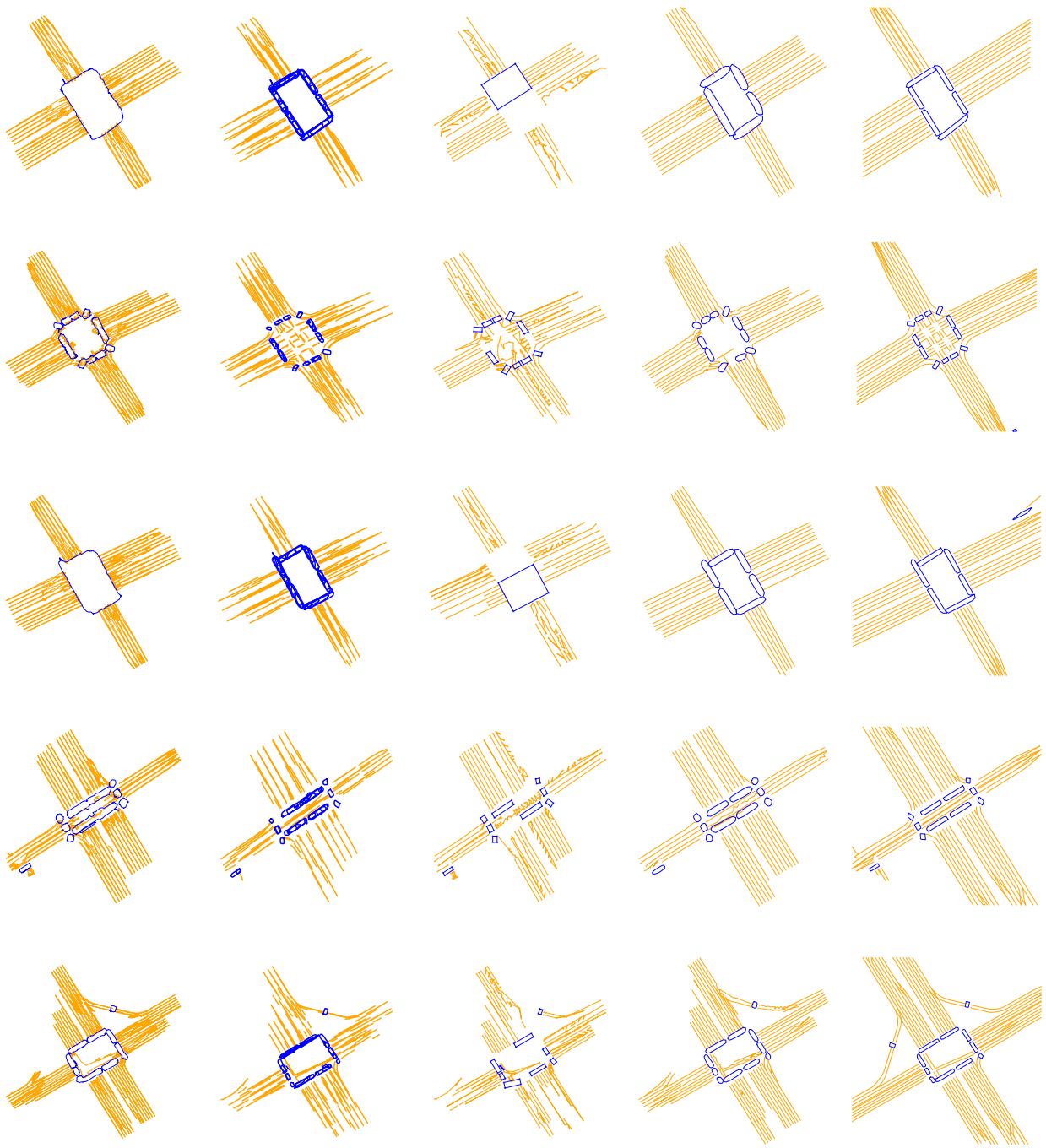
Figure 6.4: Comparison of HBM, FDC, PolyMerge, `GraphMap`, and Ground Truth on DAIR-V2X Map 06. Orange lines represent lane dividers, and blue polygons represent pedestrian crossings.

outperforms baseline methods by producing a more fine-grained and accurate representation. This improvement stems from accurate type-aware structure-preserving map matching and point-set registration, which effectively resolves misalignments caused by discrepancies in viewpoints across different vehicles when handling overlapping or conflicting map elements, as illustrated in Fig. 6.4d. Additional examples of global maps in Fig. 6.5 demonstrate that our `GraphMap` performs well across various road scenarios, thereby reinforcing its reliability and robustness.

Global map accuracy vs. bandwidth efficiency. Fig. 6.6 compares different global map construction approaches in terms of average mAP and average normalized communication bandwidth usage. Each point represents a method, with error bars denoting variance across maps evaluated. We observe that both variants of `GraphMap`, `GraphMap` (C) and `GraphMap` (C+L), substantially outperform all baselines in the accuracy-bandwidth tradeoff. Compared to baselines, which transmit all agent observations without filtering, `GraphMap` reduces communication load by 25–33% and still achieves 2–3× higher average mAP. This is attributed to our SMS mechanism, which prioritizes uploading only high-value data based on sensing value.

Notably, combined with the SMS, `GraphMap` can even achieve a 5% more accurate improvement compared to MapGlue (our ablated variant with full upload but graph-based fusion) while reducing the bandwidth requirement by over 25%. This interesting observation highlights the effectiveness of our selective data transmission in that it *not only saves network bandwidth* but also *avoids redundant and low-quality data from being uploaded* to reduce global map construction accuracy.

Computation efficiency. Fig. 6.7 shows the total computation time required to construct the global map across six DAIR-V2X maps and two V2XPnP maps over all tested frames. `GraphMap` (both C and L+C variants) achieves significantly lower compute time, due to its lightweight sparse graph representation and efficient matching algorithm. For example,



(a) HBM (b) FDC (c) PolyMerge (d) `GraphMap` (e) GT

Figure 6.5: Comparison of HBM, FDC, PolyMerge, `GraphMap`, and Ground Truth (GT) on Global Map Construction across another five different road configurations (Map 08, 09, 10, 13, 16).

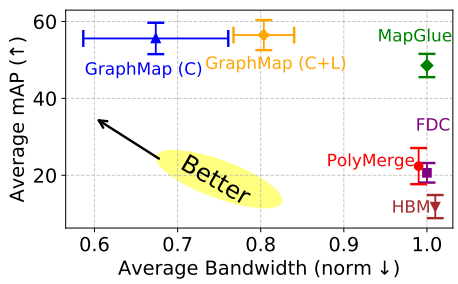


Figure 6.6: Map accuracy *vs.* BW saving.

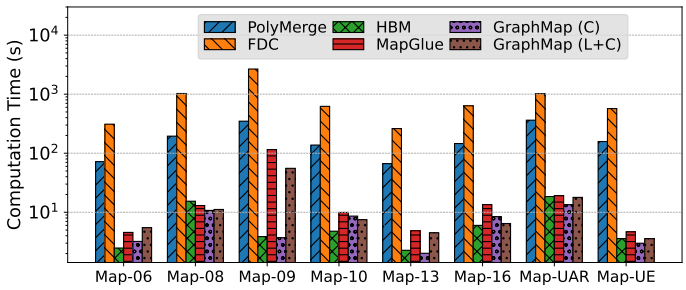


Figure 6.7: Computation cost of global map construction.

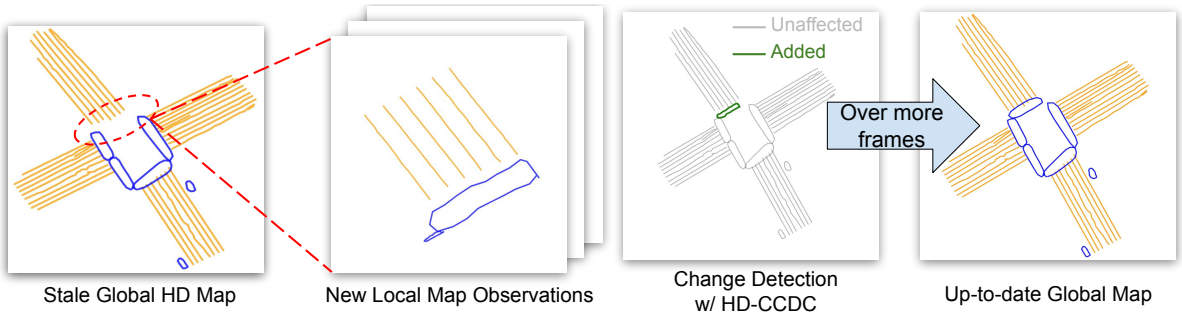


Figure 6.8: An example of global map element update detection through `GraphMap`'s HD-CCDC algorithm.

`GraphMap` (C) is 3× faster than PolyMerge and over 10× faster than FDC, which performs clustering over full pairwise trajectory distances. Even though MapGlue (without selective sensing) incurs higher bandwidth usage, it remains relatively efficient in computation due to efficient graph matching. This further confirms that our full system design balances sensing efficiency, fusion scalability, and practicality.

Performance of map update and change detection. Next, we evaluate the map element change detection using our synthetic dataset, DAIR-V2X-Change. Fig. 6.8 illustrates an example of detecting the change using the HD-CCDC algorithm (§6.3.4). The leftmost panel shows a stale global HD map with missing road markings. As new local observations are uploaded by multiple agents over time, the HD-CCDC module compares them against the existing global map and identifies candidate changes. These are visualized in the third panel: newly observed elements are marked in green (added). These elements are integrated into the global map, resulting in an up-to-date representation (right-most panel).

Fig. 6.9 summarizes the F1-Score [212] (a well-used metric that balances false positives and false negatives) for `GraphMap` to detect both addition and removal of elements. As shown, `GraphMap` significantly improves the F1 score in detecting map changes by an average of 52.9%, compared to `GraphMap` without the HD-CCDC module. Furthermore, we applied

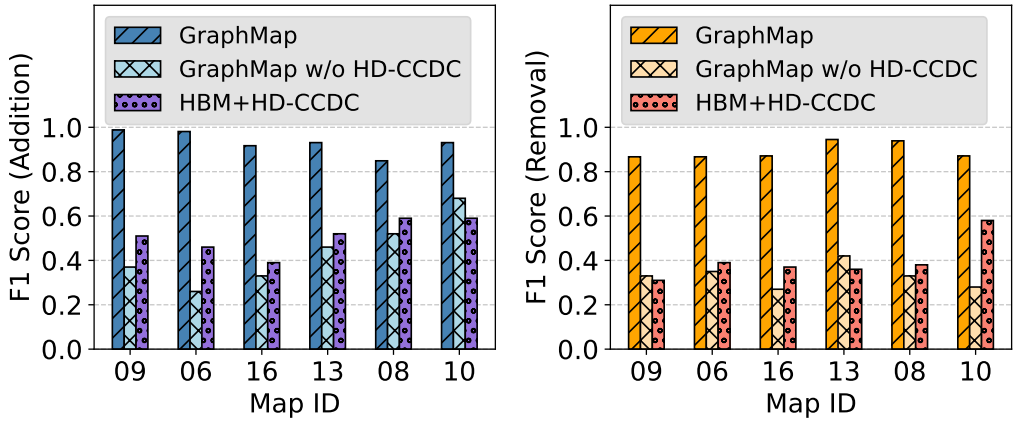


Figure 6.9: Accuracy of map element add/removal detection.

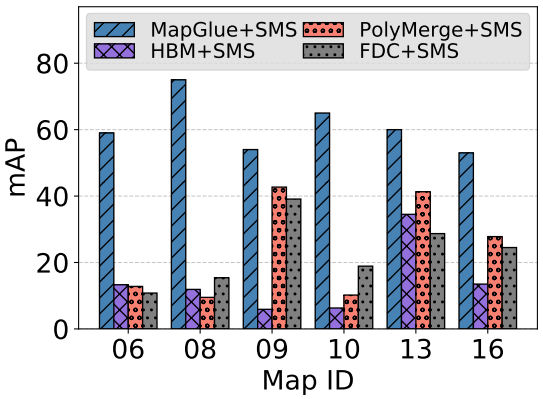


Figure 6.10: Impact of MapGlue fusion.

