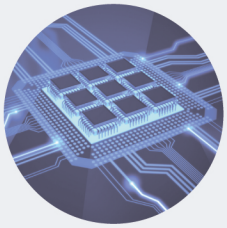
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE SISTEMAS COMPUTACIONAIS CRÍTICOS



Co-simulation of vehicle distributed perception for road safety applications

TIAGO FILIPE LONGO COSTA
outubro de 2025

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Co-simulation of vehicle distributed perception for road safety applications

Tiago Filipe Longo Costa

Dissertation submitted in partial fulfilment of the requirements for the Master's Degree in Critical Computing Systems Engineering

Supervisor: Ricardo Augusto Rodrigues Da Silva Severino

Evaluation Committee:
President:
Luis Lino Ferreira, Instituto Superior de Engenharia do Porto
Members:
António Barros, Instituto Superior de Engenharia do Porto
Ricardo Severino, Instituto Superior de Engenharia do Porto

Porto, October 9, 2025

Statement of Integrity

I hereby declare having conducted this academic work with integrity. I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

I declare that the work presented in this document is original and my own, and has not previously been used for any other purpose.

I further declare that I have fully followed the Code of Good Practices and Conduct of the Polytechnic Institute of Porto.

ISEP, Porto, October 6, 2025

Dedicatory

To Professor Ricardo, for his valuable guidance, and continuous support throughout the development of this work.

To Luís Ribeiro, for all the support and availability.

To my mother and Vitor, for their unwavering support, encouragement, and belief in me.

Abstract

The fast evolution of autonomous vehicles has introduced many new challenges, especially in perception, communication reliability, and energy efficiency. Traditional autonomous vehicles rely on onboard sensors, which are traditionally limited by range, computational cost, and occlusions. The goal of this thesis is to explore the integration of distributed perception systems to enhance situational awareness and improve autonomous vehicles decision-making. This work introduces an extension of the previously established co-simulation framework of Oliveira 2023 and Ribeiro 2024. The solution expands the capabilities of autonomous vehicles to sense beyond the onboard sensors while facilitating safe and efficient merging under different traffic conditions using roadside cameras and a roadside unit. The framework incorporates YOLO-based object detection with trilateration to combine data from a multi-camera setup for improved localization accuracy and detection robustness across various traffic conditions. The experimental results show that compared to single-camera configurations, multi-camera fusion significantly increased recall and improved localization levels in conditions with different traffic and perception conditions. The roadside unit can also identify gaps for a safe and efficient merging and provide a target speed to the ego-vehicle via a simulated V2X communication. Overall, the proposed framework demonstrates the feasibility of infrastructure-assisted cooperative perception, providing a realistic and extensible testbed for future research in distributed perception, sensor fusion, and V2X communication for road safety applications.

Keywords: autonomous vehicle, cooperative perception, distributed perception, highway merging

Resumo

A rápida evolução dos veículos autónomos tem introduzido diversos desafios, especialmente ao nível da perceção, da fiabilidade das comunicações e da eficiência energética. Os veículos autónomos tradicionais dependem de sensores embarcados, os quais são normalmente limitados em alcance, capacidade computacional e oclusões. A abordagem proposta é explorar a integração de sistemas de perceção distribuída para melhorar a consciência situacional e apoiar a tomada de decisão dos veículos autónomos. Este trabalho apresenta uma extensão da *framework* de co-simulação previamente desenvolvida por Oliveira 2023 e Ribeiro 2024. A solução proposta expande as capacidades dos veículos autónomos, permitindo-lhes detetar além dos sensores embarcados, ao mesmo tempo que facilita manobras de mudança de via seguras e eficientes em diferentes condições de tráfego, através do uso de câmaras e de uma unidade rodoviária de processamento. A solução desenvolvida integra a detecção de objetos baseada em YOLO com trilateração, combinando dados provenientes de um conjunto de múltiplas câmeras, de modo a melhorar a precisão de localização e a robustez da detecção em várias condições de tráfego. Os resultados experimentais demonstram que, em comparação com configurações de câmara única, a fusão multicâmara aumentou significativamente a deteção e melhorou os níveis de localização sob diferentes cenários de perceção e tráfego. A unidade rodoviária é ainda capaz de identificar intervalos seguros para manobras de mudança de via e de fornecer uma velocidade alvo ao veículo através de uma comunicação V2X simulada. De forma geral, a solução proposta demonstra a viabilidade da percepção cooperativa assistida por infraestrutura, fornecendo uma plataforma realista e extensível para futuras investigações em percepção distribuída, fusão sensorial e comunicação V2X aplicadas à segurança rodoviária.

Contents

List of Figures

List of Acronyms

AI	Artificial Intelligence.
AV	Autonomous Vehicles.
C-V2X	Cellular V2X.
CAV	Connected and Autonomous Vehicles.
CNN	Convolutional Neural Network.
CPM	Cooperative Perception Messages.
ETSI	European Telecommunications Standards Institute.
IoT	Internet of Things.
QoS	Quality of Service.
ROS	Robot Operating System.
RSU	Roadside Unit.
V2I	Vehicle-to-Infrastructure.
V2P	Vehicle-to-Pedestrian.
V2V	Vehicle-to-Vehicle.
V2X	Vehicle-to-Everything.
YOLO	You Only Look Once.

Chapter 1

Introduction

1.1 Problem description

The rapid advancements in Autonomous Vehicles (AV) have expanded their operational abilities to include sophisticated features, capable of reshaping transport systems, improving road safety, and increasing traffic efficiency (Achenef et al. 2024). However, as these systems advance toward higher levels of automation, they face complex challenges, opening new opportunities and problems in the field. Current AVs are almost completely ego-vehicle focused, relying on onboard sensors and Artificial Intelligence (AI) to perceive and react to the world around them.

Although this approach has been successful in more controlled environments, this ego-vehicle-centric approach still has multiple limitations. First, onboard sensors are limited by their physical range and field of view, creating blind spots that could compromise safety in dynamic scenarios. Second, the use of high-resolution sensors in real time has considerable commutation overhead for sensor data processing, as well as substantial power consumption. Third, environmental conditions like weather, occlusions, and complex urban geometries can adversely affect the performance of sensors and potentially lead to perception failures in critical situations (Ullah et al. 2025).

The mentioned limitations are particularly aggravated in situations involving high-speed and high-risk situations like highway merging, where vehicles effectively have the need to establish situational awareness in split seconds, relying on limited perception and information of the surrounding traffic. This maneuver is agreed to be one of the most elaborated tasks in autonomous driving. The challenge comes from the demands of determining the position, velocities, and movements of the surrounding vehicles. In these instances, a failure in perception could lead to severe outcomes depending on the involved context of the vehicles, such as collisions, traffic delays, and a decrease in public acceptance of advancing technologies in autonomous vehicles.

In response to these limitations, researchers and the industry are increasingly exploring distributed and cooperative perception solutions. These systems address the challenge of understanding the environment by improving the potential range of information that a vehicle may observe (Huang et al. 2025). Instead of relying on a single vehicle sensors, these alternatives use coverage from sensors on infrastructure, and nearby vehicles. This transition from an ego-vehicle standpoint to a hybrid sensing method that includes the environment, represents a shift in the sensing paradigm, by relying on lower-cost embedded devices that in combination with distributed sensing, can provide broad coverage, increasing the safety and energy efficiency without each vehicle being required to increase the cost or energy consumption.

1.2 Research Objectives

The aim of this thesis is to extend the distributed perception capabilities of a co-simulation framework previously developed by Oliveira 2023 and Ribeiro 2024, with a special focus on supporting cooperative autonomous actions in highway merging scenarios. The main objective is to design, implement, and validate a system that leverages data from various infrastructure sensor sources using wireless communication protocols, to build a more reliable and efficient perception system in dynamic environments.

The research begins with a general review of the state-of-the-art of distributed and cooperative perception approaches, and Vehicle-to-Everything (V2X) communication technologies, with an emphasis on approaches that can meet the desired real-time and scalability requirements. Building upon this foundation, a modular system architecture is proposed that integrates Robot Operating System (ROS) 2 as the backbone of the system, Gazebo for realistic environment and simulation, and OMNeT++ for network modeling. This architecture is defined to address the challenges related to sensor integration, low-latency data fusion, and adaptivity to a variety of communication conditions.

Algorithmically, the system carries out multi-camera vehicle detection via You Only Look Once (YOLO) based models, trilateration-based localization, and distributed sensor fusion mechanisms meant to operate under realistic traffic and communication conditions. The co-simulation environment is enhanced with the ability to evaluate perception, communication, and control, which enables a more detailed assessment of detection and localization accuracy, communication latency, and system robustness. The contribution of this work includes the design of a scalable cooperative perception system architecture, and a demonstration of performance improvements over ego-vehicle perception in controlled yet realistic simulation scenarios.

This research represents a contribution in the development of distributed perception systems for autonomous vehicles, with particular emphasis on practical implementation and validation through advanced co-simulation techniques.

1.3 Research Contributions

The main contributions of this thesis are as follows:

- A detailed review of the fundamentals of cooperative perception, distributed perception, computer vision, and sensor fusion.
- Review state-of-the-art simulation tools, object detection algorithms and communication technologies.
- Design of a scalable and modular system architecture that integrates distributed perception with wireless communication protocols to enable cooperative autonomous vehicle maneuvers.
- Extend the existing co-simulation framework by implementing enhanced distributed perception capabilities, wireless communication, and control evaluation.
- Experimental validation of the proposed system in simulated highway merging scenarios, evaluation of performance and exploration of the impact of different metrics and scenario settings on QoS.

1.4 Thesis Organization

The remainder of this thesis is organized to provide a logical progression from theoretical foundations to practical implementation and evaluation. Chapter 2 reviews the relevant literature, covering the fundamental concepts and related studies that form the basis for this research. Chapter 3 presents the technologies and tools used in the development of the proposed system, detailing their role and integration. Chapter 4 outlines the system architecture and overall design, including the selected use-case scenario and its requirements. Chapter 5 explains the implementation process, discussing the developed components and the integration of the different modules. Chapter 6 presents the experimental evaluation, detailing the methodology, scenarios, and performance results. Finally, Chapter 7 summarizes the main findings, reflects on limitations, and outlines potential future research directions.

Chapter 2

State of the Art

2.1 Autonomous Driving

Autonomous driving is the ability of a vehicle to perceive surroundings, navigate, and make decisions without direct human intervention. At its essence, it involves vehicles equipped with sensors (e.g. LiDAR, radar, cameras), computing units, control systems, and actuators that together replace or complement human drivers in the driving task. (Padmaja 2023).

The internationally recognized standard for classifying driving automation is given by the Society of Automotive Engineers (*SAE* 2021). Which defines six levels of autonomous driving systems based on driver intervention and automation:

- Level 0 (No Automation): The driver controls all functions, no automation is involved.
- Level 1 (Driver Assistance): Automation is limited to specific tasks like adaptive cruise control or lane keeping. The driver remains responsible for the driving environment.
- Level 2 (Partial Automation): Combines automation functions (control steering and speed), but still requires driver oversight.
- Level 3 (Conditional Automation): The system controls all functions of driving under certain scenarios (e.g. highway) but expects human intervention upon request.
- Level 4 (High Automation): No human intervention is required in defined scenarios or geofenced areas.
- Level 5 (Full Automation): The vehicle operates autonomously in all scenarios, regardless of environment or conditions.

The development of autonomous vehicle technologies has significantly advanced from a visionary concept to an established area of research due to the embedded advances in sensor, real-time perception, and artificial intelligence. Current systems incorporate heterogeneous modalities through deep-learning based fusion, incorporating bird's eye view representations and transformer/Convolutional Neural Network (CNN) architectures to detect, classify, and track dynamic agents in high-speed, urban and adverse weather environments (Y. Wang et al. 2023). These capabilities promise more than individual mobility benefits, they include reduced traffic congestion, improved energy efficiency, and reduced environmental impact. This is becoming increasingly apparent in automated production vehicles with more sophisticated automated capabilities in lane changing, urban navigating, and emergency braking.

In spite of the progress made, technical, regulatory, and societal barriers to large-scale deployment will continue to exist. From a technical perspective, the challenges of rare-event coverage, distributional shift, and computational limits for real-time multi-sensor inference

are still very active, the problem of managing edge cases and long-tail scenarios remains open. From a regulatory perspective, the safety validation, liability, and operational design categories are often seen as fragmented across jurisdictional boundaries, stalling harmonized deployment. Public acceptance surveys continue to report significant hesitance associated with safety perception, level of trust, and many surveys suggest that a considerable proportion of drivers remain resistant to fully automated driving without additional guarantees. Engaging in a strategy to realize the full promise of autonomous driving will ultimately require coordinated advancement on three fronts: technical robustness and verification, regulatory standardization and certification pathway, and sustained public engagement to build trust and legitimacy for autonomous driving implementation (Widen and Koopman 2022).

To operate safely through increasing levels of automation, an autonomous vehicle must incorporate multiple tightly coupled subsystems: perception to sense and interpret the scene, localization to estimate ego-vehicle pose on a map, prediction to expect other agent intentions, trajectories planning to determine an appropriate feasible, safe, and efficient maneuver, and control to carry out the planned trajectory through actuation with respect to vehicle dynamics (X. Wang et al. 2024). Perception serves as the upstream dependency for all downstream functions. Without reliable detection, classification, and tracking of static and dynamic objects, it is impossible to trust localization alignment, intent prediction, motion planning, or subsequent control. In addition, the most critical failure modes evident in complex, real-world deployments often stem from degrees of sensing failures (ranges, occlusions, adverse weather, and computational limits), propagating uncertainty into prediction and planning. Therefore, the emphasis of work is on perception, where the various sensors, processing frameworks, and approaches to detecting, classifying, and tracking agents in both static and dynamic environments will be traced and explored. In particular, cooperative and distributed perception through V2X, as a solution to overcome the intrinsic limits of vehicle-centric centered sensing.

2.2 Perception

Perception is the backbone of autonomous driving, enabling vehicles to interpret, comprehend, and interact with the environment. This ability is crucial for vehicles to achieve true autonomy, because it provides the information required for decision making, navigation, and vehicle control in dynamic scenarios. In practice, these systems acquire, process and interpret real-time sensor data to construct a situational awareness identifying and tracking both static (e.g. lanes, road boundaries, traffic signs) and dynamic (e.g pedestrians, cyclists, vehicles) elements (Rosique et al. 2019).

As advancements are made towards higher levels of automation in vehicles, demand for complexity and reliability from this onboard perception systems has increased. Safety-critical functions must be reliable in any situation, including bad weather, dense traffic, and unstructured urban scenes. However, vehicle-centric perception is limited due to sensors limited range, line-of-sight occlusions, and the sensors inability to effectively function under every modality and computing resources needed in real-time. The structural limitations of vehicle-centric sensing motivates the research for cooperative and distributed perception paradigms, where these can extend visibility through occlusions and range, increase detection and tracking accuracy, and consequently improve decision making in complicated spaces, such as race conditions at intersections, lane changes, and merges.

2.2.1 Cooperative perception

Cooperative sensing is a paradigm shift from an egocentric traditional approach to a cooperative one where multiple actors share data in order to obtain an enhanced situation awareness, beyond the individual sensors capacity. This structure enables vehicles to exchange sensor data and perception information with adjacent vehicles and roadside infrastructure through V2X communication technologies (figure 2.1), enhancing the perception range as well as detection precision and mitigating single-vehicle perception system limitations (Han et al. 2023).

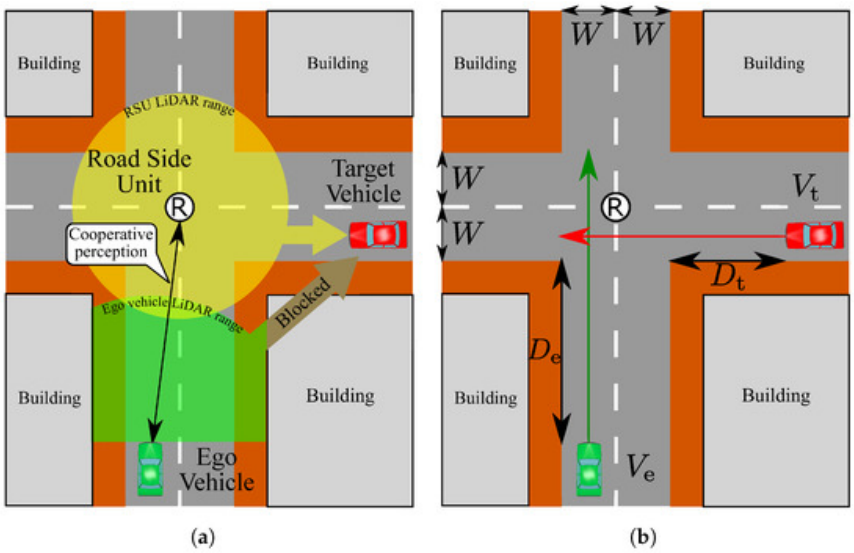


Figure 2.1: Example scenario of cooperative perception on a intersection.

Therefore, this model addresses some of the challenges of autonomous driving, such as occlusion, limited sensor range, and environmental uncertainty. The AV perception in its traditional concept is very much dependent on onboard sensors such as cameras, LiDARs, and radars to construct a detailed environmental model. This vehicle-centric approach works well in most cases but falls short in dense and dynamic scenarios where a cooperative perception system can offer much more precise understanding of the environment by sharing and fusing data from multiple sensors among different vehicles and infrastructure. In Caillot et al. 2022 and Han et al. 2023 the authors explore and outline the different architectures and challenges that are encountered by these types of systems.

System architecture of cooperative perception systems usually utilizes three different fusion methods that have different trade-offs in information completeness, communication overhead, and computation load. Early fusion (data-level fusion) involves sharing raw sensor data prior to processing, retaining maximum information at the expense of large communication bandwidth and rigid temporal synchronization among participants. Intermediate fusion (feature-level fusion) is the widely used method, where cars exchange preprocessed features extracted from sensor data, achieving an information richness, communication efficiency, and the advantage of lower latency through decentralized processing. Late fusion (decision-level fusion) involves exchanging higher-level perception results such as lists of detected objects and classifications, where the communication overhead is minimal at the cost of particular environmental information.

The effectiveness of cooperative perception depends on secure communication technologies to enable real-time data exchange between vehicles and infrastructure. Three prominent communication standards are at the forefront:

Dedicated Short-Range Communication (DSRC), based on the IEEE 802.11p standard, provides low-latency Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication over a range of a maximum of 1000 meters but degrades performance in high-density traffic due to channel clogging.

Cellular V2X (C-V2X) uses cellular network infrastructure to provide improved reliability and scalability, particularly with 5G technology providing support for both network-based and direct communication modes, though with potential latency problems in hard safety use cases.

The European Telecommunications Standards Institute (ETSI) has standardized cooperative perception communication. ETSI ITS-G5 specifies the use of IEEE 802.11p in the automotive environment in Europe, using the 5.9 GHz frequency band, as well as standardization of Cooperative Perception Messages (CPM), which establish standardized formats and triggering conditions to exchange perception data. CPM describe information about the objects being perceived, like position, velocity, and size, and are used to exchange perception data when there are variations in the environment (e.g.vehicles and vulnerable road users becoming visible, or changing velocity).

