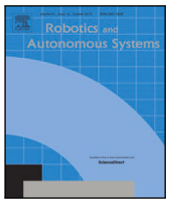
Robotics and Autonomous Systems 199 (2026) 105389

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Design insights and comparative evaluation of a hardware-based cooperative perception architecture for lane change prediction ☆

Mohamed Manzour [a],*, Catherine M. Elias [b], Omar M. Shehata [c], Rubén Izquierdo [a], Miguel Ángel Sotelo [a]

[a] Computer Engineering Department, University of Alcalá, Madrid, Spain
[b] C-DRiVes Lab, Computer Science and Engineering Department, German University in Cairo, Egypt
[c] Multi-Robot Systems Lab, Mechatronics Engineering Department, German University in Cairo, Egypt

ARTICLE INFO

Keywords:
Lane change prediction
Hardware validation
Cooperative perception
Knowledge graph embeddings
Autonomous driving safety

ABSTRACT

Research on lane change prediction has gained attention in the last few years. Most existing works in this area have been conducted in simulation environments or with pre-recorded datasets, and they often rely on simplified assumptions about sensing, communication, and traffic behavior that do not always hold in practice. Real-world deployments of lane-change prediction systems are relatively rare, and when they are reported, the practical challenges, limitations, and lessons learned are often under-documented. This study explores cooperative lane-change prediction through a real hardware deployment in mixed traffic and shares the insights that emerged during implementation and testing. The studied architecture integrates stereo-camera perception, wireless communication, a knowledge-graph-based intention prediction module, and automated longitudinal control implemented on embedded platforms. It is implemented on an ego vehicle and a target vehicle and evaluated in a three-vehicle scenario where a third vehicle acts as the preceding vehicle that forces the target vehicle to change lane. Real-road experiments show that, when the cooperative prediction module is enabled, the ego vehicle can anticipate the target vehicle's lane-change intention about 4 s before the actual lane crossing and decelerate early to open a safe gap, whereas disabling prediction leads to late reactions and aggressive braking. The experiments also reveal constraints that are critical for real deployments. Perception pipelines are sensitive to outdoor lighting, so tests must be scheduled at times and locations with more stable illumination. A precomputed lookup table keeps prediction fast on embedded devices. Communication reliability and thermal effects on the hardware, especially in hot weather, can noticeably affect the system behavior. By documenting these experiences together with the observed behavior of the vehicles with and without prediction, the study provides practical guidance for others working on similar cooperative prediction systems.

1. Introduction

Traffic accidents remain a major global concern, with lane-change maneuvers recognized as one of the significant contributors to collision risk. Anticipating these maneuvers has become an important research focus, supporting both traffic safety and the safe integration of autonomous and assisted driving technologies. Over the past decade, numerous models have been developed for lane-change prediction. However, most existing works have been designed and validated using simulation environments or pre-recorded datasets. While these settings allow for benchmarking and controlled evaluation, they often rely on simplified assumptions about sensing, communication, and vehicle behavior that do not fully capture the complexity of real-world operation. Real-world deployments of lane-change prediction systems are relatively rare, and when they are reported, their practical challenges, limitations, and insights remain under-documented. To illustrate the setting more concretely, consider the left lane change scenario shown in Fig. 1. The Ego Vehicle (EV) is driving in the left lane, while the Target Vehicle (TV) is moving in the right lane behind a Preceding Vehicle (PV). When the PV suddenly brakes, the TV must change lanes to avoid a collision. The TV senses its surroundings using onboard sensors. However, instead of acting in isolation, it communicates its sensory data to an

☆ This research has been funded by the HEIDI project of the European Commission under Grant Agreement: 101069538.
* Correspondence to: San Diego Square, s/n. 28801, Alcalá de Henares, Madrid, Spain.
E-mail addresses: ahmed.manzour@uah.es (M. Manzour), catherine.elias@ieee.org (C.M. Elias), omar.mohamad@guc.edu.eg (O.M. Shehata), ruben.izquierdo@uah.es (R. Izquierdo), miguel.sotelo@uah.es (M.Á. Sotelo).

https://doi.org/10.1016/j.robot.2026.105389
Received 25 September 2025; Received in revised form 9 December 2025; Accepted 5 February 2026
Available online 9 February 2026