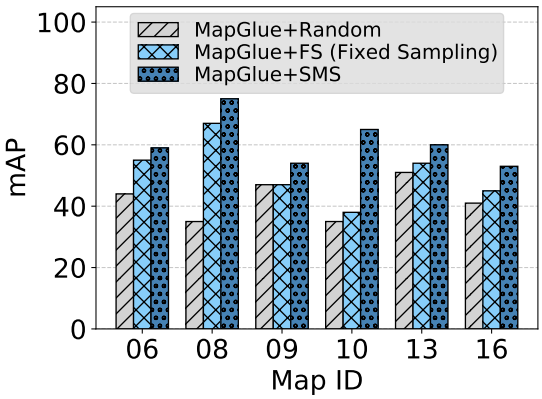


Figure 6.11: Impact of SMS strategy.

the HD-CCDC module to a baseline method HBM. We found that adding the module to a baseline can already beat `GraphMap` w/o HD-CCDC, indicating the effectiveness of our HD-CCDC module.

6.5.3 Ablation Study

Next, we conduct an ablation study to dissect the contributions of key components of `GraphMap`, and to show how our design enables robustness against environmental noise and achieves efficient, lightweight runtime performance. We run all ablation experiments on the DAIR-V2X dataset.

Impact of `GraphMap`'s key design component. Firstly, to study the effectiveness of `GraphMap`'s core map fusion algorithm, MapGlue, we replace the map fusion algorithm with other baselines while keeping the SMS for bandwidth saving for fair comparison. As shown in Fig. 6.10, MapGlue improves 37.2 absolute mAP over all baseline solutions, reinforcing the need for a type-aware, structure-preserving fusion algorithm (§6.3.2). Secondly, to study the effectiveness of `GraphMap`'s SMS module (§6.3.3), we keep the MapGlue algorithm for map

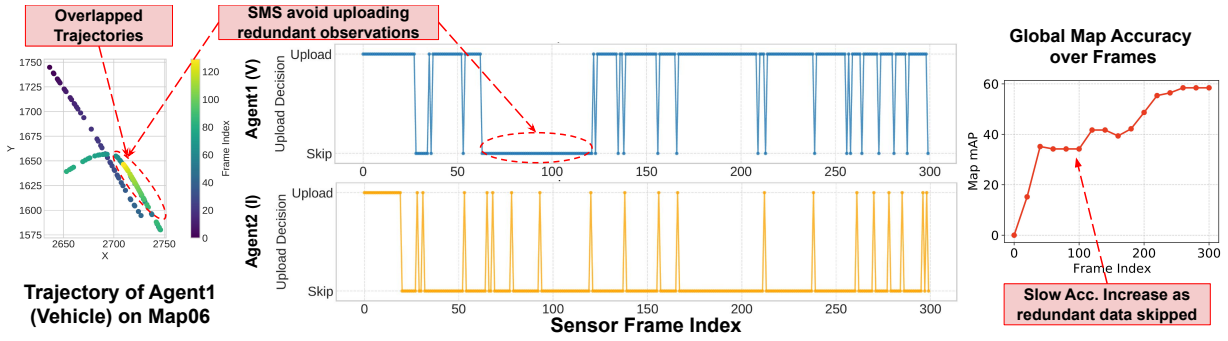


Figure 6.12: An example of a time series plot for selective map sensing on DAIR-V2X Map06.

fusion but replace the SMS with two baseline strategies that match the bandwidth savings of SMS: (1) random frame upload sampling (Random); (2) fixed frame-rate sampling (FS). Our experiments show that SMS indeed selects informative data to upload for building the global map accurately, improving over the FS and Random baseline by an average of 12.0 mAP in global accuracy under the same network bandwidth usage. Furthermore, we show a case of how SMS in `GraphMap` selects informative data and avoids redundant observations in Fig. 6.12: for agent 1 (a moving vehicle), SMS skips its data uploading as it goes through a trajectory (sensor frames 70 to 120) that has just been observed and largely overlapped. For agent2 (an infrastructure), it uploads continuously at the beginning, but then reduces its upload frequency drastically because it's stationary and its contribution to the global map becomes less over time. This is also reflected by the quick map accuracy increase at the first 20 frames. Note that the infrastructure still uploads its observation from time to time, which is powered by the MAB algorithm in SMS, making the detection of map changes easier if map elements get reconfigured (§6.3.4).

Robustness to environmental noises. We evaluate `GraphMap` under two critical and realistic types of environmental noises: *network packet loss* and agent *localization error*. These two factors are particularly important for collaborative HD map construction. Unlike latency and synchronization errors, which `GraphMap` tolerates naturally as it aims for accurate global maps, packet loss, and localization errors directly impair spatial completeness and alignment in the global map. Following statistics from prior studies [52, 143], we evaluate `GraphMap` under increasing levels of packet loss and localization noise. Fig. 6.13 and 6.14 reveal that `GraphMap` is more robust under higher noises compared to baselines, as the accuracy drop of `GraphMap` is more graceful (¡3% for 55% of packet drop rate and 0.5m of localization error). `GraphMap`'s resilience to network packet loss comes from the SMS strategy because when a frame gets scheduled to be uploaded and gets lost, the agent will redo the upload for the next observation, as the reward remains unchanged if a frame does

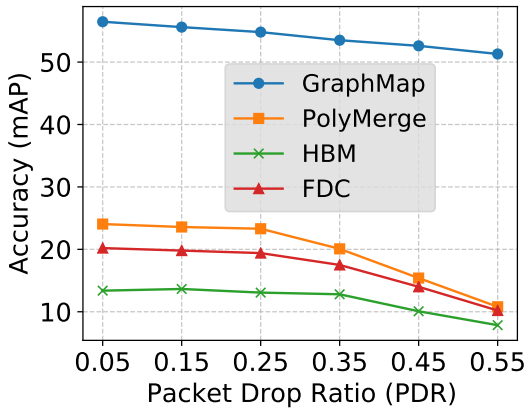


Figure 6.13: Impact of PDR.

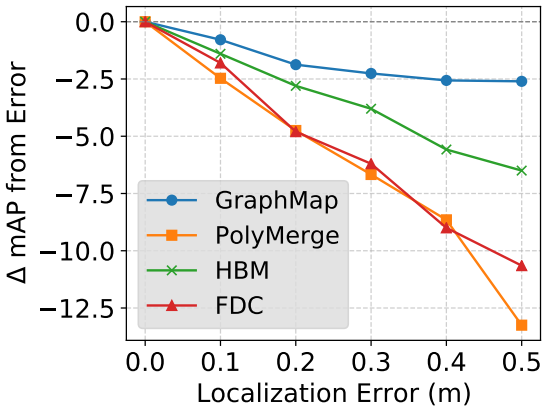


Figure 6.14: Impact of pose error.

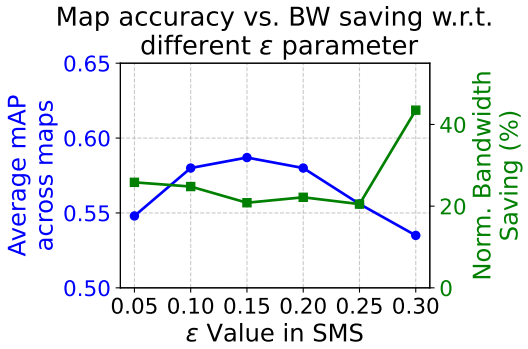


Figure 6.15: Impact of ϵ parameter.

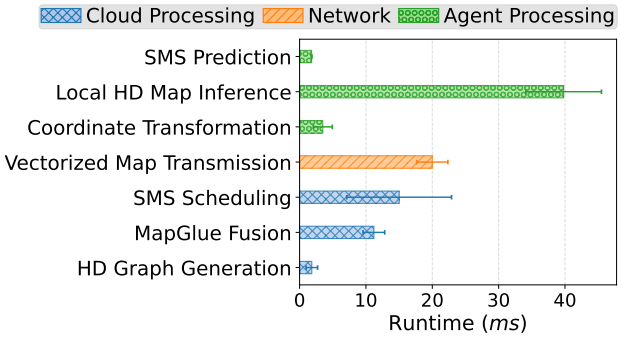


Figure 6.16: `GraphMap` Latency breakdown.

not get fused. This makes loss of frames less critical because the immediate next frame will have very close information that can contribute to the global map. `GraphMap` is also robust to localization noises because MapGlue (§6.3.2) takes drifts of map elements into account in the first place. Combined with CPD registration, spatially misaligned components still get fused accurately into the global map, thus enhancing `GraphMap`'s performance under high pose errors.

Perfomance w.r.t. different ϵ parameter in SMS. Fig. 6.15 shows the performance and bandwidth trade-offs by setting different values for the ϵ parameter in the SMS algorithm. We found that $\epsilon = 0.15$ strikes the right balance of accuracy and bandwidth saving, and thus we select it as the configuration. When ϵ becomes larger and larger, mAP decreases. At the same time, bandwidth saving will converge and saturate at 50% because $\epsilon = 1$ essentially will result in random selection.

System latency breakdown. The latency of `GraphMap` consists of agent-side processing, network transmission, and cloud processing. Fig. 6.16 provides a breakdown of different latency types. While `GraphMap` is not specifically optimized for latency, the processing in `GraphMap` is still quite efficient due to the efficient algorithm design, like MapGlue. The data

Table 6.4: Control-plane and data-plane message overhead for each sensor frame per agent (mean±std).

Metrics	GraphMap (C)	GraphMap (C+L)
Control data: GPS+IMU Pose (Bytes)	64	64
Transmission Latency (ms)	12.4±5.0	12.4±5.0
Bandwidth Requirement @ 10FPS (Mbps)	<0.1	<0.1
Sensor data: Vectorized Map Elements (KB)	7.2±2.1	8.2±2.4
Transmission Latency (ms)	19.3±7.1	20.6±7.3
Bandwidth Requirement @ 10FPS (Mbps)	0.6±0.2	0.7±0.2

transmitted over the wireless network is vectorized map elements, which are already compact and thus only consume about 20 ms to finish. Putting everything together, GraphMap results in a total of 96.5 ms, even satisfying real-time map updates at the cloud for a 10 FPS sensor rate. This opens future opportunities to develop and optimize real-time vehicle-cloud applications [95], which is left for future work.

Data message overhead. As discussed in §6.1, approaches based on dense BEV tensors suffer from high computational [132] and communication overhead [210], particularly when scaled for global map construction. This is due to the high dimensionality and redundancy inherent in dense feature maps. In GraphMap, using vectorized map elements offers a much more compact representation. For instance, a single frame of local vectorized observation in GraphMap is only 7.2 KB in size, translating to a bandwidth requirement of just 0.58 Mbps at 10 FPS (Table 6.4) or 1.73 Mbps at 30 FPS. This lightweight property makes vectorized maps especially suitable for scalable, bandwidth-efficient crowdsourced global HD mapping.

6.6 Summary

In this work, we present GraphMap, a scalable and resource-efficient framework for global high-definition (HD) map construction using crowdsourced observations from heterogeneous agents. Departing from dense, ego-centric fusion, GraphMap adopts a graph-based sparse fusion paradigm that enables robust global stitching and continuous map updates with the proposed MapGlue and HD-CCDC algorithms. GraphMap adopts a selective sensing mechanism to determine valuable observations to upload, balancing map coverage, consistency, and network resource usage.

Extensive experiments demonstrate that GraphMap significantly improves the accuracy of global HD map construction while reducing communication and computation overhead. These results validate our core insight: *sparse, selective, and semantically structured* collaboration enables scalable global HD mapping in large, dynamic environments. A full-scale real-world trace analysis on more comprehensive, city-wide datasets warrants future investi-

gation to fully showcase `GraphMap`'s capabilities as such data becomes more widely available. We hope this work will inspire further research on scalable global HD mapping and real-time map maintenance for networked and autonomous driving systems.

CHAPTER 7

Conclusion

7.1 Future Work

Future work can expand the depth and breadth of my research in several directions.

Toward a unified and robust collaborative perception system. The works in this dissertation address real-world system challenges in a modular fashion: Harbor tackles network-layer data delivery; RAO addresses temporal asynchrony; and SCORPION handles spatial-temporal misalignment under packet loss. Each was evaluated under controlled noise models derived from real-world measurements. However, in practice, these challenges do not occur in isolation: a deployed system must simultaneously contend with unreliable wireless links, clock drift across vehicles, localization noise, heterogeneous sensor quality, and varying compute budgets, all of which interact in complex and non-linear ways. A natural and important future direction is to develop a unified collaborative perception framework that holistically models and mitigates this full spectrum of system imperfections within a single end-to-end architecture. Such a framework would require co-designing the perception model with awareness of network dynamics, hardware constraints, and multi-source noise, moving beyond the current paradigm where robustness to each challenge is treated as an independent design objective.