There are, nonetheless, challenges for cooperative perception deployment (Caillot et al. 2022). Latency and packet loss in wireless communication can impair the system performance, especially for safety-critical applications requiring low latency. Localization errors by engaged vehicles can also cause misalignment of perception data fusion, thus considerably lowering the accuracy of the entire system. Heterogeneous sensor fusion requires higher-level algorithms for fusing different sensor data with different resolutions, refresh rates, and data representations. And finally, the issue of security and privacy for the over-the-air transmission of sensitive vehicle and infrastructure data leads to the need for high standards of cybersecurity.

Recent research in cooperative perception focused on integrating edge computing to reduce communication latency and improve real-time performance. Multi-Access Edge Computing (MEC) architectures position computation closer to vehicles, enabling faster data processing and fusion and reducing dependence on distant cloud infrastructure. Additional studies also explore privacy preserving methods and collaborative intelligence frameworks, trading off the benefits of shared data against security and privacy requirements.

2.2.2 Distributed Perception

Distributed perception extends the traditional cooperative approach by leveraging widespread sensing infrastructure and edge computing to build end-to-end perception networks supporting autonomous vehicle operation over wider geographies (Tsukada et al. 2020). Distributed perception can potentially address modern autonomous vehicle computational challenges, which may require the equivalent of thousands of conventional servers to compute the advanced perception needs to drive safely within urban environments. As opposed to centralized cloud paradigms of computing that are expensive in terms of data transmission latency and potential security exposure, distributed perception systems position computing resources at strategic geographic locations to minimize processing latency while maintaining data locality restrictions.

The distributed perception systems typically consist of multiple layers of sensing and processing infrastructure. RSUs consolidate sensors such as LiDAR, cameras, and radars to identify and monitor traffic participants including vehicles, pedestrians, and bicycles in specific areas of coverage. These units communicate with vehicles via V2X protocols while interconnecting with broader network infrastructure to enable cooperative perception across extended road segments. Edge computing nodes at cellular base sites or dedicated roadside equipment provide intermediate processing capability between isolated vehicles and centralized cloud facilities to enable real-time fusion and analysis for multiple vehicles in parallel. The most significant advantages of distributed perception are high availability through elimination of single points of failure by having sensing and processing distributed on numerous nodes, scalability enabling dynamic resource allocation according to traffic density and computational requirements, and data locality minimizing latency by processing data near its source of generation. High-end AI models like deep learning-based CNN and transformer structures, are crucial to handle the vast quantities of real-time multi-modal sensor data generated by distributed sensing networks.

Distributed perception systems also face a range of technical issues that research continues to address. Sensor fusion complexity increases exponentially when integration is of data from heterogeneous sensors placed in diverse physical spaces, requiring sophisticated algorithms to maintain spatial and temporal coherence. Communication bandwidth constraints restrict data amount and rate of exchange between distributed nodes, thus smart data compression and priority strategies are required. Synchronization problems are created by the coordination needs of data gathering and processing across multiple distributed nodes with different clock sources and network latency. Scalability problems occur when an increasing number of vehicles and infrastructure devices are involved, which can swamp communication networks and processing power.

The intersection of distributed perception with emerging technologies such as 5G networks and Multi-Access Edge Computing (MEC) has the potential to overcome many of the current limitations and also open up new opportunities. The pre-trained deep learning models hosted on edge AI computing platforms for real-time inference are essential to support the response times required for safety-critical autonomous driving applications. Distributed perception trends in the future are expected to focus on improving inter-node coordination, developing more effective data fusion algorithms, and developing standardized interfaces for seamless integration with various vehicle and infrastructure platforms.

Therefore the key benefits of distributed cloud computing for autonomous driving include (B. Wang et al. 2024):

- High Availability: Eliminates single points of failure by distributing data across multiple nodes.
- Scalability: Resources can dynamically adjust based on demand.
- Data Locality: Minimizes latency by processing data near its source.

2.2.3 Sensor Fusion

Sensor fusion represents one of the key segments in autonomous systems, enabling effective perception, navigation, and decision-making. A combination of information provided by multiple sensors enables AVs to construct a full picture of the world. Autonomous vehicles perceive the environment using different sensors like cameras, LiDARs, radars, and ultrasonic sensors, all of which have advantages and disadvantages:

- Cameras provide high-resolution images but perform badly under low light conditions.
- LiDARs create accurate 3D maps but are expensive and vulnerable to weather conditions.
- Radars provide good detection in bad weather conditions but with a lower resolution.
- Ultrasonic sensors detect effectively at short range with limited range.

Sensor fusion integrates these multiple sources into one coherent, accurate model of the environment, removing several of the relative shortcomings of each sensor. It is generally divided into three different approaches (Yeong et al. 2021): low-level fusion, mid-level fusion, and high-level fusion.

Low-level fusion, often referred to as data-level fusion, is a combination of unprocessed raw sensor data. It retains the most information from the sensor but requires a lot of computational resources as well as precise alignment of the data from different sensors.

Feature-level or mid-level fusion processes the raw sensor data individually to extract relevant features and then combines these features. In this way, it balances the trade-off between computational efficiency and data richness and is one of the most common techniques used in AV perception systems. effectiveness for different driving scenarios.

High-level fusion, sometimes called decision-level fusion, is the process of combining outputs from individual sensors after their independent processing. Each sensor independently detects and classifies the object in this approach, followed by combining the final decisions to provide a unified representation of the environment.

Apart from that, different sensor fusion techniques and algorithms allow the AV to perceive the environment around it in a proper manner. One of the most common methods is Kalman Filtering, in which dynamic object state estimation updates the prediction recursively using sensor measurement. Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) are the commonly used variants that handle non-linear motion models effectively. It approximates the probability distributions of object states and is therefore suitable for localization and mapping in dynamic environments. Sensor fusion with deep learning has gained traction a few years ago. CNNs and transformer based architectures take multi-modal sensor data to improve the performance of object detection and classification.

2.2.4 Computer vision

Computer vision is at the core of autonomous driving perception, enabling vehicles to interpret visual data from the environment for object detection, classification, scene understanding, and decision-making. Robust computer vision systems are essential for the accurate and reliable identification of dynamic and static entities including vehicles, pedestrians, cyclists, traffic signs, and road markings as well as for real-time analysis of complex urban scenes. The integration between computer vision and other sensor modalities, such as LiDAR and radar, further strengthens the perception pipeline, providing a detailed and robust world model critical for safe autonomous operation (Tan et al. 2024). Essentially, computer vision encompasses several staging steps: (Janai et al. 2021):

Image Processing

Image acquisition in computer vision for an autonomous vehicle is a very important step, as good perception mandates high-quality data. Cameras have to be strategically placed to maximize coverage and minimize occlusions. Raw images are then subjected to image preprocessing techniques such as lens distortion correction, noise reduction, and histogram equalization to basically enhance image clarity. Furthermore, real-time adaptive preprocessing methods make sure that the changes in lighting and weather conditions do not deteriorate the performance of downstream perception tasks. Such preprocessing algorithms prepares input data and ensures that autonomous vehicles maintain consistent image quality for reliable decision-making.

Image Segmentation

Image segmentation allocates a class label to each pixel in the image, yielding discriminative masks that mark exact object and surface boundaries across the scene. In semantic segmentation, all pixels of a class share one label (e.g. road, lane, sidewalk, vegetation), enabling drivable-area estimation, lane geometry extraction, curb and median delineation, and some rule-based restrictions for planning in complex environments. Instance segmentation distinguishes between individual objects in the same class and assigns each its mask, enabling counting, tracking, and occlusion handling in congested scenes like intersections and merges. Compared to object detection, which only coarsely localizes items via bounding boxes, segmentation identifies pixel-accurate shapes and surfaces that are often integral when planning through tight spaces, inferring precise free space, or enforcing safety margins around vulnerable road users and static obstacles.

Image Feature Extraction and Object Detection

Feature extraction involves transforming raw images into discriminative representations of salient features that allow models to recognize salient categories. Based on those features, object detection identifies and localizes discrete, countable objects and/or instances (e.g., vehicles, pedestrians, and signs), typically based upon predicting class labels and bounding boxes around each object. Current, popular object detectors (like YOLO and Faster R-CNN) utilize versatility in dimensionality for feature learning and combine that with an objective function that effectively balances accuracy and speed needed for on-vehicle and infrastructure-assisted perception. Detecting and localizing objects from images is important because it is typically lower latency than segmentation (analyzing pixel levels of images), is more bandwidth efficient, and it amounts to provisions of compact yet object-level summaries (ID, class, position, confidence). While not typically as latency efficient, segmentation provides more detail than detection in the forms of shape and area. In addition, autonomous

vehicles typically use detection for dynamic agents but utilize segmentation methods for drivable-area and boundaries and extract the benefit of a robust scene model for capabilities of safe and informed decision-making.

2.3 Related works

This section explores related works that not only apply the aforementioned concepts but also leverage pertinent technologies, which will be further explained in the next chapter.

2.3.1 COPADRIVe

COPADRIVe (Vieira et al. 2019) is a co-simulation framework that enables the simulation of complex, cooperative vehicular applications in realistic scenarios. The different tools put together in this integration include Gazebo, OMNeT++ network simulator, and the ROS framework, thus setting up a platform for testing and evaluating both control and communication aspects of advanced vehicular applications. COPADRIVe uses the Veins simulator for vehicular network simulations and Vanetza implementation of the ETSI C-ITS protocol suite. The integration between Gazebo and OMNeT++ is done through ROS's publish/subscribe mechanisms.

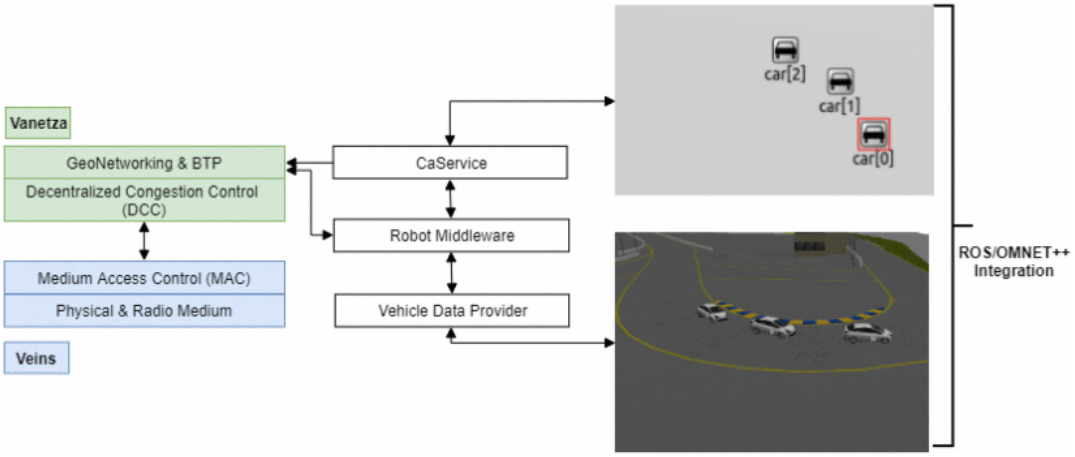


Figure 2.2: COPADRIVe framework's architecture (Vieira et al. 2019)

2.3.2 AuNa

AuNa is a modular integrated simulation framework for the purpose of cooperative autonomous navigation. In an effort to develop the integration of robotics, communication, and control simulations, AuNa merges ROS2, OMNeT++, and MATLAB for the simulation of cooperative driving scenarios (Teper et al. 2022). It was built based on COPADRIVe and inherited its basic architecture, introducing efficiency, flexibility, and more powerful synchronization between robot and communication simulations. Unlike COPADRIVe, AuNa uses ROS2 and the Artery framework, thus supporting a wide range of scenarios beyond platooning. AuNa is composed by:

- **Robot Simulation:** Simulation environment supports dynamic spawning of multiple robots. Each robot is assigned a unique identifier. Every robot's individual connections are maintained through ROS2 nodes, allowing direct control and publishing of sensor data. Gazebo supports simulation in this work by allowing plugins for sensor integration and robot control.
- **Communication Simulation:** For communication, AuNa integrates OMNeT++ and Artery. In order to have a proper synchronization between OMNeT++ and the other simulations, a custom scheduler within OMNeT++ synchronizes the execution of events, the synchronization between OMNeT++ and Gazebo, and the updates between the navigation system in ROS2 and communication module in OMNeT++ of each robot.
- **Control Simulation:** Control system designs and implementations of various complexities are done in MATLAB and Simulink. Support for ROS2 in the 2022 release is provided through the ROS Toolbox, and the framework integrates these tools directly, which means there is no need for manual controller coding.
- **Navigation Systems:** AuNa uses the Nav2 package for autonomous movement. The package provides an entire suite of algorithms to execute localization, mapping, and navigation in a dynamically adapting environment.

Compared to COPADRIVe, AuNa improves certain aspects, such as the efficiency in the synchronization of the robot simulation and communication simulation. Also, AuNa has been implemented with ROS2 and does not limit itself to platooning scenarios only, Artery is extended beyond the scope of the cooperative vehicle application.

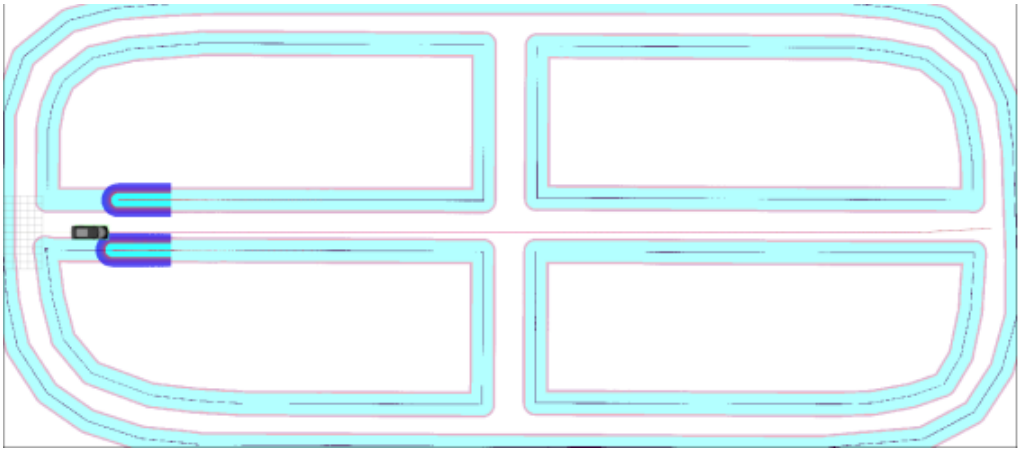
2.3.3 Towards the simulation of cooperative perception applications by leveraging distributed sensing infrastructures

This work introduces the devolvement of a simulation framework that leverages distributed sensors in the environment, thus enabling the augmentation of AV perception and its cooperation with strategically placed sensing devices. Starting from the concepts developed on AuNa, the proposed system increases its perception capability by including the extended vehicle through the presence of wireless sensors and actuators to enhance situational awareness in dynamic, unpredictable scenarios.

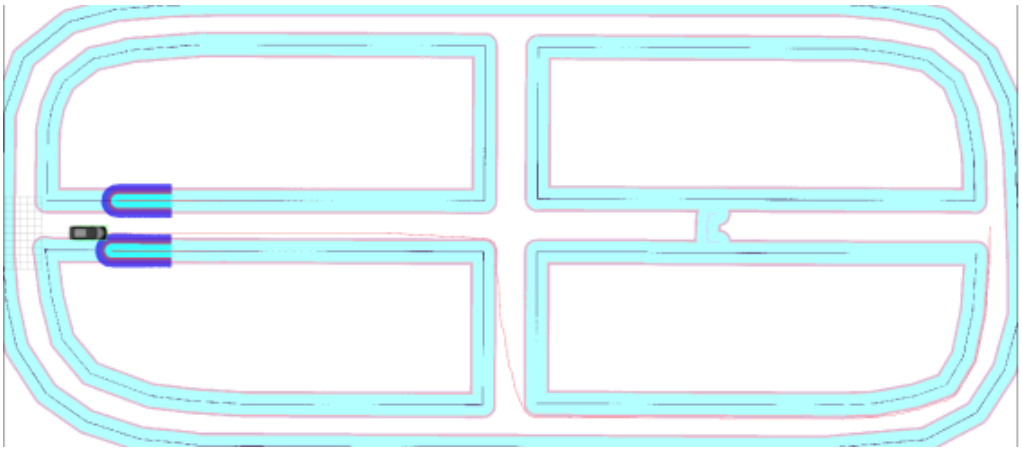
The framework is based on ROS2 and micro-ROS as the enabling technologies to integrate several simulation tools. It enables the development, testing, and validation of cooperative environments, including external sensors that collect data and communicate with a RSU.

The RSU aggregates all sensor data into obstacle messages transmitted to the AV via ITS-G5 communication. This helps the AV to undertake route planning and do the necessary maneuvering to safely pass these obstacles towards the destination.

In this work, the extended contributions to AuNa are: implementation of a new scenario (figure 2.3) where a vehicle faces a blocked road with alternative paths detected by nearby sensors. Integrated framework which embraces a LiDAR sensor model based on a Hokuyo implementation in Gazebo. Micro-ROS bridge for connecting sensor data with ROS2 nodes, and custom OMNeT++ Artery model to support efficient V2X communication. Therefore it allows for the realistic simulation of scenarios related to cooperative perception, putting the accent on the interaction of AVs with their environment.



(a) Path Generated without RSU.



(b) Path Generated with RSU.

Figure 2.3: Comparison of simulation scenarios (Oliveira 2023)

The proposed framework illustrates how distributed sensing infrastructures can contribute to AV perception and adaptability. The extension of the perception system with external sensors makes the framework effective in simulating cooperative scenarios and lays the ground for the seamless deployment of smart, cooperative environments using the same modular software architecture.

2.3.4 Supporting the Assessment of Energy Efficient Vehicular Automated Systems by leveraging Distributed Perception

This work extends Oliveira 2023 by providing the underlying support to the capabilities of the distributed sensing infrastructures and extending the simulation framework by providing new maps, vehicles and sensors on which the potential of distributed sensing infrastructures and cooperative perception can be demonstrated.

The proposed approach involves a highway inspired simulation scenario with distributed radar sensors, which provide data to the vehicles entering the highway in order to correct their speed and trajectory for energy-efficient integration into traffic.

Key contributions of this work involve:

- **Highway Simulation Environment**: A new world map with a Cloverleaf Interchange in a highway environment has been developed in Gazebo. Providing a realistic environment for testing AV scenarios, such as highway merging.
- **Optimized Vehicle Model**: Due to resource limitations, a lightweight version of the Prius vehicle was developed.
- **Dynamic Traffic Generation**: Traffic generation capabilities were introduced that allowed the simulation to include the running of vehicles in straight line motion on the highway.
- **Ultrasonic Sensor Integration**: It has integrated a new Ultrasonic Sensor Package, which detects object distances. It sends out its output in standardized ROS Range message formats.
- **Architecture Improvements**: Scalability and usability were further improved in the framework architecture. This included modularization of sensor logic into cohesive packages, standardized ROS message formats, improvements in code readability and documentation.