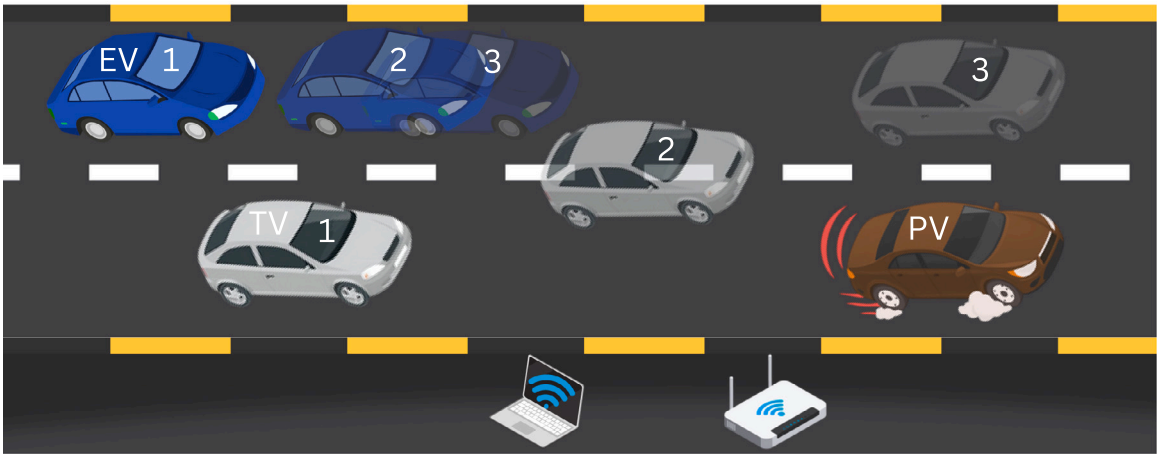


Fig. 1. Cooperative lane-change scenario. The Target Vehicle (TV) follows a Preceding Vehicle (PV) in the right lane, while the Ego Vehicle (EV) travels in the left lane. When the PV brakes, the TV shares its sensory data via a relay to the EV, enabling the EV to predict the TV's left lane change and create a safe gap.

external relay unit. This relay processes the sensory data and transmits it to the EV in a form suitable for prediction. Using this information, the EV anticipates the TV's left lane change maneuver and adapts its longitudinal behavior in time to create a safe gap for the lane change. Without this cooperative exchange, the EV would simply continue at its speed, leaving insufficient space and making the maneuver unsafe. This example illustrates how cooperation between vehicles, supported by an intermediate relay, enables timely and reliable lane-change prediction.

Building on this scenario, the present study examines a cooperative-perception pipeline deployed in mixed traffic, which contains two human-driven vehicles (the PV and TV) (drivers sitting in the vehicles; no remote-control interface is used) with an automated scaled ego vehicle (EV). The aim is not to propose a new architecture but to share the insights and experiences gained during the real-world hardware deployment. We focus on documenting the bottlenecks, reliability issues, and operational constraints encountered across perception, prediction, and control. By reporting these findings, the study provides experience-based guidance that may support researchers and practitioners working on other similar systems.

2. Literature and research foundation

2.1. Literature review

2.1.1. Dataset-based validation

Many early studies have employed traditional Machine Learning (ML) models to predict the intentions of surrounding vehicles, primarily leveraging pre-recorded datasets without real-world or simulation-based validation. eXtreme Gradient Boosting (XGBoost) was applied to the Next Generation Simulation (NGSIM) dataset by Zhang et al. [1], while Syama and Mala [2] employed both Support Vector Machine (SVM) and Random Forest (RF) on the same dataset. Logistic regression was adopted by Ma et al. [3] on the highD dataset. Across these works, evaluation was limited to dataset-based numerical validation, with no implementation in simulation environments or on physical hardware. All studies assumed seamless and error-free communication between vehicles, using datasets captured from a third-person perspective. As the highD dataset was recorded from a drone-mounted camera, and the NGSIM dataset a camera installed on fixed roadside infrastructure, both providing complete and noise-free vehicle state information which are conditions that does not reflect the challenges of real-world perception and communication systems.

Similarly, some studies relied on the Bayesian network family of models, which includes Static Bayesian Networks, Dynamic Bayesian Networks (DBNs), Hidden Markov Models (HMMs), and Naive Bayes classifiers. For example, Li et al. [4] and Liu et al. [5] applied DBNs to the NGSIM and highD datasets, respectively. Works Xia et al. [6]; Li et al. [7] employed HMMs, relying on kinematic inputs such as velocity, acceleration, and relative position obtained from utilizing the NGSIM dataset.

Other studies have relied on convolutional Deep Learning (DL) models, using image-based inputs rather than numerical features. Works Izquierdo et al. [8]; Liang et al. [9]; Fernández-Llorca et al. [10]; Biparva et al. [11]; Izquierdo et al. [12] employed variants of Convolutional Neural Networks (CNNs) and hybrid CNN–Long Short-Term Memory (LSTM) architectures on the PREdiction of VEhicles iNTenIONs (PREVENTION) dataset, which provides an ego-vehicle perspective of the driving scene. These implementations varied in their architectural choices, including standard CNNs, 3D CNNs, and spatio-temporal ConvNets for capturing both spatial and temporal dependencies in the image sequences.

Building on these spatio-temporal approaches, the research focus gradually shifted toward sequential DL models that emphasize the temporal dynamics of driving behavior. These models include Recurrent Neural Networks (RNNs), LSTM networks, and Gated Recurrent Units (GRUs), as well as their advanced variants such as bidirectional LSTMs and attention-based LSTMs. For instance, Su et al. [13] utilized an LSTM model trained on the NGSIM dataset, while work Amer and Elias [14] applied LSTMs to the PREVENTION dataset, both works relied on kinematic inputs such as vehicle positions, accelerations, and relative velocities. Work Shangguan et al. [15] trained an LSTM model on the highD dataset, while works Scheel et al. [16,17] incorporated attention mechanisms into their LSTM architectures. Additionally, Laimona et al. [18] conducted a comparative study between RNN and LSTM models using the PREVENTION dataset, where the input consisted of sequences of X–Y centroid coordinates extracted from vehicle detection outputs.