Scaling real-world deployment of collaborative vehicular perception. While the systems developed in this dissertation are designed with real-world constraints in mind and partially validated on the MCity testbed [27, 226, 237, 227], bridging the remaining gap between controlled evaluation and full open-road deployment remains a critical challenge. Most existing collaborative perception works, including those in this dissertation [226, 237, 227], are mainly developed and benchmarked in simulation environments such as CARLA or on datasets collected under limited agent counts and controlled conditions. The real-world data experiments are often conducted by emulation through replaying data traces. In practice, real-world deployments must contend with several additional challenges that are difficult to

fully replicate in simulation and emulation: the feasibility of deploying online CVP in real-world conditions, where neural features reside in GPU memory and must be serialized, compressed, and transmitted in real time under genuine bandwidth and latency constraints [207]; agent heterogeneity, where vehicles from different manufacturers operate with different sensor suites, perception models, and feature representations, creating domain gaps that hinder effective intermediate fusion [142]; and the sim-to-real gap in perception model generalization, where models trained on simulation data degrade when exposed to real-world sensor noise, lighting variation, and environmental diversity [100, 140]. Future work should investigate domain adaptation techniques for heterogeneous collaborative agents, develop robust evaluation protocols on open-road testbeds beyond controlled mock cities, and study the impact of partial CAV penetration on collaborative perception performance to understand the minimum deployment threshold required for meaningful perception gains.

Integrating multiple sensor modalities. The perception systems developed in this dissertation primarily focus on LiDAR-based 3D object detection, which reflects the dominant modality in the collaborative perception literature. However, this design choice limits applicability to vehicles equipped with lower-cost sensor suites, where cameras may be the primary or sole sensing modality. Recent work has demonstrated that the collaborative paradigm can fundamentally change the calculus of modality choice [102]. On the multi-modal front, LiDAR-camera collaborative perception framework [230] is another promising direction to account for sensor heterogeneity. More recently, researchers [104] extended collaborative perception to incorporate 4D radar alongside LiDAR, exploiting radar's robustness to adverse weather as a complementary modality. Extending SCORPION and the RAO framework to support such multi-modal inputs would broaden their applicability and robustness. This direction also connects naturally to GraphMap, where camera-only variants already demonstrate competitive map construction performance, suggesting that modality-agnostic collaborative frameworks – capable of incorporating whatever sensors a given vehicle carries – represent an important and practically motivated research direction.

Real-time and city-scale HD map maintenance. GraphMap demonstrates accurate global map construction across town-level simulated environments and real-world multi-agent datasets, but full city-scale deployment with continuous real-time updates remains an open challenge. GraphMap's current design processes crowdsourced map contributions asynchronously in the cloud and relies on a probabilistic change detection module calibrated on a fixed set of map element categories. A fundamental bottleneck hindering progress in this direction is the scarcity of large-scale, real-world datasets that capture the full complexity of city-scale collaborative map construction and maintenance. Existing benchmarks are largely limited to small geographic extents, controlled intersection scenes, or synthetic environments,

and few provide the multi-traversal, multi-agent data with ground truth map change annotations necessary to rigorously evaluate crowdsourced update pipelines at scale. Future work can prioritize the collection and release of city-scale real-world benchmarks that cover diverse road topologies, long-horizon temporal change events, and heterogeneous vehicle fleets, which would enable systematic evaluation of scalable cloud-side fusion architectures, adaptive change detection mechanisms, and tighter onboard-cloud co-design strategies. Deeper integration between the map layer and the real-time collaborative perception stack, *e.g.,* enabling map-assisted perception and perception-assisted map updating, represents a potential promising and underexplored direction that such benchmarks would help unlock.

7.2 Concluding Remarks

This dissertation aims to advance the performance and possibility of realistic collaborative vehicular sensing applications. Motivated by the gap between the idealized assumptions of prior work and the imperfect conditions of real-world vehicular deployments, this dissertation addresses four interconnected challenges across the network, perception, and mapping layers of collaborative autonomous driving systems.

The contributions presented in this dissertation are:

- **Network system support for collaborative vehicular perception.** We presented Harbor, a hybrid V2I and V2V system that improves collaborative perception for vehicles with poor cellular connectivity by dynamically assigning nearby vehicles as helpers to relay perception data. Through a weighted bipartite matching formulation and deadline-aware MAC-layer prioritization, Harbor achieves up to 36% accuracy improvement and 18–57% latency reduction, validated on a real-world testbed with up to 100 vehicles.

- **Robustness to temporal inconsistencies in collaborative perception.** We presented RAO, a system that addresses the challenge of asynchronous sensor fusion in multi-vehicle collaboration. By combining motion-compensated occupancy flow prediction with consumer-centric on-demand data scheduling, RAO achieves a 34% improvement in perception coverage and 14% improvement in accuracy with minimal communication overhead, evaluated across simulation and real-world datasets.

- **Robustness to spatial-temporal misalignment under network imperfections.** We presented SCORPION, an end-to-end collaborative perception framework that holistically addresses packet loss, localization error, and synchronization noise through

a unified transformer architecture. By co-designing the AI layer with the underlying network and hardware constraints, SCORPION advances the state-of-the-art by 5.9–13.2 absolute AP across standard and challenging evaluation scenarios on three collaborative perception benchmarks.

- **Scalable crowdsourced global HD map construction.** We presented GraphMap, a vehicle-cloud framework that constructs accurate global vectorized HD maps from crowdsourced multi-vehicle observations via sparse graph fusion. GraphMap's type-aware structure-preserving fusion algorithm, selective map sensing policy, and probabilistic change detection module collectively enable accurate, communication-efficient, and continuously maintained global maps, outperforming state-of-the-art baselines by 37.2 mAP while reducing communication cost by over 30% across real-world and large-scale simulated multi-agent datasets.

Together, these contributions demonstrate that principled co-design across the network, perception, and mapping layers is essential for deploying collaborative autonomous driving systems in the real world. By addressing the fundamental imperfections present in real vehicular environments – from unreliable wireless links and sensor noise to the challenges of scalable map construction – this dissertation charts a path toward more robust, efficient, and practically deployable collaborative sensing systems for connected and autonomous vehicles.

APPENDIX A

Supplementary Material for SCORPION

In this supplementary material, we start with elaborating on the design details of SCORPION. Next, we provide more model details and a detailed architectural specification of parameter settings for our implementation. In the end, we present more qualitative detection results and visualization of the reconstructed feature maps from our lost BEV feature reconstruction module.

A.1 Discussion of Design Choice

A.1.1 Lost BEV Feature Reconstruction

Assumptions in Underlying V2X Network. We assume that collaborative perception applications [209, 208, 143] like SCORPION utilize UDP as the transport layer protocol, aligning with mainstream vehicular network studies [51, 52]. This decision reflects common practices within the V2X communication landscape.

Encoding Feature Map into Network Packets. Previous work [118] on the system and network community has proposed methods to encode raw point cloud data into self-contained network packets. To the best of our knowledge, no prior work has discussed how to encode and partition the intermediate feature maps into network packets. In SCORPION, we detail the process of packing the Bird's-Eye-View (BEV) features into multiple *self-contained network packets*. As mentioned previously, the payload size of each network packet is limited by Maximum Segment Size (MSS) M. In the context of V2X communication, M is in the range of 50 bytes to 300 bytes in periodical transmissions [30]. Given a BEV feature of shape (C', H, W) where C' is the channel size after compression [210], a packet can fit at most $X = \lfloor M/((C'+1) \times 4) \rfloor$ cells of features in the BEV map, where the value 4 represents the bytes needed for one floating-point number (*i.e., float32*). Therefore, we design the encoding strategy of each packet to contain feature values along with the feature indices (*i.e.,* the additional 1 in $C'+1$) into the network packet. For instance, Figure 5.5 illustrates

an example of how to encode 4 cells in a feature map of size $(16, 4, 4)$ into one packet: the payload of the packet contains $4 \times 4 = 16$ bytes of indices (*i.e.,*, 1, 2, 5, 6 in the original BEV map) and the feature values consist of $2 \times 2 \times 16 \times 4 = 256$ bytes. Hence the total number of data consumes $256 + 16 = 272$ bytes. Assuming an MSS of 280 Bytes, we leave the potential remaining 8 bytes in the MSS unused. In this way, each network packet is *self-contained*, indicating that the ego agent receiver can decode it into the proper feature position from the collaborator agents.

Practical Simulation Strategy in Lost Feature Behavior. One prior work, LCRN [126] makes assumptions that the features affected by lossy communication are *filled with random floating point numbers* (essentially random noise) in the feature map $F_j^{(t)} \in \mathbb{R}^{C \times H \times W}$. However, we contend that this approach lacks realism for two main reasons: (1) *Unrealistic Encoding and Formatting Assumptions:* it omits the actual formatting and encoding of the feature map into the network packets; (2) *Inaccurate Simulation of Packet Loss Behavior*: it simulates unrealistic packet loss behavior at the application layer, which collaborative perception operates on. According to the widely-adopted OSI network layer model [199], applications like collaborative perception either receive *a full packet* or *lost the entire content* in the packet when it gets lost during transmission. In our case, based on the encoding strategy mentioned before, the consequence of a packet loss will make the ego agent lose the entire features inside all the X cell indices in the packet. To simulate the loss of feature information more realistically, we **zero out** the feature vectors at indices where the network packet gets lost following the network loss rate [52] and reconstruct a full feature map using our designed L-BEV-R module. This approach, zeroing out features, is more aligned with real-world scenarios where the application layer receives no information about the lost features. Filling these areas with random noise would falsely imply some non-exist knowledge of the lost data, which is not practical. In contrast, setting the feature vector in indices $\{(x_1, y_1), ..., (x_n, y_n)\}$ to zero prevents the *wrong interaction* between $F_i^{(t)}$ and $F_j^{(t)}$ at these locations, equivalent to masking out these regions with lost information. This strategy ensures that our simulation of packet losses in feature sharing aligns more closely with real-world network behavior, enhancing the realism and applicability of our collaborative perception framework.

A.2 Model Details

A.2.1 Model Architecture Summary

Our model `SCORPION` is an end-to-end collaborative perception model that employs a unified Transformer architecture as depicted in Figure A.1. The three key modules in `SCORPION`:

Lost BEV Feature Reconstruction (L-BEV-R), Deformable Spatial Cross-Attention (DSCA), and Temporal Alignment (TA) are integrated as functional blocks both within and external to the Transformer architecture. The input features to the *first* transformer block $\ell = 1$ can be classified into three categories: (1) the collaborator agent feature map $F_j^{(t)}$ at the current timestamp; (2) the ego agent feature map $F_i^{(t)}$ at current timestamp; (3) past ego agent feature map $F_i^{(t-1)}$. $F_j^{(t)}$ go through the L-BEV-R block to get reconstructed feature maps $F_j^{(t)'}$ and $F_i^{(t-1)}$ go through the TA module to align the historical perceptual information $F_i^{(t-1)'}$. Then $F_j^{(t)'}$ and $F_i^{(t-1)'}$ are concatenated with the ego feature $F_j^{(t)}$ to form the input features F_ℓ to the Transformer block. The output from the first transformer block is then propagated through $L-1$ successive blocks with the same architecture to produce the final output feature F_L.