2.3.5 A Camera–Radar Fusion Method Based on Edge Computing

Fu et al. 2020 proposes a sensor fusion of roadside cameras and millimeter-wave radar to enhance vehicular perception (figure 2.4). The central contribution of this research work is, therefore, directed toward how to develop an efficient fusion pipeline which can process and synchronize the data from both sensor modalities. The proposed method adopts multi-access edge computing for the realization of real-time data processing, which reduces the communication delay between roadside sensors and vehicles. Two major contributions have been made in this study: developing the fusion algorithm architecture and subsequent evaluation through simulations.

The algorithm is modularized into four key components: data preprocessing, spatial synchronization, time synchronization, and multi-object association and tracking. Data preprocessing was an important predecessor, as it is not possible to use the raw detections provided by cameras and radar due to noise and inconsistency. The used deep learning-based object detection method, YOLOv3, processes the camera images in this work, on the other hand, a DBSCAN clustering algorithm refines the radar detections. The combined use of the two allows one to robustly classify objects and make initial target detections. Spatial synchrony then follows, by jointly calibrating sensors in such a way that data from radar and cameras is

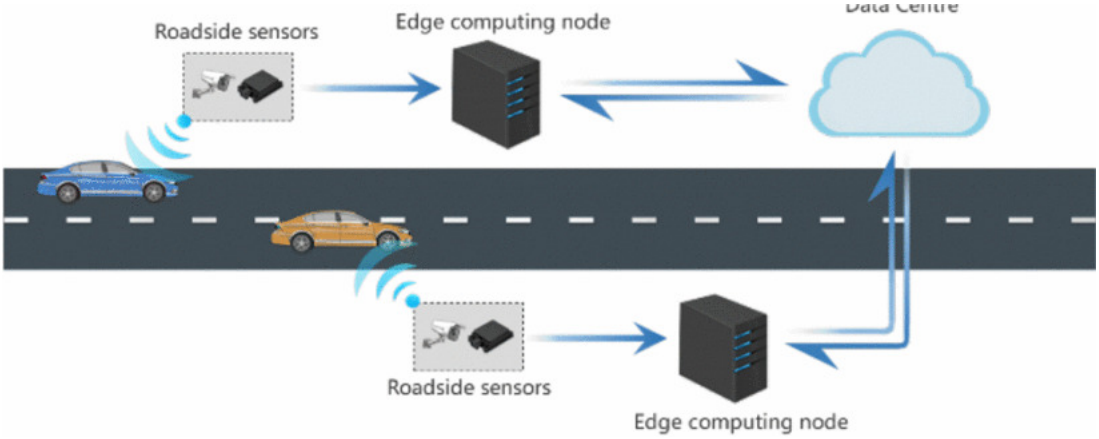


Figure 2.4: Roadside perception method based on edge computing (Fu et al. 2020)

transformed to one single coordinate system. This involves coordinate transformation and distortion correction, especially for the camera using the chessboard calibration technique.

To resolve the divergent sampling frequencies between radar and cameras, they designed a multi-thread time synchronization approach to solve this challenge. Within the paper, they introduce the three parallel threads: the thread of radar, the thread of cameras, and the thread for fusion, all these independently undertake data collection and updates a buffer through a direct update mechanism in real-time. The fusion thread fetches the latest synchronized data from both sensor, this ensures that only the most relevant detections are processed. The last step in the fusion process is the multi-object association and tracking. Object association is done using the Munkres algorithm to ensure that radar and camera detections relate to the same physical object. Then, for tracking, the Kalman filter continuously estimates the position of objects by filtering out noise and predict future states based on previous detections.

After designing the algorithm, the proposed fusion method is evaluated through a simulation experiment. The simulation environment is designed using MATLAB's Automated Driving Toolbox, which enables the creation of realistic road scenarios, including traffic interactions and sensor simulations. The authors choose a road intersection as the test environment where vehicles travel at a constant speed while roadside sensors capture the movement. This is relevant because intersections usually contain occlusions and complex object interactions where sensor fusion becomes necessary for accurate perception.

In performance assessment, the proposed approach utilizes three error metrics, that is, mean residual (MR), variance, and residual sum of squares (RSS), where MR characterizes the mean value of all errors between the perceived and true positions of every vehicle. The metric of variance, in turn, calculates the stability or instability of these detections. In other words, each of the detected squared errors cumulates to a single number. The selection of the sensor parameters is done in order to reflect the real specifications for both radar and cameras. Radar is set 0.2m above ground, with a maximum of 170m in detection range, while the camera is placed 3m high, with ±45° as its detection angle.

Experimental results show that the proposed fusion algorithm performs much better in tracking stability compared to the single-sensor methods. From the longitudinal direction analysis,

the variance of camera and radar detections is very high, meaning unstable tracking, however, after fusing the data, it improves stability, unlike the individual sensor readings. The residual sum of squares increases with slight misalignment when the vehicle approaches the sensor, on the other hand, the general accuracy of perception increases. For lateral direction analysis, noise has been filtered out by the fusion algorithm rather effectively, while it also manages to align the detected position closer to the ground truth. This therefore shows that most of the sensor limitations, particularly in conditions whereby occlusion and erratic movements beyond the capability of a single sensor occur, can be overcome.

In a nutshell, the paper provides a structured insight into edge computing-based camera-radar fusion that ensures marked improvements in perception stability. The results depicted prove that the fusion algorithm effectively merges various preprocessing, synchronization, and tracking techniques, while the simulation results confirm its advantages compared with single-sensor perception.

2.3.6 Experimental Study of Multi-Camera Infrastructure

The authors of Shan et al. 2024 address the robust perception in complex road segments, like highway merges, where onboard sensors are often limited by occlusion, and fields of view may be restricted. Their experimental setting deployed a multi-camera roadside infrastructure on the M40 motorway in the UK featuring low-cost monocular cameras with overlapping fields of view across 650 - 700 m. Unlike many studies that rely on precise calibration (which was impractical here), they explicitly focused on the absence of known camera intrinsics and extrinsic, and instead: (1) manually modeled projective transformations, and (2) exploited the redundancy across cameras. The perception pipeline included YOLOv4-based vehicle detection, projecting bounding boxes into a single bird's eye view (BEV) map, estimating centroids based on multi-camera fusion, and then tracking via Kalman filter and Global Nearest Neighbor (GNN) association method. The paper details a number of post-processing approaches to filter out false positive and occlusion artifacts including class filtering and handling partial occlusion. Evaluation against a high-precision dGPS ground truth presented an average lateral localization error of 48 cm, performing much better than other calibration-free approaches, and closely matching radar-based vehicle tracking accuracy. According to the research, the system continuously tracked around 75% of vehicles in real traffic conditions to confirm it worked effectively.

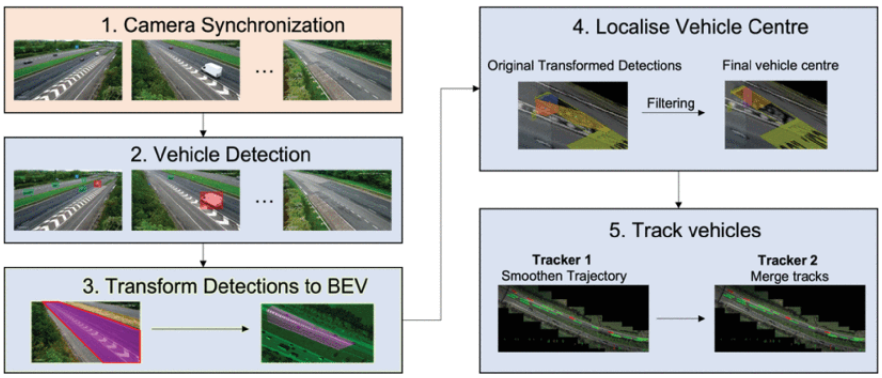


Figure 2.5: Overview of the processing pipeline Shan et al. 2024

The research emphasizes that infrastructure-assisted cooperative perception is practical using inexpensive cameras, edge computing, and V2X communication. The most notable

contributions are that overlapping multiple camera setups could significantly alleviate occlusion, estimate location with near radar quality, and recognize objects. This work represents an important experimental compromise, demonstrating the possibility for distributed camera-enabled infrastructure to assist with safety-critical maneuvers like merging on the highway, while at the same time, highlighting realistic objectives such as fine-tuning camera positions, handling occlusions, and real-time fusing of multiple sensors in an operational environment.

2.3.7 Novel Decision-Making Strategy for Connected and Autonomous Vehicles in Highway On-Ramp Merging

Kherroubi, Aknine, and Bacha 2022 specifically examines the implementation of an infrastructure assisted framework to achieve safe and efficient highway on-ramp merging of Connected and Autonomous Vehicles (CAV). The authors indicate that merging is one of the most significant and dangerous maneuvers in road traffic, and highlight the issues and challenges with relying on onboard sensors alone, which experience occlusions, limited sensing range, and computational constraints. The developed system proposes integrating roadside units that could not only assist CAV but also allow for extended perception beyond vehicle capabilities, and afford a clear view of the environment, even in mixed-traffic conditions when human-driven vehicles exist.

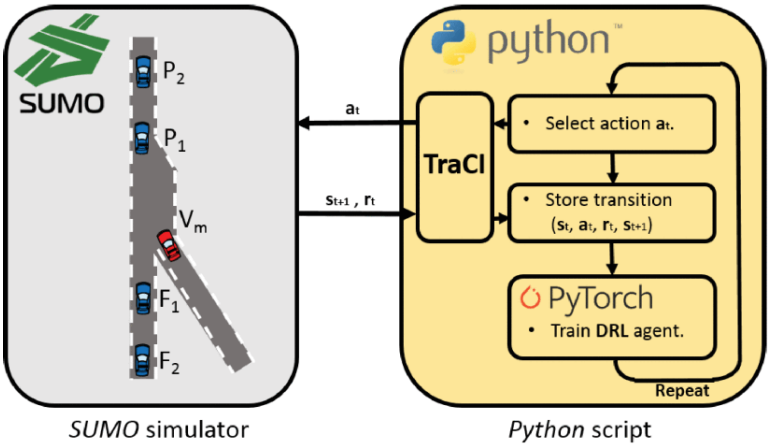


Figure 2.6: Overview of the simulation framework Kherroubi, Aknine, and Bacha 2022

One of the significant contributions of this work is the integration of distributed perception with predictive modeling of human driver intentions. The RSU gathers data from vehicles in the control zone and makes use of an Artificial Neural Network (ANN) to determine if the main-lane driver will yield or maintain course, achieving over 99% confidence in their predictions. The estimated intention is then integrated into a decision-making framework consisting of Deep Reinforcement Learning (DRL) with the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm. Incorporating predictions into the state space leads to safer and more collaborative merging maneuvers, with zero collisions over the course of thousands of simulation episodes.

An assessment was conducted through a co-simulation environment that coupled the SUMO traffic simulation system with DRL training that used PyTorch, with traffic patterns sourced from the NGSIM dataset. In the results, it was shown that intention-aware agents not only engaged in safe and comfortable behaviors, they also converge on stable policies better than

baseline DRL agents. The use of a safety controller also ensured that the agent adhered to traffic rules, including acceleration, and distance violations.

This work is significant to the state of the art as it illustrates how infrastructure assisted distributed perception can be used with predictive modeling and learning based decision making to solve one of the most safety-critical maneuvers in automated driving.

2.3.8 Merging control strategies of connected and autonomous vehicles at freeway on-ramps: a comprehensive review

Zhu, Easa, and Gao 2022 provides a comprehensive overview of cooperative merging strategies, synthesizing results from 44 studies of freeway on-ramp merging with CAV. The article notes that merging areas are one of the most significant bottlenecks in freeway operations and create traffic oscillations, increased fuel consumption, safety concerns, and congestion. CAV can coordinate at a fine-grain, vehicle-level via communication and motion control, which is different from controlling at an aggregated level like ramp metering.

The literature examined in this survey is classified by traffic composition (CAV only and mixed CAV and human-driven vehicle flows) and freeway configuration (single-lane and multilane). Most contributions addressing single-lane, CAV-only traffic consist of search strategies for trajectory optimization, virtual car following models, and integrated sequence–trajectory planning utilizing heuristic or optimization methods. In mixed traffic scenarios, the uncontrollability of human-driven vehicles creates uncertainty that requires predictive models, including neural networks and logistic regression, to predict driver intentions and use the predictions as inputs for joint trajectory planning frameworks. Multilane freeway studies take into consideration lane-changing behaviors that introduce additional complexity and/or must formulate coordination strategies to unify the optimization of longitudinal and lateral motions, which are often guided by optimization and game-theoretic principles. Furthermore, many of these studies are beginning to move beyond solely lane-changing control towards hybrid strategies that connect lane-changing control and conventional traffic management systems.

The survey delineates a number of open challenges that are particularly applicable to distributed perception and collaborative systems. Most of the work focused on trajectory design has continued to focus mostly on lower level trajectory design, with minimal interaction between the tactical (gap selection, sequencing) and operational (trajectory regulation) layers. Additionally, the human-centric vehicle behavior is usually over-simplified in these studies meaning that stochasticity is not recognized nor is variability in driving behavior under real conditions. Lastly, multilane scenarios, despite being more realistic are frequently simplified to discrete decision spaces in the literature, which points to a need for more advanced continuous control solutions.

For this thesis, the review is of specific relevance because it highlights hierarchical control architectures as well the function of predictive models in mixed-traffic scenarios. Also, it recognizes the gap in the literature where perception and coordination frameworks need to develop capabilities for handling multilane, complex environments.

Chapter 3

Technologies

This chapter overviews the principal technologies that will be used during the development of this work.

3.1 Simulation Tools

3.2 ROS2

ROS2 is an open-source middleware framework for the development of robotic systems. It addresses the limitations of its predecessor, especially those concerning real-time performance, security, and multi-robot systems, making it suitable for both research and industrial applications (Bonci et al. 2023). ROS2 uses Data Distribution Service (DDS) as the communication layer, which provides reliable and real-time communication between distributed robotic components across heterogeneous platforms.

ROS2 distributed and modular architecture makes it possible to decompose complex systems into multiple nodes (independent unit) responsible for a specific functionality (e.g. sensors, actuators, etc.). The nodes can then communicate through various communication paradigms (figure 3.1):

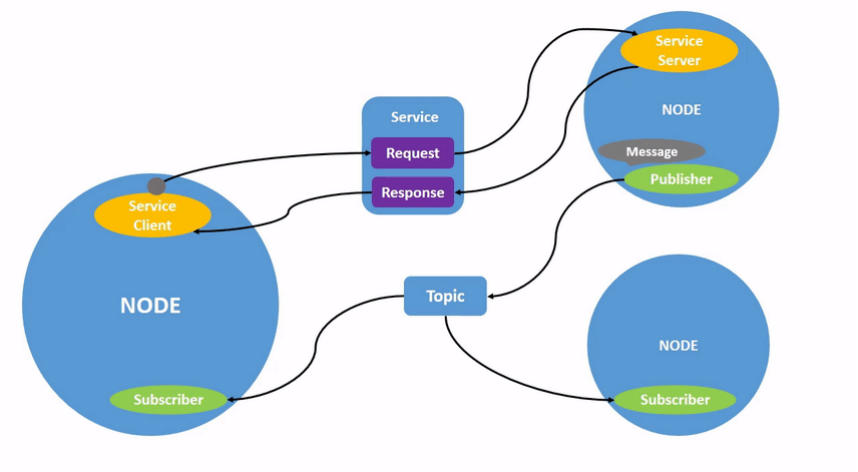


Figure 3.1: ROS2 nodes, topics and services (ROS2 2025)

- Topics: Asynchronous publish-subscribe communication channels to exchange data such as sensor information or perception outputs.
- Services: Synchronous request-response communication useful for short-lived queries, like parameter requests or control commands.
- Actions: Handles more complex, long-running tasks with feedback and cancellation support, essential for processes like motion planning or sensor calibration.
- Parameters: Configurable data shared across nodes, allowing dynamic system adjustments at runtime without recompilation.

3.2.1 OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++) (figure 3.2) is an open-source discrete-event simulation framework specifically designed to model communication networks, distributed systems, and other complex systems (Varga 2019). Due to its modular and object-oriented architecture, OMNeT++ is able to model network devices, protocols, or logical components in a flexible way. This allows the adoption of OMNeT++ in education and industry for research and development involving wireless networks, vehicular communication, Internet of Things, and cloud computing. Facilitating the evaluation of communication protocols and cooperative systems in controlled, repeatable, and scalable scenarios.

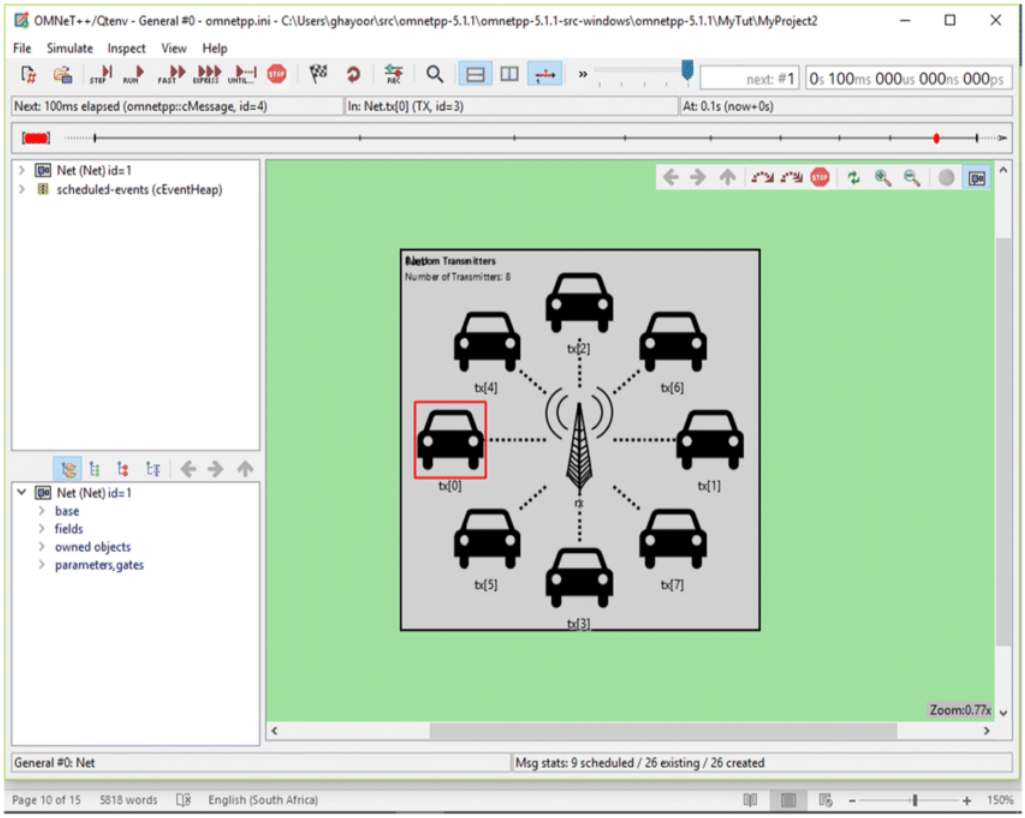


Figure 3.2: OMNeT++ simulation model (Keivani, Ghayoor, and Tapamo 2021)

The framework's fundamental building blocks include:

- Modules - are the primary computational entities, representing vehicles, roadside units (RSUs), or network infrastructure components. Each module encapsulates specific behavior, state information, and communication capabilities through modules.
- Messages - information exchange between modules, containing data payloads and scheduling information for discrete event processing.
- Channels - define communication pathways between modules, specifying transmission characteristics such as delay, bandwidth, and error rates.
- Network Description (NED) files - provide a declarative language for defining simulation topology, module hierarchies, and parameter configurations. This separation of model structure from implementation enables rapid prototyping and configuration management for complex cooperative scenarios.