More recent studies used transformers as their prediction model. For example, a dual-transformer architecture was proposed by Gao et al. [19], one for lane-change intention prediction and another for trajectory forecasting. The first transformer processes the target vehicle's historical lateral motion along with the relative distances and velocities of surrounding vehicles. Its predicted intention is then fused with the target vehicle's past lateral movements and passed to the second transformer, which generates the future trajectory. The model was trained and evaluated on the HighD and NGSIM datasets. Guo et al. [20]; Lu et al. [21] employed the NGSIM and highD datasets together. These works generally use numerical inputs such as relative distances and velocities of surrounding vehicles, along with the lateral and longitudinal positions of the EV. In contrast, the study in [22] tackled lane-change and trajectory prediction on the highD dataset by leveraging the strong

reasoning and self-explanation capabilities of Large Language Models (LLMs). Numerical features were converted into natural-language prompts to serve as inputs to the LLMs, and Chain-of-Thought (CoT) reasoning was employed to produce interpretable explanations for the predictions. A key strength of this approach is its ability to deliver high prediction performance alongside clear, human-readable explanations, relying solely on the capabilities of the fine-tuned LLMs.

Graph Neural Networks (GNNs) were also explored. For instance, Lu et al. [23] employed Graph Attention Networks (GATs) and Graph Convolutional Networks (GCNs), while Li et al. [24] adopted Spatio-Temporal Graph Neural Networks (ST-GNNs). Both approaches were trained and evaluated on the NGSIM and highD datasets.

Other works have explored graph representation learning approaches, particularly Knowledge Graph Embeddings (KGEs). For example, Manzour et al. [25] employed KGE combined with Bayesian inference as a downstream reasoning module, using the learned embeddings to predict lane-change maneuvers on the highD dataset. Linguistic inputs were incorporated to improve interpretability, relying on features such as lateral velocity, lateral acceleration, and Time-To-Collision (TTC)-based risk with surrounding vehicles. Building on this framework, Hussien et al. [26] extended the architecture by integrating a Retrieval Augmented Generation (RAG) mechanism, leveraging the capabilities of LLMs to generate context-specific explanations, that are grounded in external, verifiable knowledge. This enhancement enables the system to describe the driving scene and state the reason behind the predicted lane change in natural language, making the outputs more transparent and understandable to end users.

Beyond explicit prediction models, some recent work has used trajectory datasets mainly to analyze lane-changing behavior rather than to predict future maneuvers. For example, Chen et al. [27] analyzed highway lane-changing behavior using 6506 single lane-change trajectories extracted from the highD dataset. They proposed a fusion method that combines frame-interval information and lateral displacement thresholds to automatically identify the start and end points of each lane change, and then computed lane-changing time and speed to perform a statistical comparison under different numbers of lanes, lane-change directions, speed limits, and vehicle types. Their results show that lane-changing time and speed differ significantly across these scenarios, providing useful microscopic insight for traffic-flow and safety modeling.

Some studies did not rely on publicly available datasets such as NGSIM or highD, but instead collected private datasets to validate their architectures or models. Most of these works focused primarily on predicting the intentions of the EV rather than surrounding vehicles. For instance, Papers Xing et al. [28]; Jain et al. [29] collected their perception data using cameras, Global Positioning System (GPS), and Controller Area Network (CAN) bus signals. Other works Berndt et al. [30]; Li et al. [31]; Li et al. [32]; Rastin and Söffker [33] relied on multi-sensor setups by using different combination of sensors such as cameras, Radio Detection and Ranging (Radar), Light Detection and Ranging (LiDAR), HD-maps, Inertial Measurement Unit (IMU), and CAN bus information to capture a richer description of vehicle dynamics and its surroundings.

In a related direction, Chen et al. [34] proposed a sensing-data-supported traffic flow prediction framework based on loop-detector measurements. Their pipeline first applies several denoising schemes, such as wavelet filtering, moving-average smoothing, and Butterworth filtering, to remove anomalies and noise from the raw traffic-flow time series. The denoised sequences are then used as inputs to an artificial neural network that predicts short-term traffic flow, and the authors compare how the different filtering methods affect prediction accuracy. This study shows that appropriate preprocessing of noisy sensing data can significantly improve prediction performance.

From these studies, we see that dataset-based validation has mostly been limited to extracting perception information (often simplified) and applying prediction models, without extending to decision-making or control. No coupling was made with even basic longitudinal actions such as braking or smooth deceleration. Furthermore, none of the works progressed toward simulation-based validation or real-world hardware testing, which limits their relevance and applicability in actual driving environments.

2.1.2. Simulation-based validation

Some studies did not limit their validation to datasets alone but extended it to simulation-based evaluations combined with decision-making modules. For example, Li et al. [35] reported numerical validation on the NGSIM dataset and further tested the model in the CarSim environment. The work indicated that the planning stage was based on a Nonlinear Model Predictive Control (NMPC) scheme, but did not specify whether this planning was actually executed by the EV, it may have remained at the planning level without execution. Similarly, Manzour et al. [36] validated its approach on the highD and CARLA Risky-lane-change Anticipation in Simulated Highways (CRASH) datasets, aiming to predict both risky and safe lane-change maneuvers. The work incorporated the target vehicle's lateral velocity and acceleration, lane ID, its position within the lane, TTC with surrounding vehicles, Time Headway (THW) with the preceding vehicle, and the lane with the highest frontal gap. These features were converted into linguistic categories and used to generate a knowledge graph, which was then embedded using the AmbliGraph library [37]. Bayesian inference was applied on top of these embeddings to compute the final prediction probability. This work was extended in the CARLA simulator, where the planing was executed only in the longitudinal direction (brake or smooth deceleration), and no details were provided on the specific controller used for execution.