A.2.2 Model Configurations

Given the complexity of the `SCORPION` model, we articulate its entire formulation as follows:

$$F_i^{(t)} = \text{PointPillar}(X_i^{(t)}), F_i^{(t)} \in \mathbb{R}^{H\times W\times C} \tag{A.1}$$

$$F_j^{(t)} = \text{PointPillar}(X_j^{(t)}), F_j^{(t)} \in \mathbb{R}^{H\times W\times C} \tag{A.2}$$

$$F_i^{(t-1)} = \text{PointPillar}(X_i^{(t-1)}), F_i^{(t-1)} \in \mathbb{R}^{H\times W\times C} \tag{A.3}$$

$$F_j^{(t)'} = \text{L-BEV-R}(F_j^{(t)}), F_j^{(t)'} \in \mathbb{R}^{H\times W\times C} \tag{A.4}$$

$$F_i^{(t-1)'} = \text{TA}(F_i^{(t-1)}), F_i^{(t-1)'} \in \mathbb{R}^{H\times W\times C} \tag{A.5}$$

$$F_\ell = \text{Concat}(F_i^{(t)}, F_j^{(t)'}, F_i^{(t-1)'}) \tag{A.6}$$

$$F'_\ell = F_\ell + \text{MSWin}(\text{DSCA}(\text{LN}(F_\ell))), \text{where}$$
$$F_\ell \in \mathbb{R}^{(N+1)\times H\times W\times C} \tag{A.7}$$

$$F_{\ell+1} = F'_\ell + \text{MLP}(\text{LN}(F'_\ell)), \ell = 1, \ldots, L-1 \tag{A.8}$$

$$F_{i_fused}^{(t)} = F_L[i], F_{i_fused}^{(t)} \in \mathbb{R}^{H\times W\times C} \tag{A.9}$$

$$y = \text{DetectionHead}(F_{i_fused}^{(t)}) \tag{A.10}$$

where the input $X_i^{(t)}$ denotes the raw LiDAR point clouds captured on each agent, which go through the PointPillar Encoder [120], generating BEV features $F_j^{(t)}$ for each agent j. These tensors are then compressed, transmitted through a lossy V2X communication (except for the ego feature $F_i^{(t)}$), and decompressed at the ego agent side. The partial features are then reconstructed using the L-BEV-R module and concatenated with the ego feature and the processed historical feature using the TA module. Next, the integrated features from N

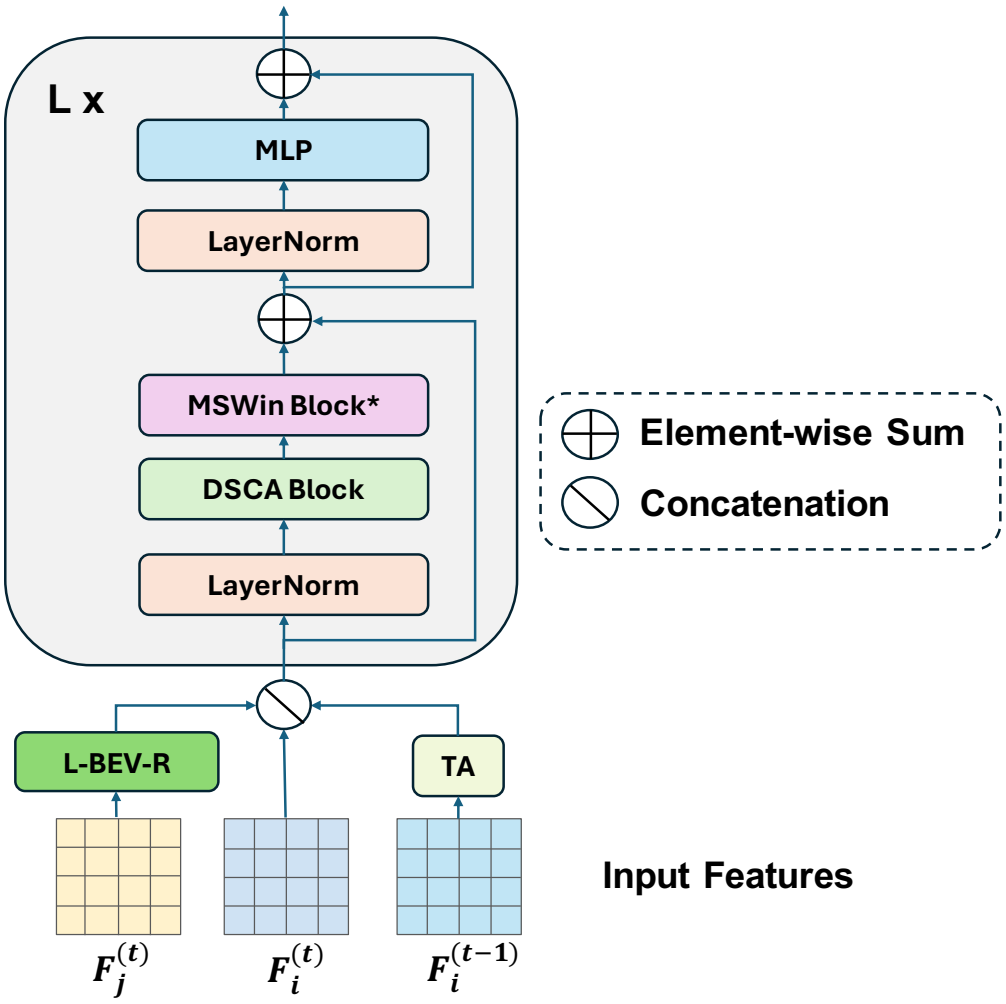


Figure A.1: The detailed transformer architecture of our proposed SCORPION. We use the same MSWin Block (*) from [209].

agents are processed using our proposed SCORPION, which consists of L layers of the SCORPION transformer blocks. Each transformer block contains a DSCA block, a multi-scale window fusion module from [209], and a standard MLP layer [84]. Following [73, 191], we apply Layer Normalization [191] before feeding the inputs into the DSCA and MLP modules. We show the detailed specifications of SCORPION architecture in Table A.1. While the code for SCORPION remains proprietary, these specifications are designed to be comprehensive enough to assist researchers in understanding and potentially replicating the SCORPION architecture.

Table A.1: Detailed architectural specifications for SCORPION.

Component	Output size	SCORPION framework
PointPillar Encoder	$N \times 176 \times 48 \times 256$	[Voxel resolution 0.4m, Scatter, 64] [Conv3x3, 64, stride 2, BN, ReLU] $\times 3$ [Conv3x3, 128, stride 2, BN, ReLU] $\times 5$ [Conv3x3, 256, stride 2, BN, ReLU] $\times 8$ [ConvT3x3, 128, stride 1, BN, ReLU] $\times 1$ [ConvT3x3, 128, stride 2, BN, ReLU] $\times 1$ [ConvT3x3, 128, stride 4, BN, ReLU] $\times 1$ [Concat3, 384] [Conv3x3, 256, stride 2, ReLU] $\times 1$ [Conv3x3, 256, stride 1, ReLU] $\times 1$
L-BEV-R module	$N \times 176 \times 48 \times 256$	[PatchEmbed, $4 \times 4 \times 256, 768$] $\times 1$ [Transformer Block [35], dim 768, head 16] $\times 12$ [Linear, 768, 512] $\times 1$ [Transformer Block [35], dim 512, head 16] $\times 8$ [Linear, 512, $4 \times 4 \times 256$] $\times 1$
TA module	$N \times 176 \times 48 \times 256$	[Affine grid, dim 176×48] $\times 1$
Transformer Backbone	$N \times 176 \times 48 \times 256$	[DSCA dim 256, head 8] $\times 3$ [MSwin [209], dim 256, head {16, 8, 4}] $\times 3$ [MLP, dim 256] $\times 3$
Detection Head	$176 \times 48 \times 16$	Cls. head: [Conv1x1, 2, stride 1] Regr. head: [Conv1x1, 1, stride 1]

A.3 More Experimental Results

A.3.1 Various Realistic V2X Network Loss Rate

The performance of the V2X network is sensitive to many factors, including road topology, traffic complexity, *etc.*. Leveraging data from a comprehensive V2X network loss measurement study [52], we conducted experiments to evaluate the impact of realistic packet loss rates in diverse road settings. Each setting presents a common driving setting affecting communication reliability between vehicles:

1. **Urban Freeway (UF)**: Measured at an eight-lane freeway that features numerous walls, tunnels, and overhead bridges, and experiences heavy vehicle traffic. The average packet drop rate is 26% on a 100-meter separation distance between vehicles, a typical distance threshold for collaborative perception communication [210, 143].

2. **Rural Freeway (RF)**: Measured at a six-lane freeway bordered by open lands, with buildings and trees set back from the guardrail. It features fewer walls, tunnels, and vehicle traffic compared to the Urban Freeway (UF). The average packet loss rate is

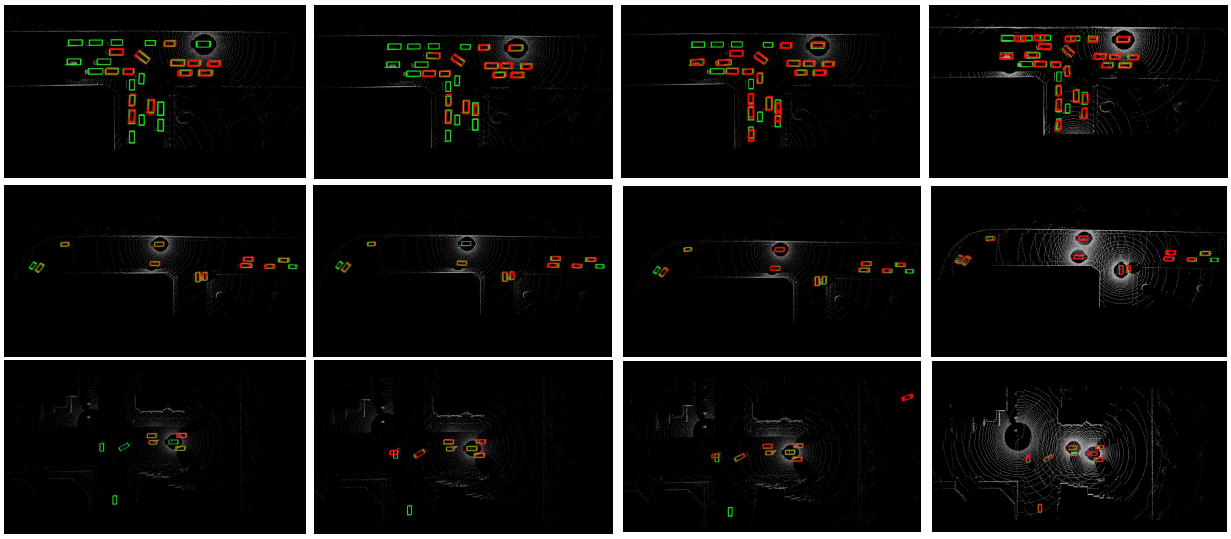


Figure A.2: Additional Visualization results on the V2XSet dataset under localization error, synchronization error, and network loss altogether. The green and red boxes are ground truth and prediction outputs, respectively.

20%.

3. **Rural Road (RR)**: Measured at a two-lane street, with one lane in each direction. It is characterized by scattered trees, houses, and hills in the surrounding area. The average packet drop rate is 32%.

4. **Suburban Road (SR)**: This environment consists of 6-lane suburban streets with widths ranging from 30 to 50 meters. Light traffic is typically present in this area. The average packet loss rate is 15%.

5. **Open Field (OF)**: This setting features a road area with no buildings and very light traffic. The average packet loss rate is 12%.

Table 5.3 demonstrates the AP0.5 and AP0.7 metric of `SCORPION` over the above five scenarios on the V2XSet dataset: `UF`, `RF`, `RR`, `SR`, `OF`. We select 4 SOTA baseline methods for comparison: CoAlign [143], V2X-ViT [209] and SCOPE [216].

Our model demonstrates robust performance across all scenarios, reflecting its superior handling of different network loss rates. It consistently outperforms the second-best method, improving the Average Precision (AP) at 0.5 and 0.7 IoU thresholds. The improvements range from 3.1 to 6.0 in AP0.5 and from 2.4 to 5.5 in AP0.7, measured in absolute percentage points. This indicates not only the effectiveness of `SCORPION` in maintaining high accuracy under variable network conditions but also its practical applicability in diverse real-world environments.

Table A.2: Effect of feature fusion strategy in DSCA (AP0.5/0.7).

DSCA Fusion Method	V2XSet	OPV2V	DAIR-V2X
Max. Fusion	88.32/77.88	93.10/85.10	74.65/56.76
Avg. Fusion	88.25/76.02	91.57/82.52	72.68/54.31

A.3.2 Impact of Feature Fusion Strategy in DSCA

In the DSCA module of SCORPION, we choose to use a simple max-fusion to aggregate the attention map after performing the deformable attention between the ego agent and every other collaborator agent. We compare our empirical selection of max-fusion with another widely used feature fusion method, average fusion.

The comparative results, as presented in Table A.2, demonstrate that max-fusion consistently outperforms average fusion in terms of AP at both 0.5 and 0.7 thresholds. Specifically, max-fusion enables the model to capture and prioritize critical features by selecting the largest values across the feature interactions at each spatial location. In contrast, average fusion may dilute these critical values, potentially attenuating important perceptual signals.

Therefore, utilizing max-fusion within our DSCA module has proven to be an effective strategy for enhancing perception performance. While this simple approach already yields substantial benefits, the exploration of more sophisticated feature fusion strategies to further elevate performance remains an area for future research, as suggested by recent studies [225, 216].

The results in Table A.2 suggest that using max-fusion is more effective in producing better perception AP performance than average feature fusion. We attribute this phenomenon to the reason that using element-wise max fusion provides the ability to capture critical perception by selecting the large feature values across the interaction between the ego agent and collaborator agent pair on each spatial location (where average may attenuate such effects). Therefore, we find that applying a simple max fusion in our DSCA module is already effective for producing good perception performance. Integration with more complex feature fusion strategy to further boost the perception performance [225] is left as future work.