To extend its vehicular networking capabilities, OMNeT++ integrates with specialized frameworks:

- VEINS – couples OMNeT++ with the SUMO traffic simulator through TraCI, allowing network decisions to influence traffic dynamics and vice versa.
- Artery – implements the ETSI ITS-G5 V2X protocol stack, enabling realistic cooperative perception scenarios with standard-compliant message exchange.

OMNeT++ is a particularly useful tool for this thesis as it allows realistic modeling and assessment of communications, which are crucial for the co-operative perception system. It enables analysis of Quality of Service (QoS) parameters like latency, jitter, packet loss, and throughput, all of which are key to ensuring that data is transmitted and received in an appropriate time frame with a level of reliability between vehicles and infrastructure. In addition, its scalability features allow simulation scenarios with thousands of communicating nodes, thus modeling system performance in relation to density of vehicles, as well as coverage area of the RSU. OMNeT++ also directly supports the comparison of different communications technologies, such as IEEE 802.11p, under the same experimental environment, and provides information on the use of protocols for vehicular real-time applications. Lastly, one of the most important feature is its support for multi-domain co-simulation, which is the ability to concurrently execute a simulation with ROS 2 and Gazebo, ensuring that the communication effects manage the perception and control processing and vice versa. Thus providing a complete experimental environment for evaluation of cooperative and distributed vehicular systems.

3.2.2 ns-3

The network simulator ns-3 (Nsnam 2025) is a open-source discrete-event simulator for modeling IP-based protocols and wireless networks used for both academic and research purposes. Ns-3 is predominantly implemented in C++, with optional bindings in Python and is released under the GNU GPLv2 license. Ns-3 is also modular and extensible which makes adding and mixing protocols and custom components easy along with modeling communication across multiple layers of the protocol stack. It has representative data-link and physical models that include PHY/MAC layer models for devices such as Wi-Fi and LTE, and some of the layered protocols support routing protocols like OLSR and AODV. The ns-3 consists of a scheduler, and the events or functions can be used to schedule the events in real-time

as well, which supports simulation-in-the-loop with real devices. A Direct Code Execution (DCE) framework enables an entire application to run within a ns-3 simulation, while the packets are still passed through a processing node and onto the physical interface, which also makes it ideal for hybrid emulation setups. Overall, ns-3 is a comprehensive simulator and flexible for a variety of high-fidelity studies, and captures reproducibility considerations.

However, despite these advantages, ns-3 has some limitations for research on vehicular and cooperative perception. One of the limitations is the absence of native support for ETSI ITS-G5 or CPM object exchange, so custom implementations of vehicular V2X protocols are necessary. Another limitation of ns-3 is that the original propagation models were not thought through in terms of highly dynamic V2V links, suggesting that additional extension may be necessary to model systems that feature Doppler shift and fast fading. One final consideration is computational expense with large scale vehicular scenarios, and the low-level architecture is still steep for anyone who wants customization beyond what is present.

3.2.3 Gazebo

Gazebo (figure 3.3) is an open-source simulation software for 3D purposes intended to aid robotic systems development, testing, and verification. It can simulate complex scenarios with high-performance physics engines. The tight integration with the Robot Operating System (ROS/ROS2) is one of the principal features of Gazebo. Gazebo makes it possible for simulated robots to publish and to subscribe to ROS topics, allowing for smooth interaction between virtual sensors and actuators and higher-level algorithms for control. In practice, this will mean simulated LiDAR or camera data can be supplied to and processed by the same perception pipelines or control systems that would run on the real hardware. The virtual world effectively allows researchers/developers to take algorithms from simulation to deployment without losing continuity during the development. For example, the algorithms can be tested, calibrated and ultimately their performance verified in simulation environments prior to its deployment in real-world robotics environments. Secondly, Gazebo's extensibility makes it easy to extend it with single plugins for various simulated traffic patterns, single road infrastructure designs, and various simulated environmental disturbances, including weather factors making it very applicable to autonomous driving research.

Figure 3.3: Gazebo simulation (AbdelHamed, Tewolde, and Kwon 2020)

From a broader perspective, Gazebo offers several key features:

- Realistic World Modeling: Ability to generate detailed urban and highway environments with lanes, road signs, and traffic participants.
- Sensor Fidelity: Camera, LiDAR, radar, and a host of other sensors available with configurable parameters and realistic noise models.
- Physics Accuracy: Rigid body dynamics and collision handling capability for realistic simulation of vehicles and obstacles.
- Scalability: Capability to simulate multiple vehicles and sensors running at the same time, a prerequisite for cooperative and distributed perception scenarios.
- Extensibility: Capability to support for custom plugins that generate events, disturbances, or novel sensor models specific to research.

Gazebo's capability to mimic realistic sensor readings and dynamic situations allows the testing of cooperative perception and data fusion methods. Gazebo can simulate situations such as occlusions, variable traffic densities, and speeds of movement that provide enough realism to realistically evaluate the proposed system. When combined with ROS 2 for system orchestration, and OMNeT++ for network modeling, Gazebo can create a realistic enough world to test the co-simulation system. In the end, Gazebo helps facilitate the connection between the theoretical design, and real-world validation to create a safe, flexible, and repeatable experimental method to test distributed perception. This fits in well with the goals of the thesis, as the goal is to efficiently study perception and decision-making in a realistic but controlled setting.

3.2.4 CARLA

CARLA (Car Learning to Act) is an open-source simulator specifically built for research in autonomous driving and released under the Apache 2.0 license. CARLA utilizes Unreal Engine and offers photorealistic rendering, realistic physics, and a client-server architecture, separating world management from agent control throughout the simulation. This system design, which can streamline experimentation by distributing the workload, as well as provide easy interaction with any external framework through its Python and C++ APIs, has made CARLA a popular choice for large-scale autonomous driving research (Silva et al. 2024).

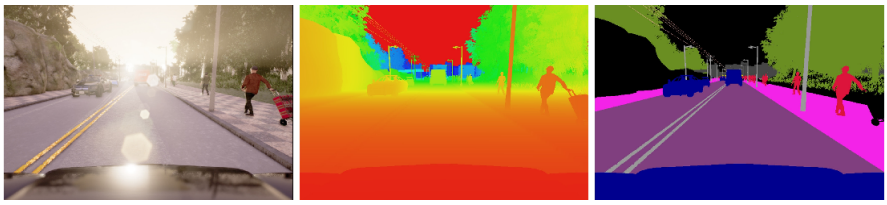


Figure 3.4: Three of the sensing modalities provided by CARLA (Dosovitskiy et al. 2017)

CARLA also has a complete suite of virtual sensors, including cameras, LiDAR, radar, and GPS which are all customizable with noise models. It also has the ability to simulate weather, light, and traffic conditions using the traffic manager, allowing multi-agent interactions like lane changing and handling intersections for realistic simulations. These features allow CARLA to simulate testing of perception and planning systems for autonomous driving in complex and dynamic road conditions.

It has also been applied in cooperative perception and V2X research. Connections with communication simulators allow comparisons of the performance of the DSRC and Cellular V2X (C-V2X) networks with different network loading levels, enabling the exploration of the consequences of latency, packet loss, and bandwidth limits on cooperative decision-making. Work in extensions to CARLA, such as R-CARLA, provides even more fidelity by modeling sensors to help bridge the simulation-to-reality gap, while platforms such as CARLA-GymDrive can integrate and support reinforcement learning pipelines and task-specific decision-making and control.

From a cooperative perception point of view, CARLA's primary strengths are its realism in representing multi-agent environments and its ability to simulate infrastructure-based sensing and V2I communication. These attributes serve as a foundation to explore scenarios such as cooperative highway merging, distributed sensing, and intersection management. However, the computational load of rendering and physics still makes it difficult to simulate at a larger scale, and hardware-independent deterministic reproducibility as a result is still an open question. Although these are limitations, CARLA is one of the most capable tools for autonomous driving research involving high-fidelity sensor simulation and is extensible for cooperative perception research.

3.2.5 Critique Analysis

This section analyzes the technologies introduced in the previous sub-sections, outlining alternatives and stating why they were not selected within this work. The purpose of this analysis is to justify the reasoning behind selecting ROS2, Gazebo, and OMNeT++ as the simulation stack, while also considering their strengths and weaknesses compared to competing options.

Starting with the robotics middleware component, the utilization of ROS2 is a natural choice, not only because it is the current state-of-the-art when it comes to robotics research, but also due to the framework somewhat being a result of prior work built around integrating ROS2. Alternatives offer reasonable capabilities however they lack the depth of community support, package ecosystem, and lack of compatibility directly from Gazebo or OMNeT++. The modular node-based structure of ROS2 as well as its use of DDS for a communication framework together provide for scalability and real-time message exchange across heterogeneous agents, which is needed when working in distributed perception principles. ROS2 also provides continuity with prior work in autonomous driving and cooperative perception, allowing us to continue from work already developed.

In the area of network simulation, two leading candidates were identified: OMNeT++ and ns-3. Both simulators are discrete-event simulators that are well established in the academic setting. ns-3 is best known for its packet-level fidelity, the large volume of IP-based protocol models, and framework to handle hybrid emulation with actual systems. However, ns-3 lacks native vehicular networking extensions, including ETSI ITS-G5 and Cooperative Awareness Messages, which means that much more customization would be necessary to support V2X scenarios. OMNeT++, on the other hand, offered a more modular architecture, better visualization, and purpose-built vehicular networking architecture in frameworks like VEINS and Artery. Artery also has built in support for ETSI ITS-G5, which lends itself well to cooperative perception type applications. In addition, this research had already integrated OMNeT++ in the overall framework for this work to support smoother interoperability with ROS2 and Gazebo.

When it comes to vehicle simulation or robotics, Gazebo and CARLA are two state-of-the-art tools. CARLA has cutting-edge visual rendering, realistic multi-agent traffic behavior, a suite of sensors and is otherwise a good platform for studying autonomous driving. Unfortunately, CARLA can present some challenges in its computing resources, flexibility beyond the autonomous vehicle realm, and synchronization obstacles in a large-scale multi-agent backtracking simulation during an autonomous vehicle task. In the other hand, Gazebo works seamlessly with ROS2, has lighter computational demands, and has enough realism to compare behavior in distributed perception running under highway merge scenarios. In addition, Gazebo functionality is geared toward extensibility using plugins, representations of infrastructure-based sensors, and lightweight vehicle models, rendering it a good option for simulating many individual agents. Gazebo is most importantly the simulator that the adopted co-simulation framework operates under, allowing us to synchronize in real-time with Omnet++ for communication effects and ROS2 for perception and control loops.

In conclusion, while ns-3 and CARLA are interesting alternatives with unique benefits, using ROS2 + Gazebo + OMNeT++ offers a unified, highly compatible stack for our work. This compatibility is primarily due to the existing versions of ROS2 + Gazebo + OMNeT++ being well-tested and supporting reproduction and efficiency, rather than exclusivity. The deep integration of these tools means this thesis can focus on the assessment of cooperative perception and distributed sensing and not on developing new integration layers.

3.3 Vision

The primary sensing modality for autonomous driving and infrastructure-assisted perception relies on vision because it delivers detailed semantic information and abundant spatial data at lower costs than LiDAR or radar sensors. The combination of cameras enables vehicles and roadside infrastructure to detect and record detailed environmental information about lane markings, traffic signs, pedestrians and vehicle appearances. Supporting object detection, tracking, semantic segmentation and behavior prediction. The success of vision-based perception heavily relies on obtaining precise and time-consistent measurements, which demands calibration and synchronization throughout the entire sensing pipeline.

Camera Models and Lens Distortion

The pinhole camera serves as the base model for performing most computer vision operations in autonomous driving systems. The pinhole camera transforms a 3D point $X = [X, Y, Z]^T$ in camera coordinates into pixel locations u through the following relationship:

$$u = K[R|t]X \tag{3.1}$$

where K contains the intrinsic parameters (focal lengths f_x, f_y, principal point c_x, c_y and skew), and $[R|t]$ are extrinsic mapping the world frame to camera coordinates. This model underpins key metric vision tasks such as ground-plane projection and 3D localization.

Real lenses produce non-ideal pinhole projection by creating radial and tangential distortions that deviate from the perfect pinhole image. The Brown-Conrady model represents these deviations through radial coefficients k_1, k_2, k_3 and tangential coefficients p_1, p_2. The precise representation of these distortions is essential for 3D reconstruction and sensor fusion operations.

Intrinsic Calibration

The aim of intrinsic calibration is to estimate the camera matrix K and distortion parameters by minimizing the re-projection error of known target points to their detected image keypoints for multiple views. Two common strategies for intrinsic calibration include Zhang's calibration approach, frequently enhanced through non-linear optimization methods such as Levenberg-Marquardt. Improved accuracy is typically achieved by capturing calibration images with a range of tilts, scales, and distances to cover the entire image frame. In validation, per-view re-projection errors are reported, and stability of parameters is checked by repeated measurements and testing at different temperatures.

Extrinsic Calibration

Multi-camera systems demand precise relative poses in order to produce geometrically correct rays for triangulation and fusion purposes. Existing methods for pose estimation generally fall into one of two categories depending on whether or not their fields-of-view overlap. The simultaneous estimation of intrinsics and extrinsics across heterogeneous sensors can provide greater metric consistency, lessen dependence on overlapping fields of view, and foster greater repeatability across experiments. Specifically, when cameras are mounted on actuators or vehicles, estimating the rigid transform between the sensor and the moving frame is essential.

Temporal Synchronization

The temporal misalignment between sensors such as a camera and LiDAR can negatively influence multi-view associations, contribute bias to triangulation, and impact ground-plane projections. Hardware-based triggers or protocols such as PTP/IEEE 1588 allow for synchronization. In cases where there are no hardware-based methods of synchronization, the time offsets can be computed through the minimization of re-projection or association errors. Assuring that time is synchronized across the context of a sequence or over the entire experiment or simulation is important in any case, and successively estimating time offsets can provide confirmation of that synchronization.

Multi-Camera Calibration for Cooperative Perception

For going to infrastructure-assisted perception, accurate extrinsics with respect to a common or absolute world reference frame are key for consistent ground-plane projection and multi-view fusion. When the cameras FOV do not overlap, poses can be "chained" together using surveyed targets or some constraints of the local environment. By jointly optimizing all cameras simultaneously in a non-linear framework, re-projection and alignment errors are minimized, and global consistency of poses is improved for object triangulation and cooperative perception tasks.

3.3.1 Object Detection Algorithms

Object detection is a crucial ability for AV perception, as it allows the detection and localization of traffic participants. There are several families of algorithms comprising the object detection domain, and each has its own strengths and weaknesses in terms of accuracy, latency, and suitability for use in real-time systems.

Two-stage detectors like Faster R-CNN represent the traditional high-accuracy option in the domain. This architecture generates the region proposals and, afterward, classifies and refines the bounding box. A multi-stage architecture has improved performance in terms of accuracy for smaller or partially occluded objects, so it is advantageous when operating in confusing urban scenarios. On the other side, a multi-stage architecture is going to be slower to make an inference decision and may not be a reasonable option for real-time perception events in highway environments, where immediate decisions are the norm.

Alternatively, one-stage detectors like SSD (single shot multi-box detector) and RetinaNet, predict bounding boxes and class labels from feature maps without the intermediate proposal stage. In general, one-stage detectors would be faster in making inference decisions, and therefore they are a more reasonable option in terms of speed and accuracy tradeoff. However, a one-stage detector would struggle in scenarios with a high density of traffic participants, as the objects are more likely to be obscured and overlapped.

The YOLO (You Only Look Once) family of models has become a popular method for real-time AV detection, evidencing a favorable balance of accuracy, with an improved family of models (e.g. YOLOv5, YOLOv8, YOLOv11) having anchor-free detection heads, label assignment strategies, and decoupled classification and regression layers. With these updated versions, YOLO models have been able to realize speed and accuracy comparable to state-of-the-art one-stage detectors, but with much faster inference and significantly less latency. Furthermore, YOLO presents a family of models that are scaled by model size as "nano", "small" models, to "large", or "extra large" models, giving a user ease to select a model depending on computing limits or accuracy. For example, nano models can achieve inference

in the millisecond range for using on embedded systems or several camera scenarios, gaps from "mid" and "large" models will trigger with less data at distance using gaming or graphics processing unit computing systems.

Comparative research frequently indicates that YOLOv5 and YOLOv8 outperform earlier YOLO variants and often exceed SSD and RetinaNet in most precision and recall metrics on automotive benchmarks, but also yield per-frame single inference times that allow them to be real-time methods.

In terms of object detection, different families of algorithms were explored. The two-stage Faster R-CNN is a solid performer for accuracy when working with small or occluded objects, however its multi-stage architecture does impose a considerable overhead in latency and is not optimal for use in real time. The SSD (Single Shot MultiBox Detector) paradigm emphasizes inference time with less computational load but generally compromises detection accuracy from the current performance standards. Accordingly, the YOLO family strikes the best compromise of accuracy and inference speed. Investments in successive YOLO versions have improved precision and recall, while maintaining real-time performance and also allowing model scaling for optimization across heterogeneous hardware platforms such as edge devices with constrained resources or roadside units supported by GPU systems. Recognizing the need for this detection across multi-camera roadside units and lane gap inferences in real-time, it was decided to use YOLO as the object detector. Moreover, the supporting ecosystem for YOLO (e.g., ONNX, TensorRT, compatibility with ROS2) does minimize engineering overhead for the implementation process and increases deployment fidelity across the balance of the framework.

3.4 Communication Technologies

The evolution of autonomous vehicles (AV) is directly related to improvements in communication technologies. Central to this domain is the IoT, which consists of an interconnected network of different devices such as sensors and actuators, using standardized machine-to-machine (M2M) communication protocols. The IoT architecture allows relevant information to be transferred in real-time across a variety of applications, including smart traffic management, urban planning, wearable devices for healthcare, and specifically autonomous driving. AVs are able to communicate with other vehicles, infrastructure, and pedestrians (V2V, V2I, V2P) and thus are capable of higher levels of efficiency, safety, and responsiveness than vehicles that do not contain similar devices for communication. However, despite the potential of all of these developments, the IoT still has to deal with ongoing issues such as limited broadband infrastructure to the internet, effective big data scalability, and cybersecurity risks. The required real-time processing has also greatly increased demand for scalability in methods and supporting technologies, including cloud computing, AI, and big data analytics. These challenges indicate the greater need for scalable architectures and concurrent low-latency methods that exploit edge computing and AI.