2.1.3. Hardware-based validation

A smaller set of works carried this further into hardware real-world testing. For instance, Morris et al. [38]; Doshi et al. [39] validated their prediction models in on-road conditions. However, these efforts focused solely on the prediction stage without incorporating any planning or control modules. In contrast, other studies combined real-world testing with decision-making. Work Manzour et al. [40] presented a real hardware implementation for predicting lane changes in which a perception module collected numerical sensor data, converted it into linguistic categories, and transmitted these via TCP/IP to a prediction module offline model based on Knowledge Graph Embeddings and Bayesian inference. The prediction was fed to a low-level longitudinal controller that decided whether the EV should stop or continue. This study was limited to planing in the longitudinal direction only. However, Gonzalo et al. [41] addressed both lateral and longitudinal planning. Nevertheless, in both works, the specific controllers employed to execute these planned maneuvers were not explicitly described. Also, Manzour et al. [40] introduced communication between vehicles, where sensory data from one vehicle was processed and transmitted to another, enabling cooperative maneuver anticipation.

2.2. Observed functional blocks

From the reviewed literature, a pattern of functional building blocks emerges. Dataset-based works focused mainly on extracting vehicle states from sensors or datasets (Perception) and applying models to anticipate maneuvers (Prediction). Later studies extended these models into simulation, where predicted intentions were connected to Planning and Control. Finally, hardware-based implementations introduced real-world execution, and in some cases Communication between vehicles, where information was exchanged to enable cooperative behavior. Together, these observations suggest that most lane-change prediction systems can be described in terms of four interconnected blocks: Perception, Communication, Prediction, and Planning & Control.

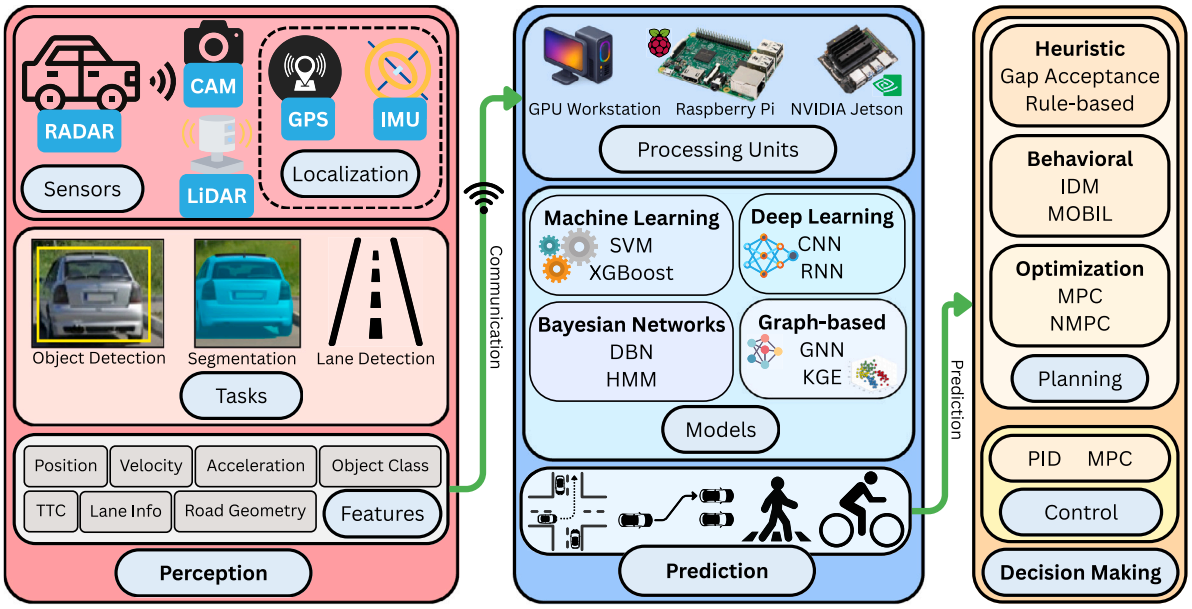


Fig. 2. Generalized prediction pipeline covering perception, prediction, and decision-making with control, applicable to diverse scenarios and agents in autonomous and cooperative systems.