A.3.3 Qualitative Results

Visualization of Detection Results. In this part, we show more qualitative visualizations of the detection results of SCORPION compared to baseline methods. Figure A.2 demonstrates more detection visualizations of CoBEVT, V2X-ViT, CoAlign, and SCORPION under challenging setting with 1.3 m of localization error, 100 ms of asynchrony latency, and 35% of

network loss (same as Sec. 4.3 in the paper). Overall, SCORPION consistently demonstrates superior robustness, characterized by fewer regression displacements and a reduced number of undetected objects, thereby highlighting its effectiveness in challenging environments.

APPENDIX B

Supplementary Material for GraphMap

B.1 Motivation for Sparse Graph Fusion on Global Map Construction

Local HD map performance. Fig. B.1 illustrates examples of local HD map prediction from vehicle and infrastructure views. The local vectorized HD map construction quality is quite accurate using the STOA approach, MapTRv2 [134]. However, the issue arises in producing a consistent and accurate global map by directly adapting the models to larger ranges (Tab. B.1 and Tab. B.2).

Applying local map models to larger ranges. However, if we retrain the local HD map model to fit to larger range for global map detection, its performance degrades significantly with the increasing perception range, as verified by previous local online HD map work [221, 196]. The mAP accuracy drops over 60% (30%) if we increase the perception range for the vehicle (infra) agent from 30×30 m to 120×60 m (75×50 m to 150×80 m). Such a drastic drop in accuracy motivates us to move from single-vehicle and dense, ego-centric methods to a sparse fusion method.

Why does `GraphMap` choose to upload vectorized map observations? As discussed in §6.1, approaches based on dense BEV tensors suffer from high computational [132] and communication overhead [210], particularly when scaled for global map construction. This is due to the high dimensionality and redundancy inherent in dense feature maps. In contrast, vectorized map elements offer a much more compact representation. For instance, a single frame of local vectorized observation in `GraphMap` is only 7.2 KB in size, translating to a

Table B.1: Performance of local vectorized HD map prediction.

Range	Method	Agent/Method	AP_{ped}	AP_{div}	mAP
30×30 m	MapTRv2	Vehicle-local	0.745	0.909	0.827
75×50 m	MapTRv2	Infra-local	0.772	0.785	0.778

Table B.2: Performance on local HD map prediction on larger area.

Range	Method	Agent/Method	AP_{ped}	AP_{div}	mAP
120×60 m	MapTRv2	Vehicle-local	0.182	0.103	0.143
150×80 m	MapTRv2	Infra-local	0.399	0.482	0.440

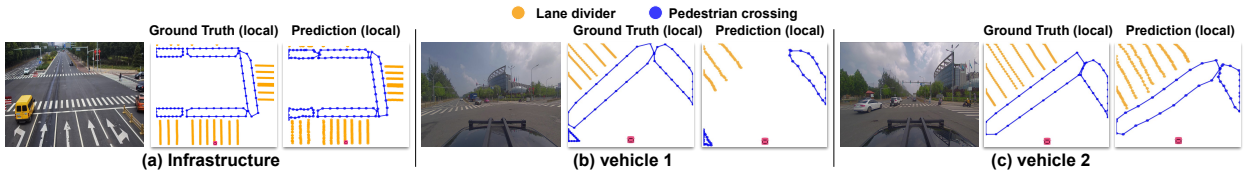


Figure B.1: Visualization of local vectorized HD map construction using infrastructure and vehicle sensors.

bandwidth requirement of just 1.73 Mbps at 30 FPS or 0.58 Mbps at 10 FPS (Tab. 6.4). This lightweight property makes vectorized maps especially suitable for scalable, bandwidth-efficient crowdsourced global HD mapping.

B.2 Background on Multi-Armed Bandit Algorithm

The Multi-Armed Bandit (MAB) problem, a classic sequential decision-making challenge, involves strategically allocating limited resources among choices (*arms*) with initially unknown reward probabilities, aiming to maximize cumulative gain over time [169]. The core task is to devise an arm selection strategy that optimally balances learning these unknown reward distributions with the goal of accumulating the highest possible rewards. The MAB problem addresses the fundamental exploration-exploitation trade-off. Exploitation means choosing the arm currently believed to be best based on past outcomes to maximize immediate rewards. Exploration, conversely, involves sampling less certain arms to gather more information, discovering superior long-term options even if it means foregoing some immediate gains. Successful MAB strategies effectively balance these conflicting objectives, often aiming to minimize *regret*—the performance difference between the chosen strategy and an optimal one with full knowledge.

Numerous algorithms navigate this trade-off. The foundational ϵ-greedy algorithm, for instance, predominantly exploits the best-known arm but explores a random arm with a small probability ϵ [185]. The MAB framework and its variants have found wide application in diverse fields, including clinical trials, A/B testing in web design, recommendation systems, financial portfolio optimization, and dynamic resource allocation in communication networks, underscoring its versatility in modeling sequential decision problems under uncertainty [55, 121]. In `GraphMap`, we extend ϵ-greedy MAB to a feedback-delayed, double-ended setting tailored for decentralized, multi-agent HD map construction—partitioning reward estimation between agent-side and cloud-side in a way that enables asynchronous, resource-aware upload decisions across heterogeneous sensing agents.

B.3 Details of DAIR-V2X-Change Benchmark Statistics

The DAIR-V2X-Change benchmark creation involves two types of modifications to reflect realistic reconfigurations such as those caused by construction or road repaintings. To simulate element removals, we randomly select and remove 5 to 10 lane dividers and 2 to 3 pedestrian crossings from the ground-truth map of a given scene. For element additions, we first build a library of realistic lane dividers and pedestrian crossing shapes by sampling them from the original dataset. These predefined elements are then randomly patched onto different map locations, typically near existing map elements to ensure plausible placement, with variations in rotation and positional offset applied to increase diversity. The system is then expected to detect these changes—either omissions or additions—by comparing unaltered sensor data to the now-altered prior map and updating its understanding accordingly.

On average, each map in the benchmark is modified with 8 lane dividers and 2.5 pedestrian crossing changes. The benchmark spans 6 distinct intersection areas and includes 60 maps with element additions and 60 maps with removals, resulting in a total of 120 manually altered maps for a comprehensive evaluation of the system's change detection and map maintenance capabilities.

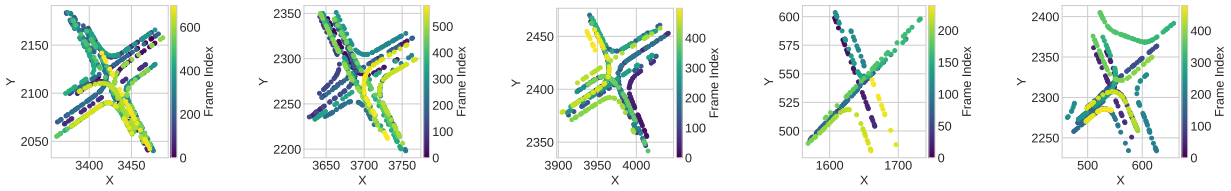
B.4 Details on Baseline Approaches

In this section, we elaborate on more details of the baseline methods used in our evaluation. These baselines serve as key reference points for understanding the performance gains and advantages introduced by our approach.

• **PolyMerge** [172]: This represents a state-of-the-art rule-based mechanism for merging vectorized polylines. Its approach, based on spatial proximity and geometric heuristics, is a common and intuitive strategy for direct vector map fusion. It has been used as a comparative method in recent works [234]. Due to the lack of open-source code, we reimplement PolyMerge following the methodology detailed in the original paper.

• **Fréchet Distance Clustering (FDC)** [54]: As a strong representative of trajectory similarity-based clustering approaches, FDC is employed to merge polylines and polygons that exhibit high geometric similarity. It is a robust technique for grouping and consolidating similar map elements based on their intrinsic shapes, often considered a more sophisticated alternative to simple proximity. These metrics are frequently benchmarked in trajectory analysis [89, 64].

• **Heatmap Buffer Merge (HBM)** [74]: This baseline embodies a common alternative fusion paradigm that involves an intermediate raster representation. It rasterizes vectorized predictions into heatmaps, applies a buffer to the polylines within this dense representation



(a) Map 08 (b) Map 09 (c) Map 10 (d) Map 13 (e) Map 16

Figure B.2: Agent trajectories on the other five different road configurations (Map 08-16).

to facilitate the merging of nearby elements, and then converts the result back to vector format [74]. Raster-based fusion followed by vectorization is a technique explored in various mapping works [59, 176].

B.5 Additional Qualitative Visualizations

Visualization of Agent Trajectories From The DAIR-V2X Dataset. To better understand agents' sensing coverage and sensing perspectives in our experiments, we visualize the trajectories of all agents in the DAIR-V2X dataset. Fig. B.2 visualizes agent trajectories across five DAIR-V2X maps, with each dot colored by frame index. These trajectories highlight the spatial distribution of vehicle and infrastructure observations that collectively contribute to global map fusion. While the current datasets span areas on the order of tens of hectares (a subset of the area in a city-level map), `GraphMap` is designed to support much broader, city-scale map buildings. Our graph-based sparse fusion design and selective sensing policies inherently scale with the number of agents and spatial extent. Once full-scale, city-wide traces become available, we believe that applying `GraphMap` to them will further validate its ability to construct, maintain, and update global HD maps at scale.

BIBLIOGRAPHY

[1] Tesla Model S software release notes v5.8. https://tinyurl.com/3zxz75c3, 2013.

[2] The KITTI benchmark suite. https://tinyurl.com/43a7kphb, 2013.

[3] AT&T and Audi to Wirelessly Connect all 2016 Model Year Vehicles. https://tinyurl.com/ynwvtbmv, 2015.

[4] Mercedes-Benz mbrace. https://tinyurl.com/22nh84m7, 2016.

[5] C-V2X Performance Assessment Project. https://tinyurl.com/2s3f2h9u, 2019.

[6] Intersection over Union (IoU). https://cocodataset.org/#detection-eval, 2019.

[7] Study shows autonomous vehicles can help improve traffic flow. https://phys.org/news/2018-02-autonomous-vehicles-traffic.html, 2019.

[8] Autonomous vehicles for safety. https://tinyurl.com/yvrenr55, 2020.

[9] Fatality Analysis Reporting System. https://tinyurl.com/25yfs73z, 2020.

[10] AT&T Maps - Wireless Coverage. https://www.att.com/maps/wireless-coverage.html, 2021.

[11] Draco 3D Graphics Compression. https://google.github.io/draco/, 2021.

[12] FCC Mobile Broadband Maps. https://www.fcc.gov/BroadbandData/MobileMaps, 2021.

[13] Linux kernel v5.8. https://github.com/torvalds/linux/tree/v5.8, 2021.

[14] Mininet-wifi: Emulator for software-defined wireless networks. https://github.com/intrig-unicamp/mininet-wifi, 2021.

[15] Next generation simulation (ngsim) dataset. https://tinyurl.com/2wk25efw, 2021.

[16] Share GPS. http://jillybunch.com/sharegps/, 2021.

[17] T-Mobile Coverage Map. https://www.t-mobile.com/coverage/coverage-map/, 2021.

[18] Velodyne LiDAR HDL-32E sensor. https://velodynelidar.com/products/hdl-32e/, 2021.

[19] Verizon Coverage Map. https://www.verizon.com/coverage-map/, 2021.

[20] ZED Stereo Camera. https://www.stereolabs.com/zed/, 2021.

[21] Autoware: Open-source software for self-driving vehicles. https://github.com/Autoware-AI, 2022.

[22] Baidu Apollo. https://apollo.auto, 2022.

[23] Cohda Wireless OBU Solution. https://www.cohdawireless.com/solutions/mk6/, 2022.

[24] OpenCDA. https://github.com/ucla-mobility/OpenCDA/, 2022.

[25] SUMO: Simulation of Urban Mobility. https://www.eclipse.org/sumo/, 2022.

[26] CARLA: Open-source simulator for autonomous driving research. https://carla.org/, 2023.

[27] Mcity - University of Michigan. https://mcity.umich.edu/, 2023.

[28] Speed limits in the United States. https://tinyurl.com/mspj8e4j, 2023.

[29] Velodyne LiDAR. https://velodynelidar.com/, 2023.

[30] Introduction to the vehicle-to-everything communications service V2X feature in 3GPP release 14. https://tinyurl.com/5n82zfzv, 2024.

[31] NVIDIA A100 Tensor Core GPU. https://www.nvidia.com/en-us/data-center/a100/, 2024.

[32] NVIDIA AGX Systems. https://tinyurl.com/2f7awneb, 2024.

[33] Open COOperative Detection framework for autonomous driving. https://github.com/DerrickXuNu/OpenCOOD, 2024.