3.4.1 ETSI ITS-G5

Europe has developed the ETSI ITS-G5 standard, also referred to as the Cooperative Intelligent Transport Systems (C-ITS) standard to allow robust communications between vehicles using Dedicated Short-Range Communications (DSRC). Like the IEEE 802.11p standard, ITS-G5 functions in the 5.9 GHz spectrum and is explicitly designed to facilitate communications in dynamic vehicle environments, such as on the highway traveling at speeds above 100 km/h. ITS-G5 provides high-priority and low-latency message exchange for Vehicle-2-Vehicle (V2V), Vehicle-2-Infrastructure (V2I), and Vehicle-to-Pedestrian (V2P) communications up to approximately 1 kilometer. As a result, ITS-G5 is appropriate in nearly every conceivable scenario, including suburban environments, highway environments and urban environments.

The ITS-G5 ecosystem is the basis for several cooperative transport applications to enhance both safety and efficiency on the road. Examples include:

- Collision Avoidance and Mitigation, which could facilitate instant alerts from other vehicles regarding hazards to limit accident risk.
- Intersection Safety, through communication with vehicles and traffic controllers to assign right-of-way and minimize conflict.
- Dynamic Traffic Management, where roadside units will continuously alert drivers to adaptive speed limits, lane assignments, or incidents.
- Efficient Routing and Navigation, using up-to-the-minute data about congestion and road conditions to provide improved travel flow and time reduction. congestion.

ITS-G5 employs Enhanced Distributed Channel Access (EDCA) as a method to prioritize safety-critical messages at the MAC layer, which will enable the timely delivery of messages, even during contention-based processes. These enhancements to the MAC layer will work toward maintaining predictable latency and reliability, even during high traffic density scenarios. However, congestion and interference on the communication channel are an unavoidable reality of a congested environment, such as an inner-city or a highway scenario, that involves

many nodes connecting to a limited spectrum. Therefore, adaptive congestion control algorithms, multi-channel coordination, and hybrid communications, such as the combination of DSRC with C-V2X, are now being researched for use with the next generation of cooperative systems to provide enhanced scalability and robustness against congestion.

3.4.2 C-V2X

Cellular Vehicle-to-Everything (C-V2X), which was standardized by 3GPP starting with Release 14 and improved upon between Releases 15 and 18, was developed to provide connectivity and services that complement short-range ITS-G5. C-V2X combines direct device-to-device sidelink (PC5) with operator-managed cellular connectivity. On the sidelink (PC5), vehicles and roadside units can exchange safety messages without sending any messages through the cellular core, which allows highly localized, low-latency V2V and V2I interactions that are resilient to loss of infrastructure. C-V2X leverages the cellular RAN/core over the Uu interface to provide wide-area services, including access to cloud services, traffic management, over-the-air updates, and coordination with edge compute. The dual-mode approach provides C-V2X with coverage and scalability. While sidelink supports immediate close proximity coordination, the network path allows for scalable further dissemination and orchestration of travel scenarios across corridors or entire cities.

Utilizing 5G for C-V2X means that ultra-reliable low-latency communication (URLLC) features can be introduced with multi-access edge computing (MEC) and advanced resource scheduling to address contention effects in dense traffic, stabilize latency/jitter and facilitate local computation for perception fusion and decision-making support. Results from past studies demonstrate that C-V2X PC5 prototype systems often exhibit effective range improvement and consistent packet reception under complex radio propagation conditions. The 5G NR Uu channels deliver the capacity to begin the conversation of more robust cooperative services. For example, iteratively distributing lists of cooperative perception messages, such as outputting CPM like object lists outside of a single intersection, distributing map and intent information, or even disseminating hazards over-the-horizon. When specifically considering infrastructure-assisted perception and highway merging, it simply represents a division of labor, as PC5 communication carries localized, time-critical advisory and handshake messages, while cellular path with MEC backstops the system with aggregate perception and market analytics, coordinated across multiple RSUs. These two communication paths enable deployments, in real time, to balance the immediacy of the communication, the coverage path if necessary and the system load, continuously maintaining reliable gap advisories and awareness of the situation as traffic density and message volumes increase.

In the end, C-V2X's dual-mode operation includes:

- PC5 (direct sidelink) communication: Allows low-latency V2V and V2I message exchange without routing through cellular base stations, essential for immediate safety applications.
- Uu (network-based) communication: Uses the cellular core network to support broader network services, including cloud access, traffic management, and non-safety services.

3.4.3 Quality of Service, Latency, and Reliability in Cooperative Automotive Communications

To maintain safe and optimal operation in autonomous and connected vehicle environments, guaranteed QoS is important. Real-time and safety-critical applications like collision avoidance or cooperative merging need a communication system that can satisfy the different performance benchmarks set by the applications. In turn, these application requirements will drive the development, standardization, and deployment of protocols like ETSI ITS-G5 or C-V2X.

With regard to QoS, latency must be kept extremely low to enable timely reactions to hazards, while reliability must remain exceptionally high to ensure uninterrupted situational awareness of the surrounding environment. Jitter, which is the variability in delay between packets transmitted from sender to receiver, must also be mitigated in order to support synchronized sensor fusion and decision-making. In addition to timeliness, networks must maintain a strong performance during peak traffic conditions or during large events where node density is substantially increased. Finally, cybersecurity is an essential component, as connected vehicles are susceptible to multiple risks (e.g., spoofing, denial-of-service). For that reason, it is necessary to employ authentication, encryption, and integrity checks.

Navigating these competing demands involves tradeoffs. Increasing redundancy may improve reliability, but it also may increase channel load. And increasing throughput may increase latency for safety-critical messages. For this, protocol design addresses these competing tensions in different ways. For example, ITS-G5 utilizes contention-based access with message prioritization. This approach allows safety messages to be prioritized, but when roadside message frequency approaches 100%, the performance of the channel deteriorates. And in the case of C-V2X, it uses scheduled resource allocation and adaptive coding to improve robustness to interference. Considerably, as C-V2X moves toward 5G, the performance becomes stronger with the use of ultra-reliable, low-latency communication and more bandwidth with edge computing.

Edge computing will be an important strategy for balancing these tradeoffs. By processing and fusing data at roadside units or base stations, edge nodes optimize the distance messages need to travel while reducing cellular service latency and jitter. Edge nodes can also transmit aggregated processing results to multiple vehicles nearby, freeing bandwidth for more important communications and improving the overall reliability.

In practice, systems with hybrid designs are rapidly gaining traction. For example, delay sensitive safety data may be sent using ITS-G5, while C-V2X is used for bulky, high-bandwidth applications. This is largely because of the computational/processing capabilities of ITS-G5 systems and, to a lesser extent, C-V2X, which could receive help from an edge process. In order to achieve operational dependability and performance, designers must also include traffic prioritization, separate safety and non-safety traffic, and add redundancy/monitoring approaches. Overall, the efficacy of cooperative vehicular systems does not rely solely on the theoretical performance of either ITS-G5 or C-V2X but rather the engineered trade-offs exercised. Balancing latency, reliability, scalability, and security at the forefront of providing communication infrastructure for experimental real-world, safety-critical autonomous driving.

Chapter 4

System Architecture and Concept

As stated above, the aim of this thesis is to extend the work of Oliveira 2023 and Ribeiro 2024 by integrating distributed perception capabilities into the existing co-simulation framework.

The goal is to develop a scalable, modular, and realistic testbed for distributed vehicular perception and coordination systems, with a particular focus on highway merging as a safety and efficiency use case. The proposed system architecture is based on three principles:

- Sensor and Actor Integration: Realistic simulation of roadside sensors, vehicles, and communications, reflecting real-world heterogeneity.
- Distributed Perception and Fusion: Enabling a RSU to process sensor observations in real-time, beyond the ego-vehicle boundary.
- Communication Simulation: Accurately modeling wireless, V2X communications, including latency, jitter, loss, and protocol stack behavior, with a focus on ITS-G5 and C-V2X.

This forms a multi-layered simulation architecture where data moves from the physical/sensor world, through distributed computation nodes, to the decision and control logics of AVs.

4.1 Application Scenario

The primary validation scenario focuses on highway merging actions, which represent one of the most safety-critical maneuvers in road traffic, requiring precise coordination between different lane vehicles and involving high-speed decision-making under time-critical constraints.

The scenario consists of a main highway lane with background traffic moving at a constant cruising speed and an acceleration lane from which the ego-vehicle attempts to merge into the main lane. Because merging with traffic is a dynamic process, the merging vehicle must reasonably predict the velocity that will allow it to safely and efficiently merge with the traffic flow.

To extend the perception capabilities of the ego-vehicle, the merging zone includes roadside cameras and a RSU. The cameras act as external sensing devices, detecting vehicles in both lanes, and transmitting this information to the RSU. While the RSU serves as an edge computing node, fusing camera observations and estimating available gaps in the main traffic lane. Based on this analysis, the RSU computes a recommended target speed for the ego-vehicle and transmits it through simulated V2X communication. The ego-vehicle then

adjusts its longitudinal velocity to align with the selected gap, ensuring a safe and efficient merge. Figure 4.1 illustrates the use-case scenario described.

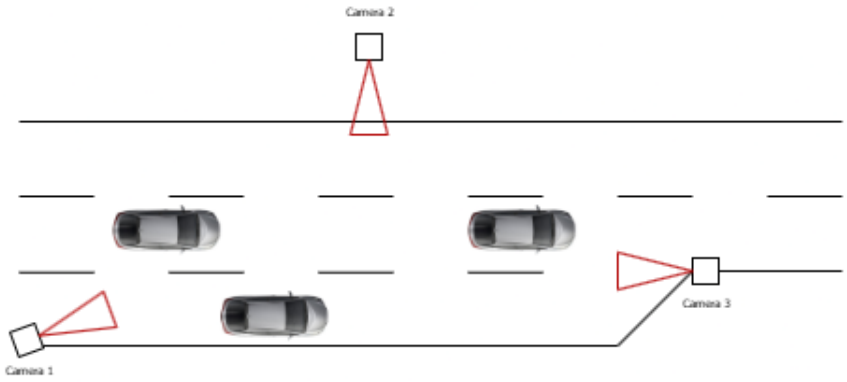


Figure 4.1: Highway merge scenario

This use-case scenario provides a concrete test case to assess the main goals of the proposed framework:

- The ability of roadside cameras to act as cooperative sensors by detecting and tracking vehicles across multiple lanes.
- The role of the RSU as an edge processing node, responsible for multi-sensor fusion, gap detection, and velocity guidance.
- The use of V2X communications to transmit guidance messages from infrastructure to vehicles.
- The capacity of the ego-vehicle to execute cooperative maneuvers by relying on distributed perception instead of purely onboard sensing.

In summary, the highway merging application situation offers a good example of the evaluation of both the perception and communication system components of the framework. The scenario illustrates how using external sensors can extend the onboard sensing horizon of vehicles, allowing for cooperative actions that can improve the difficult and unsafe nature of these scenarios. The scenario also takes those external sensors and acts as a stress test for the co-simulation framework to show analysis of whether communication networks and distributed sensing infrastructures satisfy the stricter timing requirements of a safety-critical context.

4.2 System Architecture

Building upon the application scenario, the system architecture will follow the existing approach demonstrated in figure 4.2 where ROS2 is the core of the system, acting as the middleware, responsible for the integration and communication of the sensing network. It integrates vehicle controllers, camera processing nodes, RSU logic, and launch configurations that organize the simulation pipeline. Gazebo simulates the physical environment, including the highway geometry, vehicles, and cameras. Providing realistic sensor outputs (RGB images) and vehicle dynamics. OMNeT++ simulates the vehicular communication stack, modeling network latency, packet loss, and message propagation between infrastructure and vehicles.

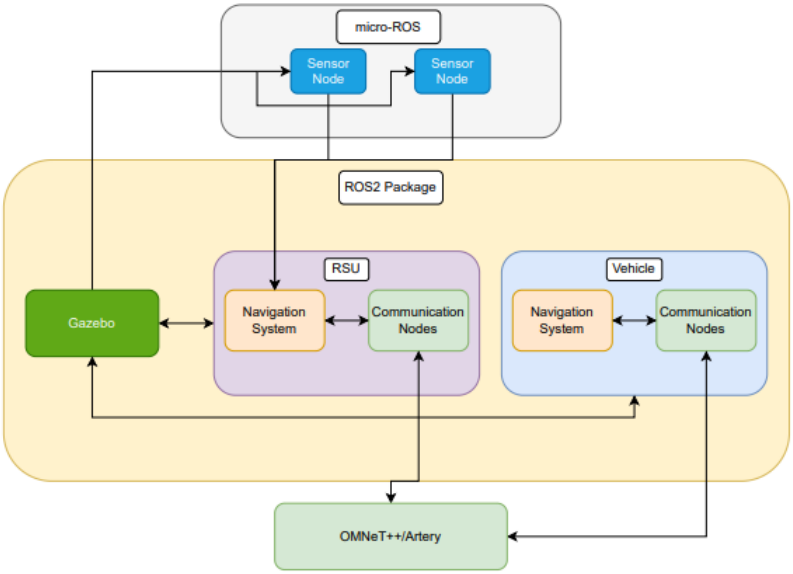


Figure 4.2: System Architecture (Oliveira 2023)

The system architecture developed by Oliveira 2023 is divided into three layers, namely:

ROS2 Package: This package wraps all the logic and information concerning ROS2 nodes, such as vehicles, sensors, Gazebo worlds, standard messages, and launch files. It acts as the core of the system, which will manage the interaction between different elements in the simulation.

micro-ROS package: It is responsible for the communication capability between micro-ROS nodes and ROS2 nodes. This enables this package to allow data transmissions from the roadside unit.

OMNeT++/Artery Module: A module integrated within the ROS2 package, offering vehicle-to-RSU communications.

ROS2 package ends up being the main devolvement focus because it includes implementations for sensor fusion, gap detection, and ego-vehicle control.

Chapter 5

Development

5.1 Overview

This chapter describes the implementation and integration of the functionalities in the co-simulation framework. The goal is to give a clear understanding of implementation and explain design choices grounded on prior work and the project constraints. The implementation is organized around three tightly coupled domains: simulation environment, perception, and RSU. The high-level implementation stages are:

1. Modular software - define ROS 2 packages and node responsibilities (vehicles, RSU, camera perception nodes).
2. Perception nodes - implement YOLO based ROS 2 nodes with acquisition, preprocessing, inference, and publishing of per-camera detection messages.
3. Projection & fusion - back-project detections using camera intrinsics/extrinsics and the ground-plane assumption, associate multi-view detections and compute trilateration.
4. RSU logic & gap selection - implement RSU node that determines vehicles location, computes gaps, applies safety & energy criteria, and communicates merge advisories.

5.2 Simulation Environment

To begin with, the development and validation of the proposed cooperative perception system required a realistic yet computationally efficient simulation environment. Building on the work of Ribeiro 2024 and Oliveira 2023, the environment leverages Gazebo for physics-based simulation, ROS 2 for middleware and node orchestration, and OMNeT++ for network simulation.

5.2.1 Map Implementation

From the perspective of Gazebo modeling, the world file (.world) does the specification of world physics, scene objects, and plugins. Models are provided as a URDF/SDF representation of geometry with links and joints. Contacts, inertia, and sensor plugins are associated with their respective link (or joint) to establish physical relationships and the generation of sensorial data. This separation of geometries, dynamics, and sensory allows the sensing configurations and scene geometry to be adjusted quickly without impacting navigation aspects.

Previously, new scenarios could not be created without a Gazebo world file and a binary occupancy grid file (.pgm). The world file specified the geometry of the 3D environment and was the file that Gazebo launched to produce the simulated world. The .pgm was another necessary file for planners in the Nav2 navigation stack, as it computed the environment in a way that could be used in global path planning. This dual dependency imposed significant constraints:

- Every new map required a full SLAM mapping process to generate the .pgm.
- Mapping large-scale environments was time-consuming and error-prone, often introducing inaccuracies that degraded navigation performance.
- The dependency limited flexibility, as switching to a different planning framework would still require .pgm compatibility.

To overcome these limitations, the work of Ribeiro 2024 restructured the simulation setup by decoupling world specification from navigation dependencies so that Gazebo functioned solely based on the .world file specification. This allowed the .pgm to be generated and utilized only when needed by Nav2 or other planning stacks. In practice, the simulation ran with the .world for visualization, sensing, dynamics, and traffic behavior, and when the planning algorithms evaluated global planning, it was instructing a .pgm that was generated independent from the instrumentation of the simulation itself. This led to quicker iterations of the scenario, experimentation, and decreased map generation overhead in large-scale scenarios with the geometry repeated.

This design offers several advantages:

- Faster scenario creation, since only the .world file is needed for visualization and vehicle simulation.
- Planner flexibility, allowing different planning frameworks to be tested independently without re-mapping the environment.
- Reduced manual effort, as large environments no longer need to be manually mapped before being usable in Gazebo.

In this work, the highway merging scenario (figure 5.1) comprising the main lane, acceleration lane, and roadside infrastructure was modeled exclusively via the .world file for perception and control experiments. This choice simplified the deployment of roadside cameras and the RSU and ensured consistent reproducibility of experiments. When evaluating global navigation behavior for the ego-vehicle, a .pgm was introduced as a separate artifact to support Nav2 without constraining the rest of the simulation pipeline.

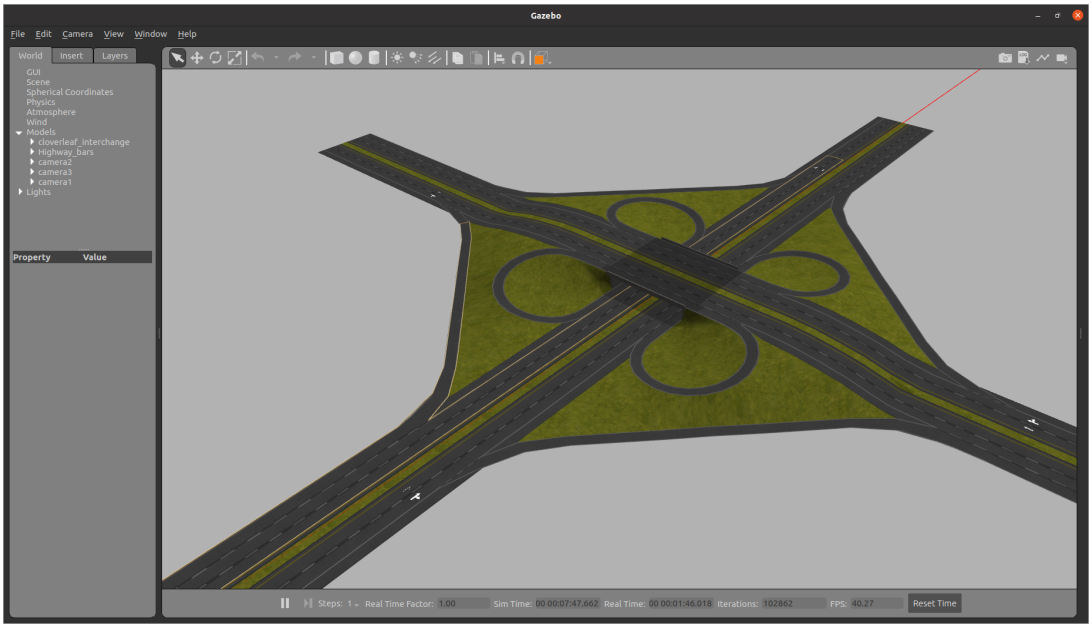


Figure 5.1: Highway scenario used in the simulation.