2.3. General prediction pipeline

Based on these observations, a generalized prediction pipeline can be formulated. As illustrated in Fig. 2, the process begins with the Perception module, where raw sensory inputs such as cameras, LiDAR, Radar, GPS, or IMU are transformed into meaningful representations. These include tasks such as object detection, lane detection, segmentation, or 3D point cloud generation. From these outputs, relevant features are extracted, including vehicle position, velocity, acceleration, TTC, surrounding object classes, and lane geometry. When cooperation is considered, the Communication module enables these features to be shared between vehicles or with roadside infrastructure using, for example, Wi-Fi links or roadside units. The Prediction module then builds on this shared information to anticipate the future behavior of observed agents. Depending on the computational platform (ranging from high-performance Graphics Processing Units (GPUs) to embedded devices such as Raspberry Pi and NVIDIA boards) different algorithmic families can be employed, including ML (e.g., SVM, XGBoost), DL (e.g., CNNs, RNNs, Transformers), and graph-based approaches (e.g., GNNs, KGEs). Such models are capable of predicting not only vehicle trajectories and lane-change maneuvers but also the intentions of other road users, including pedestrians and cyclists, in complex traffic environments. The resulting predictions are then passed to the Decision-Making and Control modules, which plan the EV's longitudinal and lateral actions. This planning may rely on behavioral models such as Intelligent Driver Model (IDM) or Minimizing Overall Braking Induced by Lane Changes (MOBIL), heuristic or rule-based strategies such as gap acceptance, or optimization-based methods like Model Predictive Control (MPC) or NMPC. Finally, controllers such as Proportional–Integral–Derivative (PID) or MPC ensure that these planned actions are carried out reliably, enabling safe and coordinated operation. A similar end-to-end architecture has also been demonstrated on real hardware in previous works. For example, Manzour et al. [40] focused on lane-change prediction, while Manzour et al. [42] addressed pedestrian crossing prediction. Both studies followed a similar architectural flow as described here, starting from perception, then communication and prediction, and ending with decision-making with control. This generality highlights the pipeline as a unifying framework for many prediction tasks in autonomous and cooperative systems. However, it should be emphasized that in this study, the focus remains specifically on lane-change intention prediction.

2.4. Study objective

The objective of this work to study how a cooperative prediction pipeline behaves when deployed in hardware and to share the insights gained from this process. The focus is on documenting what was observed in practice rather than presenting a novel method. In particular, this study aims to:

- Examine the operation of each module in the pipeline (Perception, Communication, Prediction, Planning & Control) under real-world conditions.
- Report the practical challenges and bottlenecks encountered in each module, such as sensing limitations, computational load, communication delays, and thermal issues.
- Compare different processing alternatives to highlight trade-offs in performance and feasibility.
- Share the lessons learned during implementation, providing experience-based guidance for researchers and practitioners working on similar cooperative prediction systems.

3. Experimental setup

Fig. 3 illustrates the setup of the studied scenario, which involves three vehicles: the EV, the TV, and the PV. The PV is a human-driven Peugeot 301 measuring 4.5 m in length and 1.8 m in width. Its role in the scenario is to move ahead of the TV and then apply a sudden brake, creating a risk that forces the TV to react. The TV is a golf cart equipped with sensors, with dimensions of 3.0 x 1.5 m, and is also manually driven. It perceives the environment and responds to the PV's braking by attempting a left-lane change. The EV is a scaled-down vehicle, 1.2 m long and 0.7 m wide, which does not carry its own perception sensors. Instead, it is fitted with actuation and prediction modules.

Initially, all three vehicles start from stationary positions on a straight 40-meter asphalt test track located inside the campus of the German University in Cairo (Egypt). The EV is positioned 12 m ahead, but since its average speed is only 1.5 m/s compared to 2.5 m/s for the TV and PV, it is gradually overtaken. After approximately 12 s, the PV brakes suddenly, forcing the TV to attempt a lane change to avoid collision. The experiments were conducted on this closed outdoor

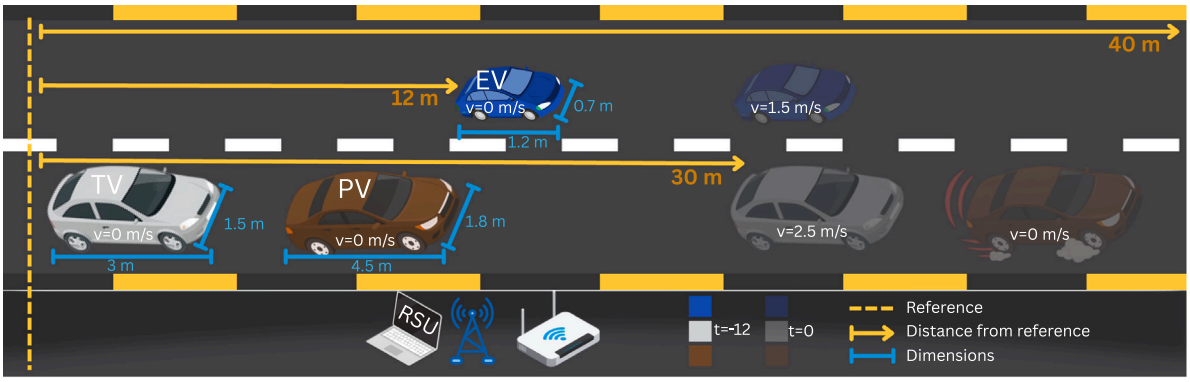


Fig. 3. Experimental scenario with three vehicles: EV, TV, and the PV. At $t = -12$ s, the vehicles are shown in solid color as they move along the 40-meter test track. As time advances, transparent overlays represent their updated positions until $t = 0$, which is the moment when the PV brakes suddenly and the TV initiates a left lane change.