[34] OXTS Inertial Navigation Systems. https://www.oxts.com/, 2024.

[35] PyTorch Image Models. https://github.com/huggingface/pytorch-image-models, 2024.

[36] Navigation data standard (nds). https://nds-association.org/, 2025.

[37] Opendrive format specification. https://www.asam.net/standards/detail/opendrive/, 2025.

[38] Putting our robots on the map. https://tinyurl.com/55va3smm, 2025.

[39] Uber av: Autonomous mobility and delivery. https://www.uber.com/us/en/autonomous/, 2025.

[40] Waymo - self-driving cars - autonomous vehicles - ride-hail. https://waymo.com/, 2025.

[41] The waymo driver handbook: How our highly-detailed maps help unlock new locations for autonomous driving. https://waymo.com/blog/2020/09/the-waymo-driver-handbook-mapping, 2025.

[42] Zoox: It's not a car — zoox. https://zoox.com/, 2025.

[43] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11 b mesh network. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 121–132, 2004.

[44] Fawad Ahmad, Hang Qiu, Ray Eells, Fan Bai, and Ramesh Govindan. {CarMap}: Fast 3d feature map updates for automobiles. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 1063–1081, 2020.

[45] Faez Ahmed, John P Dickerson, and Mark Fuge. Diverse weighted bipartite b-matching. *arXiv preprint arXiv:1702.07134*, 2017.

[46] Ganesh Ananthanarayanan, Venkata N Padmanabhan, Lenin Ravindranath, and Chandramohan A Thekkath. Combine: leveraging the power of wireless peers through collaborative downloading. In *MobiSys*. ACM, 2007.

[47] Eduardo Arnold, Mehrdad Dianati, Robert De Temple, and Saber Fallah. Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):1852–1864, 2020.

[48] Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

[49] David Avis and David Bremner. How good are convex hull algorithms? In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 20–28, 1995.

[50] Farzad Azizi, Benyamin Teymuri, Rojin Aslani, Mehdi Rasti, Jesse Tolvaneny, and Pedro HJ Nardelli. Mix-mab: Reinforcement learning-based resource allocation algorithm for lorawan. In *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*, pages 1–6. IEEE, 2022.

[51] Fan Bai and Hariharan Krishnan. Reliability analysis of dsrc wireless communication for vehicle safety applications. In *2006 IEEE intelligent transportation systems conference*, pages 355–362. IEEE, 2006.

[52] Fan Bai, Daniel D Stancil, and Hariharan Krishnan. Toward understanding characteristics of dedicated short range communications (dsrc) from a perspective of vehicular network engineers. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 329–340, 2010.

[53] Samuel M Bateman, Ning Xu, H Charles Zhao, Yael Ben Shalom, Vince Gong, Greg Long, and Will Maddern. Exploring real world map change generalization of prior-informed hd map prediction models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4568–4578, 2024.

[54] Philippe C Besse, Brendan Guillouet, Jean-Michel Loubes, and François Royer. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3306–3317, 2016.

[55] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[56] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[57] Ian D Chakeres and Elizabeth M Belding-Royer. Aodv routing protocol implementation design. In *24th International Conference on Distributed Computing Systems Workshops, 2004. Proceedings.*, pages 698–703. IEEE, 2004.

[58] Chang Chen, Jiaming Zhang, Kailun Yang, Kunyu Peng, and Rainer Stiefelhagen. Trans4map: Revisiting holistic bird's-eye-view mapping from egocentric images to allocentric semantics with vision transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4013–4022, 2023.

[59] Jiacheng Chen, Yuefan Wu, Jiaqi Tan, Hang Ma, and Yasutaka Furukawa. Maptracker: Tracking with strided memory fusion for consistent vector hd mapping. In *European Conference on Computer Vision*, pages 90–107. Springer, 2024.

[60] Pengxin Chen, Xiaoqi Jiang, Yingjun Zhang, Jiahao Tan, and Rong Jiang. Mapcvv: On-cloud map construction using crowdsourcing visual vectorized elements towards autonomous driving. *IEEE Robotics and Automation Letters*, 2024.

[61] Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 88–100, 2019.

[62] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 514–524. IEEE, 2019.

[63] Shaoyu Chen, Yunchi Zhang, Bencheng Liao, Jiafeng Xie, Tianheng Cheng, Wei Sui, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Vma: Divide-and-conquer vectorized map annotation system for large-scale driving scene. *arXiv preprint arXiv:2304.09807*, 2023.

[64] Siu-Wing Cheng and Haoqiang Huang. Curve simplification and clustering under fréchet distance. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1414–1432. SIAM, 2023.

[65] Minkyoung Cho, Yulong Cao, Jiachen Sun, Qingzhao Zhang, Marco Pavone, Jeong Joon Park, Heng Yang, and Z Morley Mao. Cocoon: Robust multi-modal perception with uncertainty-aware sensor fusion. *arXiv preprint arXiv:2410.12592*, 2024.

[66] Minkyoung Cho, Yulong Cao, Zixiang Zhou, and Z Morley Mao. Adopt: Lidar spoofing attack detection based on point-level temporal consistency. *arXiv preprint arXiv:2310.14504*, 2023.

[67] Amit Chougule, Vinay Chamola, Aishwarya Sam, Fei Richard Yu, and Biplab Sikdar. A comprehensive review on limitations of autonomous driving and its impact on accidents and collisions. *IEEE Open Journal of Vehicular Technology*, 2023.

[68] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol (olsr). 2003.

[69] Jiaxun Cui, Hang Qiu, Dian Chen, Peter Stone, and Yuke Zhu. Coopernaut: End-to-end driving with cooperative perception for networked vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17252–17262, 2022.

[70] Mobileye Developers. Mobileye: Driver assist and autonomous driving technologies., 2025.

[71] Kakan Chandra Dey, Anjan Rayamajhi, Mashrur Chowdhury, Parth Bhavsar, and James Martin. Vehicle-to-vehicle (v2v) and vehicle-to-infrastructure (v2i) communication in a heterogeneous wireless network–performance evaluation. *Transportation research part c: emerging technologies*, 68:168–184, 2016.

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

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

[74] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973.

[75] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d lidar point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, 2011.

[76] Sandeep D'souza, Heiko Koehler, Akhilesh Joshi, Satyam Vaghani, and Ragunathan Rajkumar. Quartz: time-as-a-service for coordination in geo-distributed systems. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 264–279, 2019.

[77] Renaud Dubé, Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. An online multi-robot slam system for 3d lidars. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1004–1011. IEEE, 2017.

[78] Hans-Christian Ehrlich and Matthias Rarey. Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(1):68–79, 2011.

[79] Mohammed Elbadry, Fan Ye, and Peter Milder. Wireless multicast rate control adaptive to application goodput and loss requirements. In *2024 IEEE/ACM Ninth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 25–36. IEEE, 2024.

[80] Eyal Even-Dar, Shie Mannor, Yishay Mansour, and Sridhar Mahadevan. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(6), 2006.

[81] Kevin R Fall and W Richard Stevens. *Tcp/ip illustrated*, volume 1. Addison-Wesley Professional, 2012.

[82] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[83] Alexander Frickenstein, Manoj-Rohit Vemparala, Jakob Mayr, Naveen-Shankar Nagaraja, Christian Unger, Federico Tombari, and Walter Stechele. Binary dad-net: Binarized driveable area detection network for autonomous driving. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2295–2301. IEEE, 2020.

[84] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[85] Moinak Ghoshal, Imran Khan, Z Jonny Kong, Phuc Dinh, Jiayi Meng, Y Charlie Hu, and Dimitrios Koutsonikolas. Performance of cellular networks on the wheels. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, pages 678–695, 2023.

[86] Moinak Ghoshal, Z Jonny Kong, Qiang Xu, Zixiao Lu, Shivang Aggarwal, Imran Khan, Yuanjie Li, Y Charlie Hu, and Dimitrios Koutsonikolas. An in-depth study of uplink performance of 5g mmwave networks. In *Proceedings of the ACM SIGCOMM*

Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases, pages 29–35, 2022.

[87] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5692–5703, 2021.

[88] Offer Grembek, Alex Kurzhanskiy, Aditya Medury, Pravin Varaiya, and Mengqiao Yu. Making intersections safer with i2v communication. *Transportation Research Part C: Emerging Technologies*, 102:396–410, 2019.

[89] Joachim Gudmundsson and Nacho Valladares. A gpu approach to subtrajectory clustering using the fréchet distance. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 259–268, 2012.

[90] In-Chul Ha. Kinematic parameter calibration method for industrial robot manipulator using the relative position. *Journal of mechanical science and technology*, 22(6):1084–1090, 2008.

[91] Ahmad Hassan, Arvind Narayanan, Anlan Zhang, Wei Ye, Ruiyang Zhu, Shuowei Jin, Jason Carpenter, Z. Morley Mao, Feng Qian, and Zhi-Li Zhang. Vivisecting mobility management in 5g cellular networks. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 86–100. Association for Computing Machinery, 2022.

[92] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[93] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[94] Ying He, Bin Liang, Jun Yang, Shunzhi Li, and Jin He. An iterative closest points algorithm for registration of 3d laser scanner point clouds with geometric features. *Sensors*, 17(8):1862, 2017.

[95] Yuze He, Chen Bian, Jingfei Xia, Shuyao Shi, Zhenyu Yan, Qun Song, and Guoliang Xing. Vi-map: Infrastructure-assisted real-time hd mapping for autonomous driving. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.

[96] Yuze He, Li Ma, Zhehao Jiang, Yi Tang, and Guoliang Xing. Vi-eye: Semantic-based 3d point cloud registration for infrastructure-assisted autonomous driving. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 573–586, 2021.

[97] Cherie Ho, Jiaye Zou, Omar Alama, Sai Mitheran Jagadesh Kumar, Benjamin Chiang, Taneesh Gupta, Chen Wang, Nikhil Keetha, Katia Sycara, and Sebastian Scherer. Map

it anywhere (mia): Empowering bird's eye view mapping using large-scale public data. *arXiv preprint arXiv:2407.08726*, 2024.

[98] Shixin Hong, Yu Liu, Zhi Li, Shaohui Li, and You He. Multi-agent collaborative perception via motion-aware robust communication network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15301–15310, 2024.

[99] Shengtuo Hu, Qi Alfred Chen, Jiwon Joung, Can Carlak, Yiheng Feng, Z Morley Mao, and Henry X Liu. Cvshield: Guarding sensor data in connected vehicle with trusted execution environment. In *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security*, pages 1–4, 2020.

[100] Xuemin Hu, Shen Li, Tingyu Huang, Bo Tang, Rouxing Huai, and Long Chen. How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence. *IEEE Transactions on Intelligent Vehicles*, 9(1):593–612, 2023.

[101] Yue Hu, Shaoheng Fang, Zixing Lei, Yiqi Zhong, and Siheng Chen. Where2comm: Communication-efficient collaborative perception via spatial confidence maps. *Advances in neural information processing systems*, 35:4874–4886, 2022.

[102] Yue Hu, Yifan Lu, Runsheng Xu, Weidi Xie, Siheng Chen, and Yanfeng Wang. Collaboration helps camera overtake lidar in 3d detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2023.

[103] Xiaoshui Huang, Guofeng Mei, Jian Zhang, and Rana Abbas. A comprehensive survey on point cloud registration. *arXiv preprint arXiv:2103.02690*, 2021.

[104] Xun Huang, Jinlong Wang, Qiming Xia, Siheng Chen, Bisheng Yang, Xin Li, Cheng Wang, and Chenglu Wen. V2x-r: Cooperative lidar-4d radar fusion with denoising diffusion for 3d object detection. In *Proceedings of the computer vision and pattern recognition conference*, pages 27390–27400, 2025.

[105] Xun Huang, Jinlong Wang, Qiming Xia, Siheng Chen, Bisheng Yang, Cheng Wang, and Chenglu Wen. V2x-r: Cooperative lidar-4d radar fusion for 3d object detection with denoising diffusion. *arXiv preprint arXiv:2411.08402*, 2, 2024.

[106] Fabian Immel, Richard Fehler, Mohammad M. Ghanaat, Florian Ries, Martin Haueis, and Christoph Stiller. Hd map generation from noisy multi-route vehicle fleet data on highways with expectation maximization. In *35th IEEE Intelligent Vehicles Symposium (IV)*, 2023.

[107] Zhou Jiang, Zhenxin Zhu, Pengfei Li, Huan-ang Gao, Tianyuan Yuan, Yongliang Shi, Hang Zhao, and Hao Zhao. P-mapnet: Far-seeing map generator enhanced by both sdmap and hdmap priors. *IEEE Robotics and Automation Letters*, 2024.