5.2.2 Vehicle Model

Realistic highway merging requires the presence of multiple vehicles to simulate real traffic conditions. The baseline Prius vehicle model available in the Gazebo distribution, while accurate, posed two major challenges:

- High computational cost. Each Prius spawns with a full Nav2 navigation stack and a complex Gazebo control plugin (Ackermann drive), making the simulation resource-intensive. Running multiple instances can quickly become infeasible on standard hardware.
- Unnecessary model complexity. The baseline Prius includes detailed meshes and textures for interior components that are irrelevant for perception or control in highway traffic scenarios.

While accurate, this model results in high computational cost per vehicle and limits traffic density on standard hardware. Moreover, intricate visual meshes and interior details are unnecessary for the perception and control tasks evaluated in this thesis. To address these limitations, this work adopts and extends the approach of Ribeiro 2024 with a more simplistic Prius vehicle model (figure 5.2). The "Prius-Lite" uses three design methods to relieve the computational overhead. First, it significantly minimizes the ROS 2 node footprint by eliminating the per-vehicle Nav2 stacks, leaving full navigation only to the ego-vehicle itself. Second, its Ackermann drive plugin was replaced by Gazebo's Planar Move plugin, which applies linear and angular velocities directly to the base link via geometry_msgs/Twist. For linear traffic flows along highway lanes, the Planar Move plugin is sufficiently realistic with regard to kinematics and uses relatively fewer computational resources. Third, the model removes all unnecessary interior meshes and textures that added to rendering, and thus GPU load is reduced under multi-vehicle operation.

While potentially less realistic, these simplifications maintain important kinematic behavior and allow for large-scale traffic simulation of multiple agents in parallel. Under the assumption of linear movement, which is common in highway scenarios, the loss of some front steering wheel realism is acceptable. If an experiment requires high-fidelity vehicle dynamics (for example, aggressive maneuvers), the original model can easily be swapped back into the simulation environment with minimal changes to the ROS 2 interface. Notably, this approach was robust to Gazebo physical perturbations (i.e., gravity, drag, and contact dynamics) after the correct inertial parameters were tuned to prevent drift at high velocities.

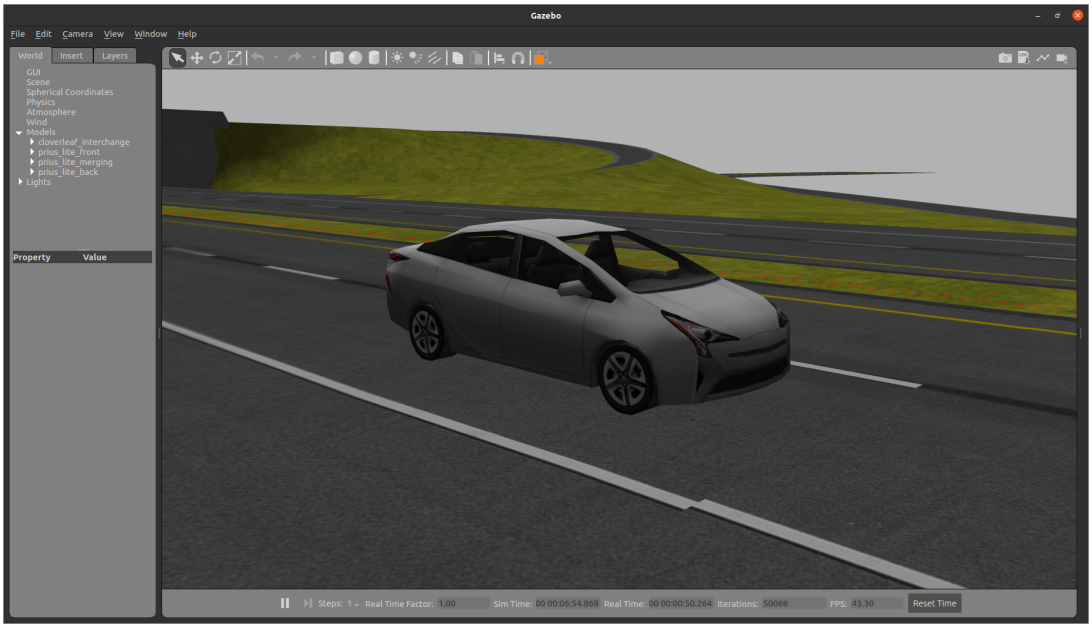


Figure 5.2: Prius Lite model in Gazebo.

5.2.3 Camera Modeling in Gazebo

The fixed roadside cameras in the co-simulation framework are represented as Gazebo sensor plugins, which provide a software abstraction for the camera behavior and expose the camera model parameters through the Simulation Description Format (SDF). Each virtual camera is characterized by both intrinsic parameters (resolution, focal length, field of view, and distortion model) and extrinsic parameters (pose relative to the world frame). This specification enables the simulator to generate photorealistic RGB image streams that simulate real traffic surveillance devices. The cameras are configured to publish RGB image streams via ROS 2 topics at a frame rate of (10–20 Hz).

The camera subsystem within Gazebo allows for explicit specification of projection matrices and distortion coefficients. Because of the defined geometric representations, the calibration profile obtained from a standard tool such as OpenCV or MATLAB is reproduced accurately. In addition to optical configuration, the sensors in the simulator's sensor abstraction layer are strictly aligned with the simulation time. When a frame is captured, it is tagged with the simulation clock, and the camera's alignment with the vehicle's ground-truth states is perfectly synchronized with the communication events. This is important to achieve coherency in time (especially to successfully remove time offsets) during multi-view association for trilateration. Small offsets in time produce large errors in the projected ground-plane object location and will be especially problematic when there is a need for confidence during

cross-camera fusion. Thus, the camera data pipeline produces time-consistent inputs for multi-camera fusion and trilateration during the process, thereby improving projection bias and stabilizing localization in the later steps.

5.2.4 Integration and Deployment

Although the previous subsections explained the components of the simulation environment on an individual level, they must now be integrated into a single coherent and reproducible launch and orchestration workflow. This design was envisioned for the sake of consistency, modularity, and scalability across experimental runs to ensure reproducibility. Each experiment needs to initialize the same world, vehicles, and sensors the same each time to compare experimental results correctly. At the same time, flexibility is needed in the simulation while changing parameters like traffic vehicle density or camera placement without having to write rules or duplicate large sections of the launch and orchestration workflow.

In practice, the environment is instantiated through a hierarchy of ROS 2 launch files, which organizes the simulation into layers of steps. At the bottom level, the Gazebo server is started with the specified .world file, which describes the highway merging scenario. Launching the world this way makes sure that the physical layout of the scenario will be consistent across experiments, helping ensure a stable basis for testing perception and control algorithms.

Vehicles are spawned after the world initialization phase is completed. Each vehicle will have a unique namespace (e.g. /prius1, /prius2) to avoid topic conflicts and to facilitate independent control. The spawn process is parameterized, meaning that the vehicle's initial position and velocity can be defined prior to spawns to allow for systematic testing of, for example, changing the positions of vehicles or changing the number of vehicles. In the case of the ego-vehicle, localization is initialized at the front of the acceleration lane to represent a realistic onset condition, and background vehicles are spread out down the main lane. The distinction made between the ego-vehicle and the background vehicles reflects both aspects of simulation: the ego-vehicle being exposed to environments that may have variability, but their features remain constant.

The vehicle's control interfaces further illustrate the tradeoff between realism and efficiency. Background traffic vehicles are directly controlled with simple velocity profiles published to the /cmd_vel topic. This plugin applies linear and angular velocities to the base link of the vehicles. In contrast, the ego-vehicle is running on a complete control pipeline. The ego-vehicle will subscribe to the target velocity from the RSU and execute a PID based longitudinal controller.

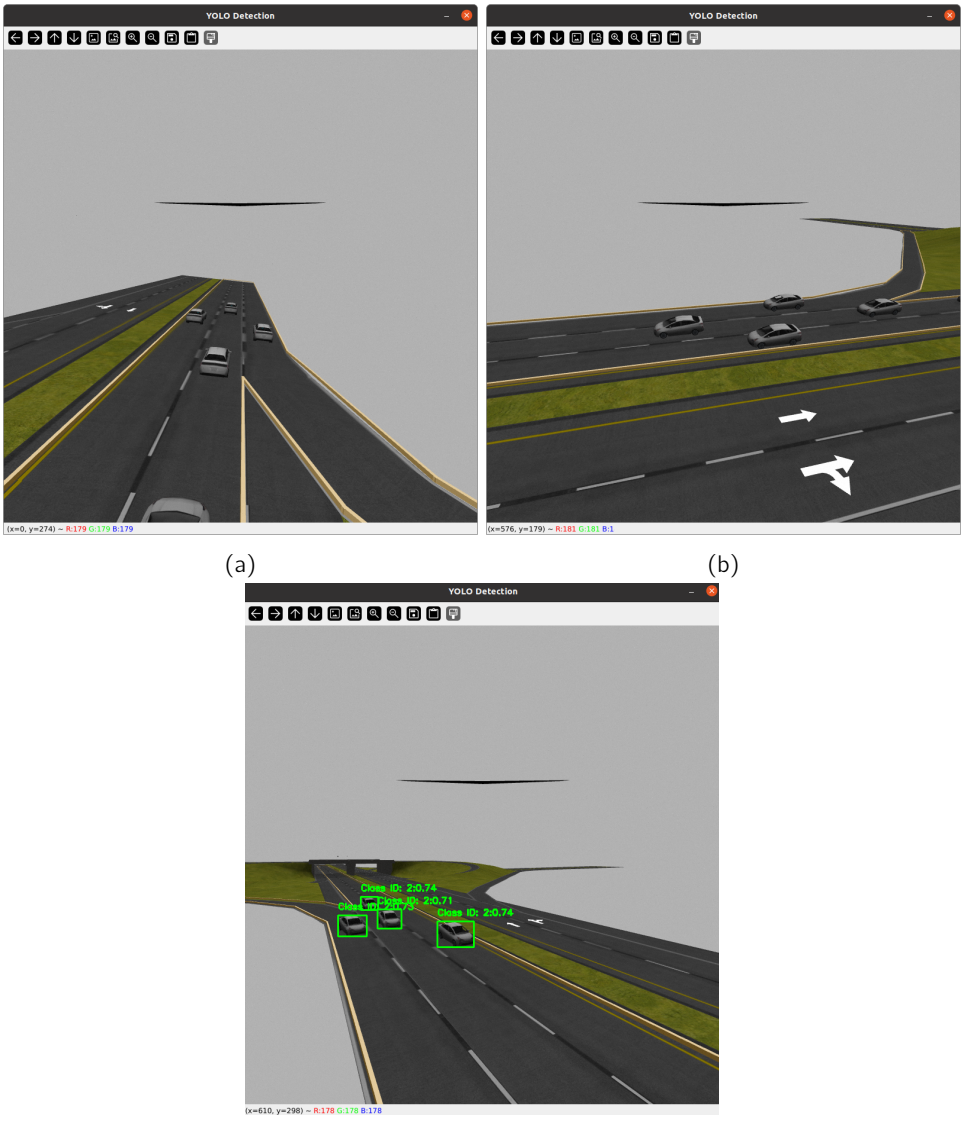
The sensing infrastructure, which includes roadside cameras and the RSU, is integrated with dedicated ROS 2 nodes. Each camera is instantiated with its own launch file that defines both extrinsic parameters (pose in the world frame) and intrinsic parameters (focal length, field of view, and distortion model). This launch file provides Gazebo's sensor abstraction layer with the necessary parameters to instantiate RGB image streams. The RSU, implemented as a separate computational node, collects multi-camera detections and then applies higher-level reasoning, such as multi-view association, trilateration, and gap selection, to send the target velocity suggestion back to the ego-vehicle. This division of perception and decision-making into nodes supports the architectural goals of distributed perception and edge computing, where the roadside infrastructure supports the vehicle by supplying some additional sensing and computation beyond the capabilities of the vehicle's sensors and processing onboard.

Overall, this layered launch framework creates a modular, extensible, and reproducible simulation practice. The framework has the benefit of dual efficiencies and realism through the various decoupling of world specification from dynamic agent behaviors, the simplification of traffic control while also maintaining ego-vehicle fidelity, and the parameterization of all the major components of the setup. With all key parameters being externally configurable to support rapid scenario iteration and analysis. These include: the main-lane reference speed, the merge point position, control frequency, and thresholds such as the minimum safe gap distance or maximum acceleration. This parameterization facilitates reproducible experiments and fair comparisons across runs. In short, the architecture is capable of evaluating distributed perception and cooperative merging strategy but can also function as an environment for forty behavioral investigations, including multi-agent coordination, reliability of communication, and the scalability of the edge-assisted cooperative approach.

5.3 YOLO-Based Camera Detection

5.3.1 Camera Placement and Coverage

The three cameras used for detection were strategically placed along the merging zone (figure 5.3): two cameras providing longitudinal coverage before and after the defined merge point, and one lateral camera providing transverse lane coverage.



(a) (b)

(c)

Figure 5.3: (a) Camera1 feed. (b) Camera2 feed. (c) Camera3 feed.

This geometry ensures that the cameras have overlapping fields of view (FoV) that cover the different lanes and the critical merging region. The placement scheme also attempts to create redundant viewpoints in order to minimize occlusion and maximize detection recall rates for high-volume traffic, where vehicles are more likely to occlude each other. The overlapping field-of-view design will also enable multi-viewpoint detection fusion and triangulate the positions of vehicles with geometric constraints based on the array of cameras.

5.3.2 YOLO Detection Pipeline

Each camera in the system operates as a standalone ROS2 node - Camera1, Camera2 and Camera3. This design creates detection functionalities that are isolated and scalable, making it easier to deploy multiple cameras as part of the simulation deployment.

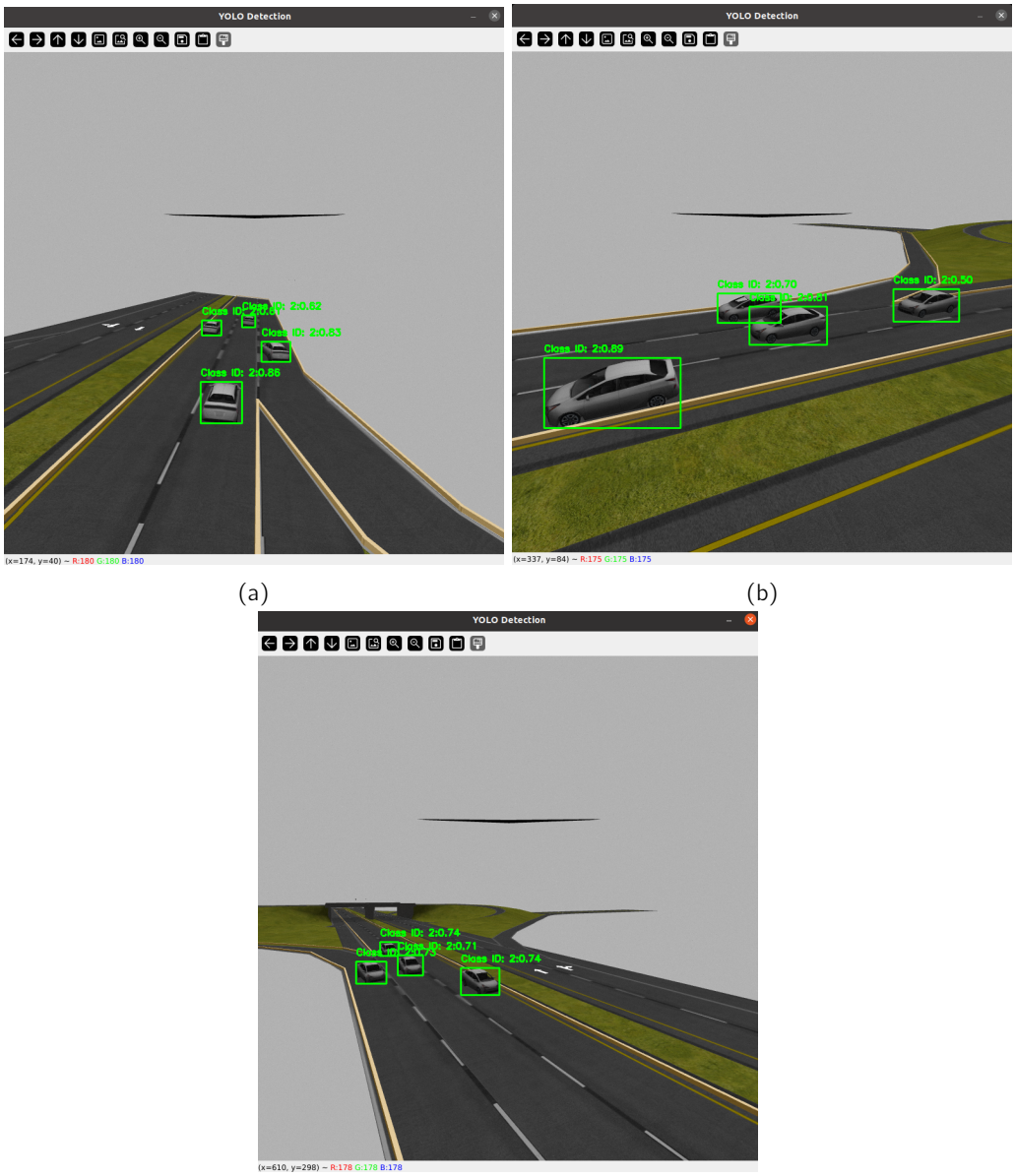
The pipeline begins when images are received and published from the simulation. Each roadside camera publishes its frame to a dedicated ROS 2 topic (e.g., /camera1/image_raw). The detection node subscribes to its respective image topic using BEST_EFFORT QoS profile, which is suitable for low-latency and high-frequency sensor streams.

After receiving a new image message from the topic, the processing of this image proceeds through a conversion step using the cv_bridge package, which translates the sensor_msgs/Image message to an OpenCV compatible array. The frame is to be compatible with the YOLO model to run properly.

The converted frame is processed by the YOLOv8 model for object detection, which is contained within the Ultralytics API in Python, to identify the bounding boxes, class labels, and confidence scores for every object detected in the image. To reduce computational overhead, inference is performed with a confidence threshold, ensuring that uncertain detections are filtered out before publication. Also, in this work, the detections are adjusted to only track and label vehicles, since they are the only agent of interest and relevance.

For each valid detection (figure 5.4), bounding box coordinates are packaged into a ROS 2 message structure, timestamped using the simulation clock, and tagged for consistency across cameras. In the end, each camera detection is published on the respective ROS 2 topic (eg, /camera1/detection_coordinates). This message stream includes bounding box-derived detection data (pixel coordinates, class labels, and confidence scores) and is made available for higher-level perception modules.

Maintaining independent detection nodes for each camera (e.g., camera1, camera2, camera3) provides modularity, robustness, and scalability because each detection process runs in parallel and no camera is dependent upon another. This design is valuable in larger scale cooperative perception scenarios when dozens of roadside sensors together collectively contribute to form a unified situational awareness view.



(a) (b)

(c)

Figure 5.4: (a) Camera1 detection. (b) Camera2 detection. (c) Camera3 detection.

5.4 RSU Deployment

The RSU acts as the central decision-making component of the framework. Its responsibilities include multi-camera fusion, vehicle clustering, gap analysis, and maneuver recommendation for the merging ego-vehicle.