section, with no other traffic present and ambient temperatures around 30–40°C. Early tests were performed in the afternoon under strong sunlight, which produced glare and distortion in the stereo camera images and increased the thermal load on the embedded boards. For this reason, the final trials reported in this article were carried out close to sunset, when illumination was more uniform and temperatures were slightly lower. Two experimental cases were considered. In the first, the EV has no prediction module and continues straight without anticipating the TV's maneuver, leaving the TV with no safe gap and forcing it into harsh braking. In the second case, the EV is equipped with the prediction module. Here, the TV sends its perceptual data to a relay device, which processes and forwards it to the EV. Using this cooperative information, the EV anticipates the TV's lane-change intention, decelerates in advance, and creates the necessary space for the maneuver. After several module-level experiments to determine the best suitable hardware and configuration settings for the perception, communication, and prediction modules, we fixed the final setup used in the integrated tests. With this configuration in place, each of the two system-level cases (with and without prediction) was executed four times under the same conditions: two preliminary trials were used to verify the setup and communication, and two additional trials were recorded and used for analysis. In total, eight executions were conducted (four without prediction and four with prediction). The detailed module-level experiments and comparisons between alternative configurations (e.g., different perception, communication, and prediction implementations) are reported in the following sections, where each module has its own baseline, while at the system level the scenario without prediction serves as the baseline against the proposed cooperative architecture with prediction. Fig. 3 shows the positions of the vehicles at two reference times. At $t = -12$ s, when the vehicles are still in steady motion before the PV applies the brake, the vehicles are shown in solid color. As time progresses toward $t = 0$, transparent overlays are used to indicate the advancing positions of the vehicles showing the critical moment when the PV brakes suddenly and the TV initiates a left lane change.

Since the experiments are conducted with scaled vehicles rather than full-scale passenger vehicles, their mass, speed range, and thrust-to-weight ratio differ from real vehicles. As a consequence, the absolute braking distances and accelerations observed in our trials are specific to this scaled testbed and are not intended to be interpreted as direct safety margins for full-scale autonomous driving. Instead, in this study we use the platform to compare the behavior of the system with and without cooperative prediction and to analyze time-based quantities such as the anticipation horizon and the timing between perception, communication, and prediction. A more detailed discussion of this dynamic gap and its implications is provided in Section 9.1.5.

Fig. 4 provides an overview of the hardware components integrated into each vehicle and the functional modules they support. In the TV, a ZED stereo camera captures the environment, and its outputs are processed by an onboard laptop *(Device 1)* that executes different perception algorithms. A microcontroller is also installed to measure the TV's own velocity. This perceptual information is then sent to the relay module, implemented as a Roadside Unit (RSU) *(Device 2)*, which is responsible for receiving, processing, and transmitting the data to the EV. The EV contains the prediction module, which was tested across different computing platforms including a standard laptop, a Raspberry Pi, and an NVIDIA Jetson Nano board. For control, the EV separates longitudinal and lateral functions. Longitudinal actuation (throttle and braking) is implemented through a Raspberry Pi that serves as a high-level device, receiving inputs from the prediction module and the EV's onboard encoder, and sending throttle/braking commands to another low level controller. Lateral control (responsible for lane-keeping) is handled by a microcontroller that reads IMU data and sends steering commands to the vehicle. The PV, on the other hand, is a human-driven vehicle used only to create the risky scenario by braking suddenly and does not incorporate additional sensing or communication modules. Figs. 3 and 4 therefore illustrates the scenario flow and how each vehicle is set up to fulfill a distinct role in the cooperative experiment: the PV as the disturbance, the TV as the sensor-equipped agent that will make the lane change, and the EV as the cooperative partner equipped with prediction and control. Fig. 5 illustrates the studied methods and tasks within each module. The perception module handles sensing operations such as object detection, segmentation, and the estimation of TTC and THW between the TV and the PV. The relay module acts as an intermediate processing and communication unit, converting numerical values into linguistic categories and ensuring a standardized format for data transmission to the prediction module. The prediction module integrates KGE with Bayesian inference to anticipate the TV's maneuver, supported by an offline implementation for real-time operation on embedded platforms. Finally, the maneuvering module translates these predictions into vehicle actions: longitudinal braking and throttle commands are executed through a Raspberry Pi–Arduino chain, while lateral steering is coordinated through a low level microcontroller. Together, these studied architecture demonstrate how sensing, communication, prediction, and actuation are connected across the cooperative pipeline. The following subsections detail the operation of each module under real-world conditions.