[108] Shuowei Jin, Ruiyang Zhu, Ahmad Hassan, Xiao Zhu, Xumiao Zhang, Z Morley Mao, Feng Qian, and Zhi-Li Zhang. Oasis: Collaborative neural-enhanced mobile video

streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 45–55, 2024.

[109] Anand Kashyap, Samrat Ganguly, and Samir R Das. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 242–253, 2007.

[110] Lorenzo Keller, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. Microcast: Cooperative video streaming on smartphones. In *MobiSys*. ACM, 2012.

[111] John B Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.

[112] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 399–404. IEEE, 2017.

[113] Seong-Woo Kim, Wei Liu, Marcelo H Ang, Emilio Frazzoli, and Daniela Rus. The impact of cooperative perception on decision making and planning of autonomous vehicles. *IEEE Intelligent Transportation Systems Magazine*, 7(3):39–50, 2015.

[114] Pantelis Kopelias, Elissavet Demiridi, Konstantinos Vogiatzis, Alexandros Skabardonis, and Vassiliki Zafiropoulou. Connected & autonomous vehicles–environmental impacts–a review. *Science of the total environment*, 712:135237, 2020.

[115] Dimitris E Koulouriotis and A Xanthopoulos. Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. *Applied Mathematics and Computation*, 196(2):913–922, 2008.

[116] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[117] Swarun Kumar, Shyamnath Gollakota, and Dina Katabi. A cloud-assisted design for autonomous driving. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 41–46, 2012.

[118] Swarun Kumar, Lixin Shi, Nabeel Ahmed, Stephanie Gil, Dina Katabi, and Daniela Rus. Carspeak: a content-centric network for autonomous driving. *ACM SIGCOMM Computer Communication Review*, 42(4):259–270, 2012.

[119] John Lambert and James Hays. Trust, but verify: Cross-modality fusion for hd map change detection. *arXiv preprint arXiv:2212.07312*, 2022.

[120] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[121] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[122] Seokju Lee, Francois Rameau, Fei Pan, and In So Kweon. Attentive and contrastive learning for joint depth and motion field estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4862–4871, 2021.

[123] Zixing Lei, Shunli Ren, Yue Hu, Wenjun Zhang, and Siheng Chen. Latency-aware collaborative perception. In *European Conference on Computer Vision*, pages 316–332. Springer, 2022.

[124] Chenglin Li, Hongkai Xiong, Junni Zou, and Dapeng Oliver Wu. Joint dynamic rate control and transmission scheduling for scalable video multirate multicast over wireless networks. *IEEE Transactions on Multimedia*, 20(2):361–378, 2017.

[125] Hao Li, Manabu Tsukada, Fawzi Nashashibi, and Michel Parent. Multivehicle cooperative local mapping: A methodology based on occupancy grid map merging. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2089–2100, 2014.

[126] Jinlong Li, Runsheng Xu, Xinyu Liu, Jin Ma, Zicheng Chi, Jiaqi Ma, and Hongkai Yu. Learning for vehicle-to-vehicle cooperative perception under lossy communication. *IEEE Transactions on Intelligent Vehicles*, 8(4):2650–2660, 2023.

[127] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4628–4634. IEEE, 2022.

[128] Ruibo Li, Guosheng Lin, and Lihua Xie. Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15577–15586, 2021.

[129] Yang Li, Hao Lin, Zhenhua Li, Yunhao Liu, Feng Qian, Liangyi Gong, Xianlong Xin, and Tianyin Xu. A nationwide study on cellular reliability: Measurement, analysis, and enhancements. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 597–609, 2021.

[130] Yiming Li, Dekun Ma, Ziyan An, Zixun Wang, Yiqi Zhong, Siheng Chen, and Chen Feng. V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving. *IEEE Robotics and Automation Letters*, 7(4):10914–10921, 2022.

[131] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European conference on computer vision*, pages 1–18. Springer, 2022.

[132] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: learning bird's-eye-view representation from lidar-camera

via spatiotemporal transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[133] Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. Maptr: Structured modeling and learning for online vectorized hd map construction. In *International Conference on Learning Representations*, 2023.

[134] Bencheng Liao, Shaoyu Chen, Yunchi Zhang, Bo Jiang, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Maptrv2: An end-to-end framework for online vectorized hd map construction. *International Journal of Computer Vision*, pages 1–23, 2024.

[135] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 751–766, 2018.

[136] Hansi Liu, Pengfei Ren, Shubham Jain, Mohannad Murad, Marco Gruteser, and Fan Bai. Fusioneye: Perception sharing for connected vehicles and its bandwidth-accuracy trade-offs. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2019.

[137] Jun Liu and Asad J Khattak. Delivering improved alerts, warnings, and control assistance using basic safety messages transmitted between connected vehicles. *Transportation research part C: emerging technologies*, 68:83–100, 2016.

[138] Qiang Liu, Tao Han, Jiang Linda Xie, and BaekGyu Kim. Livemap: Real-time dynamic map in automotive edge computing. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[139] Qiang Liu, Yuru Zhang, and Haoxin Wang. Edgemap: Crowdsourcing high definition map in automotive edge computing. In *IEEE International Conference on Communications (ICC)*, pages 4300–4305, 2022.

[140] Si Liu, Chen Gao, Yuan Chen, Xingyu Peng, Xianghao Kong, Kun Wang, Runsheng Xu, Wentao Jiang, Hao Xiang, Jiaqi Ma, et al. Towards vehicle-to-everything autonomous driving: A survey on collaborative perception. *arXiv preprint arXiv:2308.16714*, 2023.

[141] Yicheng Liu, Tianyuan Yuan, Yue Wang, Yilun Wang, and Hang Zhao. Vectormapnet: End-to-end vectorized hd map learning. In *International Conference on Machine Learning*, pages 22352–22369. PMLR, 2023.

[142] Yifan Lu, Yue Hu, Yiqi Zhong, Dequan Wang, Yanfeng Wang, and Siheng Chen. An extensible framework for open heterogeneous collaborative perception. *arXiv preprint arXiv:2401.13964*, 2024.

[143] Yifan Lu, Quanhao Li, Baoan Liu, Mehrdad Dianati, Chen Feng, Siheng Chen, and Yanfeng Wang. Robust collaborative 3d object detection in presence of pose errors. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4812–4818. IEEE, 2023.

[144] Guiyang Luo, Chongzhang Shao, Nan Cheng, Haibo Zhou, Hui Zhang, Quan Yuan, and Jinglin Li. Edgecooper: Network-aware cooperative lidar perception for enhanced vehicular awareness. *IEEE Journal on Selected Areas in Communications*, 2023.

[145] Hongyu Lyu, Julie Stephany Berrio Perez, Yaoqi Huang, Kunming Li, Mao Shan, and Stewart Worrall. Online high-definition map construction for autonomous vehicles: A comprehensive survey. *Journal of Sensor and Actuator Networks*, 14(1):15, 2025.

[146] Jie Mei, Kan Zheng, Long Zhao, Yong Teng, and Xianbin Wang. A latency and reliability guaranteed resource allocation scheme for lte v2v communication systems. *IEEE Transactions on Wireless Communications*, 17(6):3850–3860, 2018.

[147] Lili Miao, John Jethro Virtusio, and Kai-Lung Hua. Pc5-based cellular-v2x evolution and deployment. *Sensors*, 21(3):843, 2021.

[148] David L Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, 39(10):1482–1493, 1991.

[149] Ruoshen Mo, Bo Wu, Zhaowei Tan, and Hang Qiu. See-v2x: C-v2x direct communication dataset: An application-centric approach. In *Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems*, pages 305–311, 2025.

[150] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010.

[151] Wassim G Najm, Raja Ranganathan, Gowrishankar Srinivasan, John D Smith, Samuel Toma, Elizabeth D Swanson, August Burgett, et al. Description of light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications. Technical report, United States. Department of Transportation. National Highway Traffic Safety Administration, 2013.

[152] Arvind Narayanan, Eman Ramadan, Rishabh Mehta, Xinyue Hu, Qingxu Liu, Rostand AK Fezeu, Udhaya Kumar Dayalan, Saurabh Verma, Peiqi Ji, Tao Li, et al. Lumos5g: Mapping and predicting commercial mmwave 5g throughput. In *Proceedings of the ACM Internet Measurement Conference*, pages 176–193, 2020.

[153] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, et al. A variegated look at 5g in the wild: performance, power, and qoe implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 610–625, 2021.

[154] Mohammad Naserian, Fan Bai, Xumiao Zhang, Ruiyang Zhu, Qingzhao Zhang, Xiao Zhu, and Zhuoqing Morley Mao. Cooperative v2x sensor sharing, December 21 2023. US Patent App. 17/844,978.

[155] Yuanzhi Ni, Jianping He, Lin Cai, and Yuming Bo. Data uploading in hybrid v2v/v2i vehicular networks: Modeling and cooperative strategy. *IEEE Transactions on Vehicular Technology*, 67(5):4602–4614, 2018.

[156] Zhenyang Ni, Zixing Lei, Yifan Lu, Dingju Wang, Chen Feng, Yanfeng Wang, and Siheng Chen. Self-localized collaborative perception. *arXiv preprint arXiv:2406.12712*, 2024.

[157] Cătălin Nicutar, Dragoş Niculescu, and Costin Raiciu. Using cooperation for low power low latency cellular connectivity. In *CoNEXT*, pages 337–348. ACM, 2014.

[158] Jitendra Padhye, Sharad Agarwal, Venkata N Padmanabhan, Lili Qiu, Ananth Rao, and Brian Zill. Estimation of link interference in static multi-hop wireless networks. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pages 28–28, 2005.

[159] Anshul Paigwar, Özgür Erkent, David Sierra-Gonzalez, and Christian Laugier. Gndnet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2150–2156. IEEE, 2020.

[160] Yosub Park, Taehyung Kim, and Daesik Hong. Resource size control for reliability improvement in cellular-based v2v communication. *IEEE Transactions on Vehicular Technology*, 68(1):379–392, 2018.

[161] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[162] Jon Postel. User datagram protocol. Technical report, 1980.

[163] Hang Qiu, Fawad Ahmad, Fan Bai, Marco Gruteser, and Ramesh Govindan. Avr: Augmented vehicular reality. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 81–95. ACM, 2018.

[164] Hang Qiu, Pohan Huang, Namo Asavisanu, Xiaochen Liu, Konstantinos Psounis, and Ramesh Govindan. Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving. *arXiv preprint arXiv:2112.14947*, 2021.

[165] Alexander Rabitsch, Karl-Johan Grinnemo, Anna Brunstrom, Henrik Abrahamsson, Fehmi Ben Abdesslem, Stefan Alfredsson, and Bengt Ahlgren. Utilizing multiconnectivity to reduce latency and enhance availability for vehicle to infrastructure communication. *IEEE Transactions on Mobile Computing*, 21(5):1874–1891, 2020.

[166] Andreas Rauch, Felix Klanner, and Klaus Dietmayer. Analysis of v2x communication parameters for the development of a fusion architecture for cooperative perception systems. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 685–690. IEEE, 2011.

[167] Zaydoun Yahya Rawashdeh and Zheng Wang. Collaborative automated driving: A machine learning-based method to enhance the accuracy of shared information. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3961–3966. IEEE, 2018.

[168] Frédéric Rivard, Jonathan Bisson, François Michaud, and Dominic Létourneau. Ultrasonic relative positioning for multi-robot systems. In *2008 IEEE International Conference on Robotics and Automation*, pages 323–328. IEEE, 2008.

[169] Herbert Robbins. Some aspects of the sequential design of experiments. 1952.

[170] T-YLPG Ross and GKHP Dollár. Focal loss for dense object detection. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2980–2988, 2017.

[171] Xin Xia Xu Han Jinlong Li Jiaqi Ma Runsheng Xu, Hao Xiang. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[172] Mohamed Sayed, Stepan Perminov, and Dzmitry Tsetserukou. Polymerge: A novel technique aimed at dynamic hd map updates leveraging polylines. In *2023 21st International Conference on Advanced Robotics (ICAR)*, pages 94–99. IEEE, 2023.

[173] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[174] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5135–5142. IEEE, 2020.

[175] Vishal Sharma, Harsukhpreet Singh, and M Kaur. Implementation and analysis of ofdm based ieee 802.11 g vanet. *International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC), ISSN*, pages 2250–1568, 2013.

[176] Anqi Shi, Yuze Cai, Xiangyu Chen, Jian Pu, Zeyu Fu, and Hong Lu. Globalmapnet: An online framework for vectorized global hd map construction. *arXiv preprint arXiv:2409.10063*, 2024.