5.4.1 RSU as Cooperative Perception Node

The RSU acts as the central infrastructure node responsible for collecting camera data and supporting the merging decision. In the co-simulation, the RSU is deployed as a ROS 2 node. Its main functions are:

1. Receive vehicle detections from the cameras.
2. Determine the localization of each vehicle on the highway, fusing information from the different cameras.
3. Maintain an updated global object list with the position and velocity of all vehicles in the merging zone.
4. Compute available gaps in the main lane traffic flow.
5. Select the most appropriate gap for the merging based on safety and efficiency criteria.
6. Transmit the required maneuver velocity to the merging vehicle via the V2X communication channel.

5.4.2 Multi-Camera Fusion and Vehicle Clustering

To localize vehicles in the RSU implementation, an approach is designed that fuses multiple camera detections based on a ground-plane assumption and trilateration. The approach starts when each camera publishes the detections (the 2D bounding boxes after the YOLO inference) representing the bottom centers of vehicles in image space to the RSU. The RSU then collects all of the observations per camera, projects them into a common world (map) coordinate system assuming that vehicles are located on a flat road surface ($Z = 0$), associates all of the detections that belong to the same vehicle (according to mapping detections into adjacent cameras), and ultimately fuses the detections into a single, more accurate localization.

Ground-Plane Assumption and Projection

Vehicle detections must be converted from 2D image coordinates to a common global reference frame to enable multi-camera fusion. This conversion leverages the ground-plane assumption, which asserts that all vehicles are in contact with a planar road surface ($Z=0$ in world coordinates). Since the scenario is a highway environment, this approach can be used as an adequate approximation.

The process of transforming the coordinates occurs in two steps:

Back-Projection: The pixel coordinates of the footpoint of each detection (typically the bottom center point of the bounding box) are back-projected into a 3D ray by leveraging the intrinsic matrix K of the camera and the extrinsic parameters [R|t]:

$$r = R^{-1}K^{-1}u \tag{5.1}$$

where u represents the homogeneous pixel coordinate.

The implementation normalizes the ray direction and applies rotation matrices derived from Euler angles (roll, pitchl, and yaw) to transform from camera coordinates to world coordinates.

Ray-Plane Intersection: The back-projected ray is intersected with the ground plane (z=0) to yield world coordinates (X,Y,0).

This projection allows each detection to be expressed in a shared top-view ground-plane coordinate system, facilitating trilateration across multiple cameras. Similar inverse perspective mapping techniques are widely employed in multi-camera cooperative perception systems.

5.4.3 Trilateration and Multi-Camera Fusion

Once vehicle detections from individual cameras are projected onto the global ground plane, they must be associated and fused to obtain accurate and reliable position estimates. Multi-camera fusion is necessary because a single camera may yield noisy or partially occluded detections, while overlapping views allow redundancy and higher accuracy.

Detection Association

The first step is to determine which detections from different cameras correspond to the same physical vehicle. This is accomplished by:

- Proximity-based matching: Candidate detections are clustered based on Euclidean distance in the ground plane, using a threshold.
- Temporal consistency: Only detections within the same simulation timestamp (synchronized across cameras) are considered.
- Confidence weighting: When multiple candidate matches exist, the detection with the highest YOLO confidence score is given priority.

This yields groups of detections that represent the same object across cameras.

Trilateration Method

Once detection clusters are formed, the RSU computes an optimal fused estimate of each vehicle's world position. Because the back-projected rays from different cameras generally do not intersect perfectly due to measurement noise, a least-squares trilateration approach is used: by finding the 3D point (constrained to z=0) that minimizes the total squared distance to all rays in the cluster.

When all three cameras provide observations, this least-squares optimization yields a robust intersection point. In cases where only two cameras detect the vehicle (e.g. due to partial

occlusion), the system relies on a weighted midpoint between the two projected positions, weighted by detection confidence. If only a single camera detects it, the single-camera projection is used but flagged with lower confidence or reliability.

This hierarchical fusion strategy ensures continuous tracking even under partial visibility, which is common in highway merging because vehicles may occlude one another. The fused (X,Y,0) position, along with a velocity estimate (from successive frames), is stored in the RSU.

5.4.4 Gap Computation in the Main Lane

At this stage, once the vehicle positions are available, the RSU sorts the vehicle positions along the x-axis. The goal is to determine, which gaps can be considered feasible opportunities for the merging. The gaps between consecutive vehicles are computed as:

$$gap\_size = x_{front} - x_{rear} \tag{5.2}$$

For the current phase, the system adopts a pragmatic approach, with a spatial gap analysis built up with a constant velocity projection to the merge point. Concretely, detections are sorted by longitudinal position X in ascending order, and adjacent pairs define candidate gaps. For each pair (front, rear), the system computes the current gap size as the rear-to-front spacing along X.

Under a constant main-lane speed assumption, the positions of the front and rear vehicles are propagated forward by t_{merge} to estimate their future longitudinal positions at the merge instant. In the present implementation, the system assumes equal velocities for those main-lane vehicles, which keeps the predicted gap equal to the current gap size. While this simplifies selection, it is structured for incremental refinement, once per-vehicle velocities are estimated, unequal-velocity projection naturally replaces the equal-velocity simplification.

Each gap record produced by this stage contains: the current size, the projected size (equal to the current size in the equal-velocity case), the gap center, the future positions of the rear and front vehicles at t_{merge} and the time until the gap reaches the merge.

5.4.5 Gap Selection Criteria

After determining the available gaps, the next step is to evaluate them. Selecting among viable gaps requires balancing safety and efficiency. In this case, two selection paradigms were considered. A safety-first paradigm defines a minimum safe distance to both the leading and trailing vehicles at the merge instant as hard constraints, eliminating any gap that cannot guarantee those margins. On the other hand, an energy-first paradigm optimizes for minimal required speed change (and thus minimal acceleration effort), which enhances comfort and energy efficiency but still must respect safety bounds. Multi-objective selection can be implemented either by combining safety and efficiency into a single cost or by adopting a filter by safety constraints first, then optimizing efficiency within the feasible set.

The present system adopts an incremental approach consistent with the current sensing and estimation fidelity. It first filters by viability (as defined by the spatial criterion above), then chooses the gap that minimizes the required deviation from the ego's current speed to synchronize arrival at the merge point with the center of that gap. This procedure produces energy-friendly choices without yet implementing full headway constraints.

Operationally, for each candidate gap, the RSU computes the longitudinal distance from the gap center to the merge point (d_{gap}) and divides by the main-lane speed (v_{main}) to estimate the time at which that gap center will align with the merge location. The ego's target longitudinal speed v_{target} is then computed as the remaining distance to the merge point divided by this time:

$$v_{target} = \frac{d_{merge}}{(d_{gap}/v_{main})} \tag{5.3}$$

Then it is clamped to the feasible range [0, v_{max}] to respect actuation and speed limits. When multiple gaps yield feasible targets, the system selects the one closest to the ego's current speed to minimize acceleration effort. The predicted safe gaps to the leading and trailing vehicles at the merge instant are computed explicitly. Any gap failing to meet the safe considerations is discarded before applying the energy criterion. This decomposition preserves safety as a hard requirement while still favoring energy-efficient maneuvers among the safe options. If multiple gaps satisfy both constraints, the RSU selects the one closer to the merging point to reduce maneuver uncertainty.

5.4.6 Control Loop and Target Speed Publication

The RSU is the component that closes the cooperative perception and decision-making loop by publishing a recommended target speed to the ego-vehicle at a fixed control frequency. Two potential actuation strategies were considered. The first strategy, publishing target speeds directly to the ego-vehicle, while tracking and maintaining closed-loop operation (such as compliance with acceleration limits), would be left to the ego-vehicle's onboard longitudinal controller. The second option would attempt to transfer more accountability over to the infrastructure by relaying messages containing overall trajectory goals or desired maneuver actions. Ultimately, because the system is cooperative, it was decided to use the first approach. That is, the RSU maintains accountability for the assessment of the situation and selection of a gap, while the ego-vehicle remains accountable for low-level actuation.

In the deployed system, the RSU computes the target velocity and publishes it as the linear.x field of a Twist message, using the ROS topic '/ego_vehicle/target_speed'. The ego-vehicle subscribes to this topic and applies its onboard longitudinal controller (designed as PID with feedforward compensation) to track the suggested speed. By using this architecture, the ego-vehicle's local controller can maintain compliance with acceleration at the same time that the RSU works to deliver assistance in a cooperative maneuver.

To facilitate simulation, the control loop has been expanded with an additional component that regulates the merging maneuver. While the RSU continuously sends a longitudinal velocity command to the ego-vehicle through the linear.x field of the Twist message, once the ego-vehicle reaches the merging point, the RSU also begins to send a lateral velocity command through the linear.y field of the same Twist message. The lateral velocity command smoothly moves the vehicle into the adjacent traffic lane, effectively simulating the process of merging. When the vehicle is already in the lane, the lateral velocity goes back to zero, and the vehicle assumes the constant velocity of the main lane for the longitudinal motion.

In the current prototype, ego-vehicle position and velocity were originally provided as placeholders. In the implemented system, ego-vehicle position and velocity comes from a subscription to the ego's odometry stream to calculate the distance to the merging point and to receive live velocity data. If a viable merging gap is not available at any moment, the RSU

will provide a conservative approach and recommend a reduced target velocity. This delayed target velocity will help ensure that the ego-vehicle does not arrive early to the merging point and increases the opportunity for new gaps to become available.

5.4.7 Communication of Gap Decision

Once the RSU selects the optimal merging gap, it transmits a message to the ego-vehicle via the OMNeT++ simulated V2X channel. In the current implementation, the message is simplified to contain only the recommended target speed that the ego-vehicle should adopt in order to synchronize with the main-lane traffic at the merging point.

This design choice emphasizes efficiency and minimal communication overhead, ensuring that only the most critical information is shared in real time. The ego-vehicle's onboard controller uses this target speed to regulate longitudinal motion and align with the dynamics of the selected gap. By focusing solely on the velocity recommendation, the RSU reduces complexity while still enabling safe and cooperative merging.

5.4.8 Extensions and Next Steps

The present realization forms a good basis for gap selection with infrastructure augmentation with scope for necessary improvements. To begin with, per-vehicle velocity estimation needs to be integrated by incorporating temporal tracking into augmented fused detections. This will eliminate the equal velocity hypothesis and permit unequal speed projection of gap formation, with improved prediction accuracy in real-world traffic.

Second, safety constraints need to be elevated to explicit, time-dependent verification at the merge instant, computing progress to both leading and trailing vehicles in the candidate gap. Imposition of a minimum headway (e.g. 2.0 s) ensures that accepted gaps are compatible with time safety margins, rather than relying on fixed distance criteria.

Third, if there are multiple gaps satisfying both safety and energy constraints, then the decision can be made on which reduces time-to-merge jitter while maximizing decision confidence, for example, by preferring gaps supported by more sensors and more recent updates.

Finally, the advisory can be represented in a formal cooperative perception message that includes a gap identifier, identifiers (or coordinates) of the lead and trail vehicles defining the gap, the recommended target speed, the expected time to merge, and an expression of confidence depending on sensing coverage and recency.

Chapter 6

Evaluation and Results

6.1 Overview

This chapter provides the experimental assessment of the proposed cooperative perception and co-simulation framework. The objective of this chapter is to demonstrate that the proposed system can accurately detect and localize vehicles, provide gap calculations for highway merging, and maintain real-time communication performance in various network conditions and sensing conditions. In essence, this chapter assesses the system's perception quality, scalability, and maneuver support capabilities as outlined by the following research questions:

- Perception Quality - How accurately can the system detect and localize vehicles using multi-camera YOLO detections and trilateration?
- Scalability - How does system performance evolve when scenario complexity increases?
- Maneuver Support - Does the RSU compute safe and feasible merging gaps under varying conditions?

6.2 Experimental Setup

The baseline scenario considers a highway merging scenario, where an ego-vehicle must safely and efficiently merge into the main lane populated by multiple background vehicles traveling at constant speed. The standard configuration uses three fixed roadside cameras deployed along the highway with overlapping fields of view to provide the visual sensing infrastructure. These cameras run YOLO-based detection and feed their outputs into an RSU that applies trilateration to estimate the vehicle positions under a ground-plane assumption. The RSU then aggregates fused perception data, computes the list of available merging gaps, and communicates this information to the ego-vehicle through OMNeT++.

Alongside the conventional set of three cameras, the system is evaluated using both one and two cameras to investigate the influence of sensing redundancy. The density of the main-lane traffic and the speed of the vehicles will also be manipulated. Varying these parameters enables the further evaluation of the scalability and robustness of the perception system in the presence of denser and faster traffic. High traffic density increases the occlusion of objects of interest and increases the load on the perception pipeline (more objects to be detected and tracked), while greater speed reduces the time any object remains in the view of the camera and increases the performance threshold for real-time processing.

To test the scalability of the solution, different main lane traffic densities are simulated to explore timing sensitivity, occlusion effects, and load impacts.

In every run the following metrics are considered:

- Ground-truth vehicle states from Gazebo.
- Detection outputs stored with timestamps and IDs.
- YOLO inference time per frame and hardware utilization.
- Fused localizations recorded after trilateration.
- Merge outcome and target velocity.

6.3 Results and Discussion

6.3.1 Multi-Camera Detection and Localization Evaluation

Each roadside camera was firstly evaluated independently to quantify its detection rate and localization accuracy relative to the ground-truth vehicle trajectories. This measures the performance of single-camera detection and localization, providing a point of reference for determining performance improvements through multi-camera fusion. The actual positions obtained from Gazebo are the baseline to which the output from each roadside camera is compared, evaluating detection rate (recall), the accuracy of position estimates, and any biases or patterns of error and performance in estimating distance or position.

The localization errors for detected vehicles across all angles of view from the cameras ranged typically from approximately 0.29 to 0.33 m (table 6.1), with median errors especially usually less than 0.26 m. Even under difficult conditions, errors rarely exceeded 2–3 m. This indicates that while every monocular camera by itself does indeed possess limited accuracy in depth measurement, the errors are usually limited to a couple of centimeters. These examples provide baseline performance details and inform the limitations of using a single perspective: a monocular camera cannot make any kind of direct depth measurements, nor does it have the benefit of depth ambiguity by clearly seeing (or not seeing) the presence of occluding objects. Thus, to address any shortfalls, and more importantly, to improve recall when possible through occlusion itself or absence of presence, multiple cameras are useful: overlapping angles of view can greatly broaden recall by seeing around blocking objects and also improve localization through fusion of multiple views.

Table 6.1: Localization error for each camera..

Camera	Mean Error (m)	Median Error (m)
1	0.29	0.22
2	0.33	0.26
3	0.30	0.24

The precision of localization for Camera 1, which viewed the merge junction, produced an average error of 0.29 m and a median error of 0.22 m, providing support for the claim that most detections were closer than a couple centimeters to the true location. A small observable bias in its estimates for X and Y positions was noted: the estimate for X position was consistently underestimated, giving a position that was somewhat behind the detected position of the vehicle, while the Y position estimates were slightly overestimated, which, in combination with the underestimation in X position, generally gave lateral biases. The X and Y biases are consistent with the angled geometry that was positioned quite close to the merge junction and was detecting vehicles on the road at an angle. Typical position errors for vehicles that were mostly in the camera's central field of view tended to be between 0.1 m and 0.5 m, aside from a few outliers (2.5 m around a distant target vehicle). Camera 1 had a high recall for merging vehicle detection, while farther away (main-lane) vehicles were detected less frequently owing to partial occlusion at the convergence of the vehicles and limited exposure time.

Camera 2, held in a higher position and positioned at a steeper downward angle, achieved the highest mean error (0.33 m) and median (0.26 m). Its localization was nearly unbiased for both axes. Camera 2 had a lower recall compared to Camera 1. However, Camera 2 had a sideways coverage of main-lane traffic and maintained stable detections of vehicles approaching the merge point.

Camera 3, which was oriented downstream and directed back towards the merge, demonstrated an overall mean error of 0.30 m and a median of 0.24 m, which was similar to the other two cameras but had somewhat greater variance. Camera 3 had a slight systematic overestimation of longitudinal position and lateral offset. Near the camera, or toward the edges of the frame, performance dropped severely, resulting in occasional error estimates exceeding 3 m. However, for mid-range targets, Camera 3 held performance estimates below 0.4 m of accuracy.

The results validate the assertion that each individual camera contributes to the overall situational awareness for the highway-merge problem in a unique but complementary way. This rationale reflects the fundamental idea behind a distributed perception framework: combining camera observations should vastly improve the overall coverage and robustness of vehicle detection. Given that all three have different viewing angles, a vehicle missed by one is likely to be detected by another. The proof in numbers also reinforces why a cooperative solution is needed. A single monocular view does not equate to a complete reliable perception within a merge scenario.

6.3.2 Multi-Camera Fusion and Trilateration-Based Localization

By utilizing the three cameras located at the roadside, the system follows a process of spatial sensor fusion via trilateration to provide a more precise and complete vehicle localization than the use of any single camera can offer. Each YOLO detection, as a 3D ray based on the pixel coordinates of the detection and known camera calibration, is placed into the global coordinate frame. At each time sample, if the cameras detect the same vehicle, the RSU computes vehicle position as the point on the ground plane that minimizes the squared distance to all of the camera rays - a least-squares trilateration solution. If only one camera detects a vehicle, its single projected position is used temporarily until other detections are provided. The fused positions would then be compared to ground truth trajectories to provide a measure of improvement from the cooperation of the multi-camera system.

Quantitatively, fusion resulted in more accuracy and coverage improvements compared to single cameras. Average localization error was cut to 0.18 m. Indicating that most fused estimates are within twenty centimeters of the true vehicle position. The results show that trilateration dampens the outlier projection errors from individual cameras. Equally importantly, overall detection recall is almost 100% in the merging zone when all cameras are fused. This confirms that overlapping fields of view successfully eliminated blind spots and reduced occlusion effects to give a near-continuous perception of the entire merge area.

In addition to the numerical gains, several qualitative behaviors help explain why multi-camera fusion is so effective.

Complementary Fields of View: Each camera produced detections involving different perspectives, and the fusion process blended these partial views into a single global scene. For example, when Camera 1 lost sight of a vehicle, it often happened that Camera 2 or 3 was still able to observe the vehicle. Therefore, every vehicle was observed by at least one sensor.

Error Compensation: Systematic positive bias from one camera was compensated by one or both of the other cameras. For example, Camera 2 would show a top-down view and slightly compensate for the Camera 3 distance estimation to provide a balanced fused position. This form of cross-camera constraint stabilized the depth estimation and reduced lateral drift and contributed to smoother trajectories and a more accurate final position.

Temporal Continuity: The overlapping coverage allowed for smooth hand-offs between the cameras as vehicles moved through the coverage of the scene. Even if one view missed the detection of a vehicle for a very brief moment, another camera viewed the vehicle nearly simultaneously, and the continuity of tracking remained. This directly accounts for the recall result increasing, because far fewer vehicles were lost between observations.