4. Perception and localization modules

4.1. Target vehicle system breakdown

In a regular Autonomous Driving Stack (ADS), the vehicle is equipped with the localization, perception, mapping, planning, control, and system integration. However, in this work, the focus is to investigate the

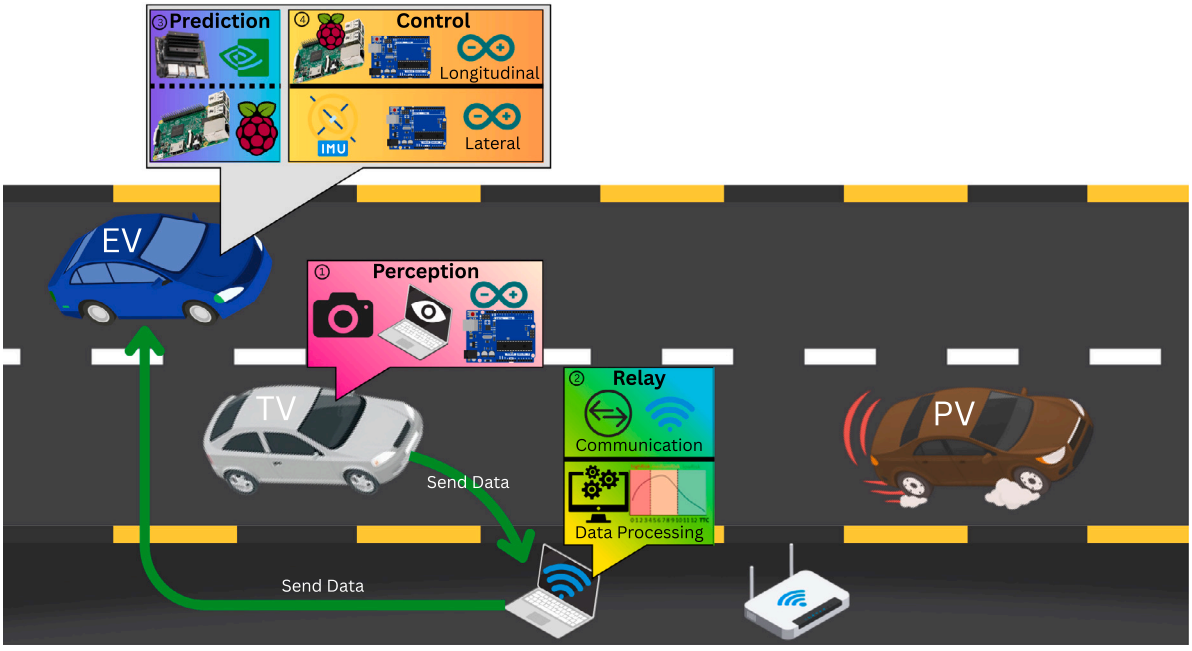


Fig. 4. Cooperative lane-change prediction scenario illustrated through the hardware components used in the experiment. The TV is equipped with a ZED camera, an onboard laptop, and a microcontroller to sense the environment. This perceptual data is transmitted to a relay Roadside Unit (RSU), which processes and communicates the information to the EV. The EV hosts the prediction module on Nvidia Jetson Nano board and executes control through a Raspberry Pi–microcontroller chain for longitudinal actuation, and a microcontroller for lateral lane-keeping. Through this cooperation, the EV anticipates the TV's maneuver and adjusts its longitudinal speed to ensure safe interaction.

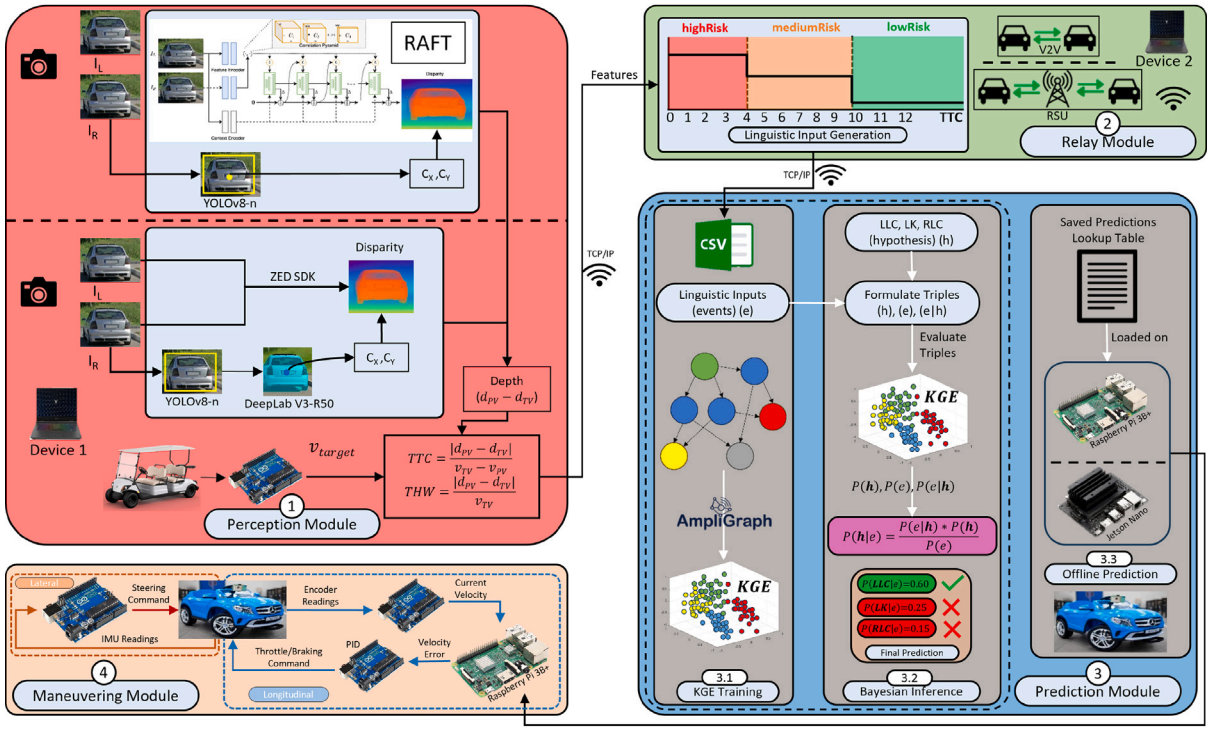


Fig. 5. Functional workflow of the prediction pipeline, showing the studied methods applied in perception, relay, prediction, and maneuvering modules.