[177] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[178] Shuyao Shi, Jiahe Cui, Zhehao Jiang, Zhenyu Yan, Guoliang Xing, Jianwei Niu, and Zhenchao Ouyang. Vips: Real-time perception fusion for infrastructure-assisted autonomous driving. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 133–146, 2022.

[179] Zhiying Song, Fuxi Wen, Hailiang Zhang, and Jun Li. An efficient and robust object-level cooperative perception framework for connected and automated driving. *arXiv preprint arXiv:2210.06289*, 2022.

[180] Mariusz Specht. Consistency analysis of global positioning system position errors with typical statistical distributions. *The Journal of Navigation*, 74(6):1201–1218, 2021.

[181] Jan Erik Stellet, Fabian Straub, Jan Schumacher, Wolfgang Branz, and J Marius Zöllner. Estimating the process noise variance for vehicle motion models. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 1512–1519. IEEE, 2015.

[182] Alexis Stoven-Dubois, Kuntima Kiala Miguel, Aziz Dziri, Bertrand Leroy, and Roland Chapuis. A collaborative framework for high-definition mapping. *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[183] Rémy Sun, Li Yang, Diane Lingrand, and Frédéric Precioso. Mind the map! accounting for existing map information when estimating online hdmaps from sensor. *arXiv preprint arXiv:2311.10517*, 2023.

[184] Sanjib Sur, Xinyu Zhang, Parmesh Ramanathan, and Ranveer Chandra. {BeamSpy}: Enabling robust 60 {GHz} links under blockage. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, pages 193–206, 2016.

[185] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[186] Muhammad Naeem Tahir, Pekka Leviäkangas, and Marcos Katz. Connected vehicles: V2v and v2i road weather and traffic communication using cellular technologies. *Sensors*, 22(3):1142, 2022.

[187] C-K Toh, Minar Delwar, and Donald Allen. Evaluating the communication performance of an ad hoc wireless network. *IEEE Transactions on Wireless communications*, 1(3):402–414, 2002.

[188] James Tu, Tsunhsuan Wang, Jingkang Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Adversarial attacks on multi-agent communication. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7768–7777, 2021.

[189] Nicholas Vadivelu, Mengye Ren, James Tu, Jingkang Wang, and Raquel Urtasun. Learning to communicate and correct pose errors. In *Conference on Robot Learning*, pages 1195–1210. PMLR, 2021.

[190] Rens Wouter van der Heijden, Stefan Dietzel, Tim Leinmüller, and Frank Kargl. Survey on misbehavior detection in cooperative intelligent transportation systems. *IEEE Communications Surveys & Tutorials*, 21(1):779–811, 2018.

[191] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[192] Anna Maria Vegni and Thomas DC Little. Hybrid vehicular communications based on v2v-v2i protocol switching. *International Journal of Vehicle Information and Communication Systems*, 2(3-4):213–231, 2011.

[193] Vladimir Vukadinovic, Krzysztof Bakowski, Patrick Marsch, Ian Dexter Garcia, Hua Xu, Michal Sybis, Pawel Sroka, Krzysztof Wesolowski, David Lister, and Ilaria Thibault. 3gpp c-v2x and ieee 802.11 p for vehicle-to-vehicle communications in highway platooning scenarios. *Ad Hoc Networks*, 74:17–29, 2018.

[194] Dian Vektorendra Wahyuda, Dedy Achmadi, Riri Fitri Sari, et al. Comparison of different wlan standard on propagation performance in v2v named data networking. In *2017 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*, pages 128–133. IEEE, 2017.

[195] Qing Wang, Pingyi Fan, and Khaled Ben Letaief. On the joint v2i and v2v scheduling for cooperative vanets with network coding. *IEEE Transactions on Vehicular Technology*, 61(1):62–73, 2011.

[196] Shuo Wang, Fan Jia, Weixin Mao, Yingfei Liu, Yucheng Zhao, Zehui Chen, Tiancai Wang, Chi Zhang, Xiangyu Zhang, and Feng Zhao. Stream query denoising for vectorized hd-map construction. In *European Conference on Computer Vision*, pages 203–220. Springer, 2024.

[197] Tianhang Wang, Guang Chen, Kai Chen, Zhengfa Liu, Bo Zhang, Alois Knoll, and Changjun Jiang. Umc: A unified bandwidth-efficient and multi-resolution based collaborative perception framework. *arXiv preprint arXiv:2303.12400*, 2023.

[198] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *European Conference on Computer Vision*, pages 605–621. Springer, 2020.

[199] Christine Ware. The osi network layer: Standards to cope with the real world. *Proceedings of the IEEE*, 71(12):1384–1387, 1983.

[200] Dehui Wei, Jiao Zhang, Haozhe Li, Zhichen Xue, Yajie Peng, and Rui Han. Multipath smart preloading algorithms in short video peer-to-peer cdn transmission architecture. *IEEE Network*, 2023.

[201] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[202] Lena Wild, Ludvig Ericson, Rafael Valencia, and Patric Jensfelt. Exelmap: Explainable element-based hd-map change detection and update. *arXiv preprint arXiv:2409.10178*, 2024.

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

[204] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[205] Fei Wu, Yang Yang, Ouyang Zhang, Kannan Srinivasan, and Ness B Shroff. Anonymous-query based rate control for wireless multicast: Approaching optimality with constant feedback. In *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 191–200, 2016.

[206] Deguo Xia, Weiming Zhang, Xiyan Liu, Wei Zhang, Chenting Gong, Xiao Tan, Jizhou Huang, Mengmeng Yang, and Diange Yang. Ldmapnet-u: An end-to-end system for city-scale lane-level map updating. *arXiv preprint arXiv:2501.02763*, 2025.

[207] Hao Xiang, Zhaoliang Zheng, Xin Xia, Seth Z Zhao, Letian Gao, Zewei Zhou, Tianhui Cai, Yun Zhang, and Jiaqi Ma. V2x-realo: An open online framework and dataset for cooperative perception in reality. *arXiv preprint arXiv:2503.10034*, 2025.

[208] Runsheng Xu, Zhengzhong Tu, Hao Xiang, Wei Shao, Bolei Zhou, and Jiaqi Ma. Cobevt: Cooperative bird's eye view semantic segmentation with sparse transformers. *arXiv preprint arXiv:2207.02202*, 2022.

[209] Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. V2x-vit: Vehicle-to-everything cooperative perception with vision transformer. In *European conference on computer vision*, pages 107–124. Springer, 2022.

[210] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2583–2589. IEEE, 2022.

[211] Zhigang Xu, Xiaochi Li, Xiangmo Zhao, Michael H Zhang, and Zhongren Wang. Dsrc versus 4g-lte for connected vehicle applications: A study on field experiments of vehicular communication performance. *Journal of advanced transportation*, 2017, 2017.

[212] Reda Yacouby and Dustin Axman. Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In *Proceedings of the first workshop on evaluation and comparison of NLP systems*, pages 79–91, 2020.

[213] Fanny Yang, Sivaraman Balakrishnan, and Martin J Wainwright. Statistical and computational guarantees for the baum-welch algorithm. *Journal of Machine Learning Research*, 18(125):1–53, 2017.

[214] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2020.

[215] Jing Yang, Sen Yang, Xiao Tan, and Hanli Wang. Histrackmap: Global vectorized high-definition map construction via history map tracking. *arXiv preprint arXiv:2503.07168*, 2025.

[216] Kun et al. Yang. Spatio-temporal domain awareness for multi-agent collaborative perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23383–23392, 2023.

[217] Jingang Yi, Hongpeng Wang, Junjie Zhang, Dezhen Song, Suhada Jayasuriya, and Jingtai Liu. Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation. *IEEE transactions on robotics*, 25(5):1087–1097, 2009.

[218] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[219] Haibao Yu, Yizhen Luo, Mao Shu, Yiyi Huo, Zebang Yang, Yifeng Shi, Zhenglong Guo, Hanyu Li, Xing Hu, Jirui Yuan, et al. Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21361–21370, 2022.

[220] Jingyi Yu, Zizhao Zhang, Shengfu Xia, and Jizhang Sang. Scalablemap: Scalable map learning for online long-range vectorized hd map construction. *arXiv preprint arXiv:2310.13378*, 2023.

[221] Tianyuan Yuan, Yicheng Liu, Yue Wang, Yilun Wang, and Hang Zhao. Streammapnet: Streaming mapping network for vectorized online hd map construction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 7356–7365, 2024.

[222] Yunshuang Yuan, Hao Cheng, and Monika Sester. Keypoints-based deep feature fusion for cooperative vehicle detection of autonomous driving. *IEEE Robotics and Automation Letters*, 7(2):3054–3061, 2022.

[223] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5067–5073. IEEE, 2017.

[224] Hongqiang Zhai and Yuguang Fang. Physical carrier sensing and spatial reuse in multirate and multihop wireless ad hoc networks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12. Citeseer, 2006.

[225] Jingyu Zhang, Kun Yang, Yilei Wang, Hanqi Wang, Peng Sun, and Liang Song. Ermvp: Communication-efficient and collaboration-robust multi-vehicle perception in challenging environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12575–12584, 2024.

[226] Qingzhao Zhang, Shuowei Jin, Ruiyang Zhu, Jiachen Sun, Xumiao Zhang, Qi Alfred Chen, and Z Morley Mao. On data fabrication in collaborative vehicular perception: Attacks and countermeasures. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 6309–6326, 2024.

[227] Qingzhao Zhang, Xumiao Zhang, Ruiyang Zhu, Fan Bai, Mohammad Naserian, and Z Morley Mao. Robust real-time multi-vehicle collaboration on asynchronous sensors. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.

[228] Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y. Ethan Guo, Feng Qian, and Z. Morley Mao. Emp: edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 545–558. Association for Computing Machinery, 2021.

[229] Zixu Zhang and Jaime F Fisac. Safe occlusion-aware autonomous driving via game-theoretic active perception. *arXiv preprint arXiv:2105.08169*, 2021.

[230] Binyu Zhao, Wei Zhang, and Zhaonian Zou. Bm2cp: Efficient collaborative perception with lidar-camera modalities. *arXiv preprint arXiv:2310.14702*, 2023.

[231] Pengyuan Zhou, Wenxiao Zhang, Tristan Braud, Pan Hui, and Jussi Kangasharju. Arve: Augmented reality applications in vehicle to edge networks. In *Proceedings of the 2018 Workshop on Mobile Edge Communications*, pages 25–30, 2018.

[232] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[233] Zewei Zhou, Hao Xiang, Zhaoliang Zheng, Seth Z. Zhao, Mingyue Lei, Yun Zhang, Tianhui Cai, Xinyi Liu, Johnson Liu, Maheswari Bajji, Jacob Pham, Xin Xia, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. V2xpnp: Vehicle-to-everything spatio-temporal fusion for multi-agent perception and prediction. *arXiv preprint arXiv:2412.01812*, 2024.

[234] Jiangtong Zhu, Zhao Yang, Yinan Shi, Jianwu Fang, and Jianru Xue. Ic-mapper: Instance-centric spatio-temporal modeling for online vectorized map construction. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 9961–9969, 2024.

[235] Ruiyang Zhu, Minkyoung Cho, Shuqing Zeng, Fan Bai, Xiang Gao, and Z Morley Mao. Scalable crowd-sourced global hd map construction via collaborative map perception and sparse graph fusion. In *The 4th Workshop on Transformers for Vision (T4V) at CVPR 2025*, 2025.

[236] Ruiyang Zhu, Minkyoung Cho, Shuqing Zeng, Fan Bai, and Z Morley Mao. Scorpion: Robust spatial-temporal collaborative perception model on lossy wireless network. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 14194–14201. IEEE, 2025.

[237] Ruiyang Zhu, Xiao Zhu, Anlan Zhang, Xumiao Zhang, Jiachen Sun, Feng Qian, Hang Qiu, Z Morley Mao, and Myungjin Lee. Boosting collaborative vehicular perception on the edge with vehicle-to-vehicle communication. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, pages 141–154, 2024.

[238] Xiao Zhu, Jiachen Sun, Xumiao Zhang, Y Ethan Guo, Feng Qian, and Z Morley Mao. Mpbond: efficient network-level collaboration among personal mobile devices. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 364–376, 2020.

[239] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[240] Zhe Zhu and Curtis E Woodcock. Continuous change detection and classification of land cover using all available landsat data. *Remote sensing of Environment*, 144:152–171, 2014.

[241] Zhengxia Zou, Rusheng Zhang, Shengyin Shen, Gaurav Pandey, Punarjay Chakravarty, Armin Parchami, and Henry X Liu. Real-time full-stack traffic scene perception for autonomous driving with roadside cameras. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 890–896. IEEE, 2022.