The trilateration method of camera mitigates the random element of position error, due to the geometric constraints imposed by each independent view of the scene. In the case of fused estimates using synchronized cameras with overlapping views, the result is much more precise and less noisy than either camera's single projection. Thus, trilateration produces the significantly improved localization found in multi-camera systems. Trilateration is sensitive to three sources of error: (1) bounding-box bottom estimation, the ground contact proxy; (2) extrinsics/intrinsics of the camera and calibration accuracy; and (3) timing and latency mismatch across the sensors and RSU. In this implementation the bounding-box base is a reasonable approximation. However, partial occlusions or misalignment of box bottoms produce biased rays. When compared to the timing skew, these biased rays can lead to

error accumulations and localization outliers, necessitating confidence weighting and outlier rejection heuristics in the fusion stage.

When the three cameras were simultaneously able to see the same vehicle, the accuracy of the fused position was consistently high. This result conveys the value of cooperative sensing: three traditional monocular cameras, once geometrically fused, behave analogously to a single, high-fidelity 3D sensor. The RSU was able to reconstruct a common model of the environment closely reflecting Gazebo ground truth data by synchronizing multiple camera nodes. This was done in an object-level data sharing level, rather than using raw video streams, each camera only communicated detection messages.

6.3.3 Detection behavior and sensing redundancy

The detection efficacy of the system was largely mediated by the number of active cameras and their geometric disposition. When a single camera was utilized, the system was very susceptible to occlusions and perspective limitations. For example, vehicles that were occluded by other vehicles while the camera was at an extreme angle would be missed or detected with low confidence. Detections that were missed resulted in lower recall rates and potential inaccuracies in localization, as detections would sometimes disappear when vehicles were no longer in the optimum field of view.

The introduction of a second camera with overlapping coverage subsequently led to greatly improved detection efficacy due to the additional perspectives that reduce occlusions. Overall, recall increased significantly between one camera and two cameras, as most detected misses were recovered. The inclusion of a third camera more effectively reduced false negatives, particularly when positioned at an alternate angle or elevation that provided better visibility of the merging zone.

From an information processing viewpoint, each additional camera brought an increase in processing load. The YOLO inference time per image was fairly stable, but given the pre- and post-processing demands, message generation, and fusion load, the total system latency increased as the number of cameras and detected vehicles increased. In heavy traffic scenarios, the increase in the number of detections per frame increased hardware utilization.

In summary, sensing redundancy improved detection robustness in a variety of ways, including covering occlusions, while also increasing recall. However, as the cost to perception was similar to the increase in detections, there is a trade-off between perception scalability and hardware constraints, essentially preserving system responsiveness.

6.3.4 Effect of traffic density

Adding vehicles impacted perception in two primary ways: occlusions and compute load. More cars led to occluded views more frequently, as vehicles often obstructed one another from the view of a single camera. This led to decreased recall and increased false negatives for that camera as well. In high-density runs, some vehicles were simply not visible or only detected for a short duration. However, the multi-camera fusion reduced this degradation to a large degree. Because overlapping fields of view existed, occluded vehicles could still be observed by a different camera. Thus, the overall recall and localization accuracy was stable during moderate traffic scenarios. Only in extremely dense conditions, when all cameras had overlapping occlusion, did perception accuracy significantly degrade.

The second impact was on computational and communication performance. As the vehicle count increases, each camera produces more detections per frame, increasing RSU processing and network transmission demands. YOLO inference time slightly increased, and the CPU load on the RSU increased due to fusing additional detections. Under heavy load, this could cause the system to lag.

In summary, moderate density increases were well tolerated, with redundancy across cameras sustaining perception quality. Yet at very high densities, computational bottlenecks and latency became limiting factors, leading to slower updates and reduced reliability. The cooperative perception approach thus remains robust up to a threshold, beyond which resource constraints begin to dominate system performance.

6.3.5 Effect of vehicle speed

As speed increases, challenges arise. First, observation time is decreased, fast moving vehicles are shown for less time in each camera to trigger detection recall, and therefore, there are fewer opportunities for detection. If detection recall is less than 100% at each frame, the odds of that particular car being missed are higher. On the other hand, slow cars are shown in the frame for several frames, increasing the chance for robust detection recall.

Second, timing sensitivity is correlated to speed, or more precisely, to the time something takes to move from one place to another. At a speed of 30 m/s (108 km/h), a vehicle moves 3 m in 0.1 s. Therefore, any timing misalignment can lead to meter scale position errors at average to high speed due to the short exposure time in combination with the speed the vehicle is traveling. In short, speed creates the possibility for every fused observation to become more outdated by the time it's used.

Third, computing gaps will become more sensitive. At higher speeds, any small spatial or temporal errors will have a magnified effect. For example, a small position error may be inconsequential for slow-moving traffic, but it could be crucial when the vehicle is moving at a higher speed because it would change the projected safety margin and potentially change the merge. Any of these errors could lead to either excessively conservative behavior (missing a gap that could have been used) or risky behavior (by misjudging the approaching vehicle).

In summary, elevated speeds marginally reduced the detection rate and localization accuracy, significantly resulting from a combination of reduced exposure time and latency effects. Very fast velocities caused the merge advisories to become less reliable and accurate. This suggests higher-speed operation will require either faster perception updates with tighter synchronization or merge logic specifically modified to account for increased uncertainty in cooperative perception data.

6.3.6 Maneuver performance

The principal objective of the cooperative perception system is to facilitate the ego-vehicle merging process in a safe and seamless manner. All of the previously discussed aspects have a direct impact on whether the RSU can share the merging maneuver accurately and whether the ego-vehicle can perform the merging maneuver safely. The results were evaluated in terms of merge success rate and merge quality.

In this scenario, the RSU noted the presence of gaps in the vehicles in the main lane and directed the ego-vehicle on when to merge. If the merge behavior completed without a collision and was efficient (less acceleration) for the ego-vehicle, then it was considered a

successful merge. Two primary situations caused a merge to fail. The first was the case in which there was no detection or a stale detection. If a vehicle situation becomes occluded for even a second, and the RSU miscalculates the gap, either an unsafe merge is suggested or a valid opportunity is missed. Most of the missed merge opportunities would occur with a single camera run and would be less common in a multi-camera run. Second, the location of the vehicles could have bias or drift, meaning the true position would be inaccurate and affect the RSU estimation of the gap.

Whenever traffic density and speed increased at the same time, the merge success rates decreased. At the moderate level, with three cameras, the merges went consistently smooth and well-timed. However, at the extreme level when the traffic was dense or fast moving, there were times when the ego-vehicle had unsuccessful merges. Regardless, the system exhibited considerable robustness under simple highway approaches, permitting good merges without compromising safety. Performance began to suffer in extreme or demanding edge cases.

Overall, the cooperative perception and advice system operated well under realistic conditions. The performance degradation in extreme loads shows us the design considerations: provide redundancy in referring sensors and conservative decision policies.

6.3.7 Discussion

The results validate the core hypothesis: infrastructure-assisted cooperative perception with modest camera redundancy and calibrated fusion can support safe, efficient highway merging when coupled to low-latency V2X advisories and a conservative ego-vehicle controller.

The qualitative predictions boil down to simple and coherent implications aligned with the pipeline implemented earlier: increasing the number of cameras enhances recall and localization until computational limits are reached. Higher numbers of vehicles result in increased occlusion and processing/network load that deteriorates freshness of detection and reliability of communications. Increased speed reduces the temporal margin for detection while also increasing the penalty associated with timing skews. The combination of rapid speeds and dense vehicles represents the most challenging condition.

The results show that the solution achieves consistently high detection precision and localization accuracy, though performance declines under higher vehicle speeds and dense occlusions, particularly when fewer cameras are used. Scalability analysis indicates hardware as the main limiting factor on perception throughput. Real-time factor degradation follows expected patterns, with synchronization stability maintained until resource contention forces the simulation to run slower than real-time. This creates a feedback loop where delayed perception processing affects the entire co-simulation timeline, demonstrating the critical importance of maintaining adequate computational capacity. Finally, maneuver support demonstrates that the RSU cooperative advisories allow successful merges, with failures occurring in cases of extreme density and velocity.

Chapter 7

Conclusions

This chapter concludes the work developed in this thesis, providing an assessment of the limitations encountered, identifying directions for future improvements, and consolidating the key findings and contributions.

7.1 Limitations

The development of the proposed co-simulation framework revealed some limitations that impact both the fidelity of the simulation and the overall scalability of the system.

One key limitation is camera modeling. In Gazebo, it was possible to incorporate multiple fixed cameras with realistic intrinsic and extrinsic parameters. However, it could not simulate realistic variations, such as changes in weather, changing levels of lighting, or sensor noise. This diminishes the realism of detection results and could result in different performance between the simulation and real-world deployment. For example, while the road is flat and we are measuring ground contact with a single-ray back-projection method, we could still have bias in localization with a gently sloped or banked road, or roads with elevation changes.

YOLO performed well in terms of detection in baseline experiments but deteriorated when additional cameras and detected objects were introduced, where the performance was lacking in inference as well as processing in a full chain. In heavy traffic conditions, the simultaneous decoding of the images, resizing of the images, and inference rapidly equaled the available computational power and inserted delays into the processing while reducing the performance of detection and quality of fusion. This poses an upper limit to scale for large sensing systems within a single workstation.

Simplification of gap selection limits generalization of scenarios. The RSU-based gap analysis is founded on uniform speeds of automobiles and level road profiles and therefore does not analyze complex maneuvers such as acceleration, deceleration, lane changes, and multi-lane merging. Such simplification limits the ability of the framework to simulate a range of heterogeneous and dynamic traffic scenarios.

7.2 Future Work and Improvements

The identified limitations of the current setup indicate several avenues for future research and development. Improved sensor simulation is one such area. The incorporation of environmental dynamics such as rain, fog, or nighttime would provide improved detection robustness evaluation under more realistic conditions. Integration of other sensor modalities such as LiDAR and radar along with cameras would improve perception coverage and enable heterogeneous sensor fusion development.

Perception algorithm advancements are also of significant potential. Research on lighter, real-time detection models or optimization techniques could offset the scalability limitations observed. Incorporating object tracking filters would also reduce localization drift and improve temporal consistency between frames, ensuring overall perception is more stable.

Improving realistic gap prediction is another avenue for enhancement. The RSU decision-making process could be extended to account for variable vehicle dynamics, driver reaction times, and multi-lane merging scenarios. Integrating advanced motion prediction algorithms would allow the system to support cooperative maneuvers under more complex and realistic traffic assumptions.

Another important limitation is that the communication subsystem has not yet been properly analyzed or evaluated in the thesis. As of now, the experiments have been predominantly focused on the perception and maneuver support pipeline. There is still a need to evaluate and to quantify an analysis of end-to-end communication latencies, jitter, and packet delivery ratio. In addition to this, while we examined detection accuracy, localization error, scalability, and maneuver success, stronger evidence and reproducibility are needed to conduct more systematic evaluations and quantify these metrics.

7.3 Conclusion and Final Considerations

This thesis has presented the design, development, and validation of a distributed vehicle perception system in a co-simulation framework. That serves as a comprehensive testbed for cooperative perception approaches. Building on the combination of ROS 2, Gazebo, and OMNeT++, the work extended perception capabilities from the ego-vehicle into the environment with the use of roadside cameras and a Roadside Unit (RSU) to promote safe and efficient merging. The system illustrated how infrastructure-assisted perception could improve vehicle perception by using multi-camera YOLO detection combined with trilateration. At the decision level, the RSU was able to successfully determine available gaps in highway traffic and suggest merging velocities that balanced safety margins and acceleration minimization, validating the viability of infrastructure-assisted cooperative maneuvers.

At the same time, this work also highlighted limitations important for interpretation of the findings, including scalability of the perception pipeline in heavy traffic and the robustness of detection under variance. These limitations emphasize further research, including future work related to advanced fusion of multiple kinds of sensors and redundancy mechanisms in the event of a failure of perception.

To summarize, the extension of the co-simulation framework offers a proof-of-concept platform showcasing the promise of infrastructure-aided cooperative perception whilst providing a platform for future research into other distributed sensing, vehicular communications, and cooperative decision frameworks.

Bibliography

AbdelHamed, Ahmed, Girma Tewolde, and Jaerock Kwon (Sept. 2020). "Simulation framework for development and testing of autonomous vehicles". en. In: *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*. Vancouver, BC, Canada: IEEE.

Achenef, Tadele Bishaw et al. (2024). "Latest breakthroughs, research results, and challenges in intelligent control of autonomous vehicles". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 0.0, p. 09544070251327616. doi: 10.1177/09544070251327616. url: https://doi.org/10.1177/09544070251327616.

Bonci, Andrea et al. (Nov. 2023). "Robot Operating System 2 (ROS2)-based frameworks for increasing robot autonomy: A survey". en. In: *Appl. Sci. (Basel)* 13.23, p. 12796.

Caillot, Antoine et al. (2022). "Survey on Cooperative Perception in an Automotive Context". In: *IEEE Transactions on Intelligent Transportation Systems* 23.9, pp. 14204–14223. doi: 10.1109/TITS.2022.3153815.

Dosovitskiy, Alexey et al. (2017). "CARLA: An Open Urban Driving Simulator". In: *Conference on Robot Learning*. url: https://api.semanticscholar.org/CorpusID:5550767.

Fu, Yanjin et al. (2020). "A Camera–Radar Fusion Method Based on Edge Computing". In: *2020 IEEE International Conference on Edge Computing (EDGE)*, pp. 9–14. doi: 10.1109/EDGE50951.2020.00009.

Han, Yushan et al. (Nov. 2023). "Collaborative Perception in Autonomous Driving: Methods, Datasets, and Challenges". In: *IEEE Intelligent Transportation Systems Magazine* 15.6. issn: 1941-1197. doi: 10.1109/mits.2023.3298534. url: http://dx.doi.org/10.1109/MITS.2023.3298534.

Huang, Tao et al. (May 2025). "Vehicle-to-Everything Cooperative Perception for Autonomous Driving". In: *Proceedings of the IEEE* 113.5, pp. 443–477. issn: 1558-2256. doi: 10.1109/jproc.2025.3600903. url: http://dx.doi.org/10.1109/JPROC.2025.3600903.

Janai, Joel et al. (2021). *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*. arXiv: 1704.05519 [cs.CV]. url: https://arxiv.org/abs/1704.05519.

Keivani, Arghavan, Farzad Ghayoor, and Jules-Raymond Tapamo (June 2021). "Collaborative Mobile Edge Computing in eV2X: A Solution for Low-Cost Driver Assistance Systems". In: *Wireless Personal Communications* 118. doi: 10.1007/s11277-019-06401-2.

Kherroubi, Zine el abidine, Samir Aknine, and Rebiha Bacha (2022). "Novel Decision-Making Strategy for Connected and Autonomous Vehicles in Highway On-Ramp Merging". In: *IEEE Transactions on Intelligent Transportation Systems* 23.8, pp. 12490–12502. doi: 10.1109/TITS.2021.3114983.

Nsnam (2025). *About*. url: https://www.nsnam.org/about/.

Oliveira, Miguel Ferreira (Dec. 2023). *Towards the simulation of cooperative perception applications by leveraging distributed sensing infrastructures*. url: http://hdl.handle.net/10400.22/24053.

Padmaja B., Moorthym (2023). "Exploration of issues, challenges and latest developments in autonomous cars". In: *J Big Data* 10.

Ribeiro, Luis (2024). *Supporting the Assessment of Energy Efficient Vehicular Automated Systems by leveraging Distributed Perception*.

ROS2 (2025). *Understanding nodes ROS 2 Documentation: Rolling documentation*. https://docs.ros.org/en/rolling/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html. [Accessed 06-07-2025].

Rosique, Francisca et al. (2019). "A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research". In: *Sensors* 19.3. issn: 1424-8220. doi: 10.3390/s19030648. url: https://www.mdpi.com/1424-8220/19/3/648.

SAE (2021). en. https://www.sae.org/. Accessed: 2025-1-10.

Shan, Kang et al. (2024). "Experimental Study of Multi-Camera Infrastructure Perception for V2X-Assisted Automated Driving in Highway Merging". In: *IEEE Transactions on Intelligent Transportation Systems* 25.11, pp. 16207–16220. doi: 10.1109/TITS.2024.3424673.

Silva, Ivo et al. (Sept. 2024). "Realistic 3D simulators for automotive: A review of main applications and features". en. In: *Sensors (Basel)* 24.18, p. 5880.

Tan, Kai et al. (Jan. 2024). "Integrating Advanced Computer Vision and AI Algorithms for Autonomous Driving Systems". In: *Journal of Theory and Practice of Engineering Science* 4.01, pp. 41–48. doi: 10.53469/jtpes.2024.04(01).06. url: https://centuryscipub.com/index.php/jtpes/article/view/427.

Teper, Harun et al. (2022). *AuNa: Modularly Integrated Simulation Framework for Cooperative Autonomous Navigation*. arXiv: 2207.05544 [cs.RO]. url: https://arxiv.org/abs/2207.05544.

Tsukada, Manabu et al. (2020). "Networked Roadside Perception Units for Autonomous Driving". In: *Sensors* 20.18. issn: 1424-8220. doi: 10.3390/s20185320. url: https://www.mdpi.com/1424-8220/20/18/5320.

Ullah, Irfan et al. (2025). "Assessing the barriers and implications of autonomous vehicles: Implementation in sustainable cities". In: *Sustainable Futures* 9, p. 100564. issn: 2666-1888. doi: https://doi.org/10.1016/j.sftr.2025.100564. url: https://www.sciencedirect.com/science/article/pii/S2666188825001340.

Varga, Andras (2019). "A Practical Introduction to the OMNeT++ Simulation Framework". In: *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*. Ed. by Antonio Virdis and Michael Kirsche. Cham: Springer International Publishing, pp. 3–51. isbn: 978-3-030-12842-5. doi: 10.1007/978-3-030-12842-5_1. url: https://doi.org/10.1007/978-3-030-12842-5_1.

Vieira, Bruno et al. (2019). "COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications". In: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 1–6. doi: 10.1109/ICCVE45908.2019.8965161.

Wang, Baoming et al. (2024). "Current State of Autonomous Driving Applications Based on Distributed Perception and Decision-Making". In: *World Journal of Innovation and Modern Technology*. url: https://api.semanticscholar.org/CorpusID:270871572.

Wang, Xu et al. (2024). *Moving Forward: A Review of Autonomous Driving Software and Hardware Systems*. arXiv: 2411.10291 [cs.RO]. url: https://arxiv.org/abs/2411.10291.

Wang, Yingjie et al. (2023). *Multi-Modal 3D Object Detection in Autonomous Driving: a Survey*. arXiv: 2106.12735 [cs.CV]. url: https://arxiv.org/abs/2106.12735.

Widen, William H. and Philip Koopman (2022). *Autonomous Vehicle Regulation and Trust*. url: http://dx.doi.org/10.2139/ssrn.3969214.

Yeong, De Jong et al. (2021). "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review". In: *Sensors* 21.6. issn: 1424-8220. doi: 10.3390/s21062140. url: https://www.mdpi.com/1424-8220/21/6/2140.

Zhu, Jie, Said Easa, and Kun Gao (2022). "Merging control strategies of connected and autonomous vehicles at freeway on-ramps: A comprehensive review". In: *Journal of Intelligent and Connected Vehicles* 5.2, pp. 99–111. doi: 10.1108/JICV-02-2022-0005.