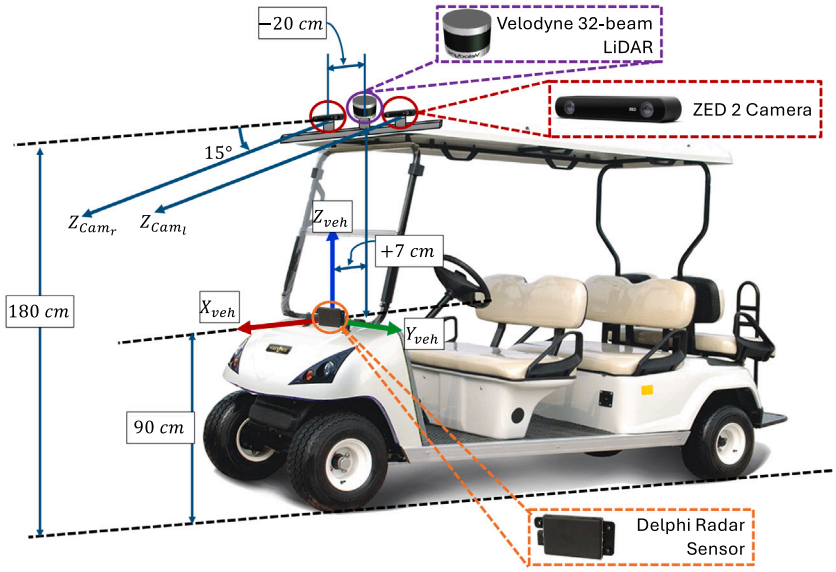


Fig. 6. The TV sensors mounting.

insights and challenges associated with incorporating the Cooperative Perception (CP) and how it will affect the quality of the risky maneuver prediction rather than presenting a full ADS. Therefore, the utilized TV is designed to be equipped with 2 main modules from the ADS. The first module is the localization module that is responsible for having information about the vehicle states. This information is necessary for the second module, which is perception. This module is mainly responsible for two main tasks; Object Detection and Safety Features Extraction. In this section, an experiment setup of the TV will be demonstrated as well as the implementation of the perception module and the message prepared to be sent to the relay.

4.2. Target vehicle setup

During the example pipeline, the TV used is a golf cart of type Marshell Utility Electric Golf Cart DG-C4+2 shown in Fig. 6. The TV is equipped with a number of sensors and processors that are used to perform the localization and perception modules. These sensors are mounted in the vehicle as shown in Fig. 6. The added perception sensors are:

- 2 Stereolabs ZED 2 camera with a resolution of $672\times376px$@10FPS (neural depth mode) and 119.89 mm baseline. The 2 cameras are mounted on the vehicle roof on both sides with a spacing of ±20 cm from the center of the vehicle and on a distance of 180 cm measured from the ground.
- The vehicle is also equipped with 32-beam Velodyne LiDAR and Delphi radar sensors. However, in this work, visual perception will be the main method implemented. Yet, the TV platform is designed to test other different perception pipelines/algorithms. Therefore, these essential sensors are added.
- The perception is done on two separate laptops which are used to run each studied pipeline solely. The specs of these laptops are: a laptop with Nvidia GeForce RTX 4060 GPU, and a laptop with NVIDIA GeForce RTX 3050 GPU.

4.3. Localization module

The localization module in the regular ADS is used mainly to get the TV's states including its position P_{TV} using Global Navigation Satellite System (GNSS) sensors either globally or locally, the vehicle's orientation Yaw_{TV} using IMU sensor, and some other important states customized to the work such as the longitudinal velocity V_{TV} using encoders and acceleration A_{TV} using accelerometers. In this work, since it is decided to disable the planning and control of the TV and only focus on the perception, it is only necessary to measure the vehicle's longitudinal velocity V_{TV_t} at each time sample t. This velocity is needed to compensate the relative measurement of the perception module. It is also important to mention that the time sample t is determined based on the perception module rate. The velocity of the vehicle is obtained in this work using direct measurement from the vehicle' throttle with calibration using the gear state. This velocity measurement is essential in estimating the absolute velocity of the PV $V_{PV_{Abs}}$. This measured TV's velocity is then compared with the velocity obtained from a mobile phone GPS. As shown in Fig. 7, the maximum absolute error is $3,\text{km/h}$ in the absence of rapid speed changes. It can also be observed that the throttle measurement responds faster to speed changes compared to the velocity calculated by the GPS. Although the error in throttle-based measurement is relatively high, the throttle method was chosen for the rest of the experiments. This decision is mainly due to the fact that GPS readings can be faulty and are strongly affected by urban environments and poor signal quality. A closer inspection of the experimental site revealed the presence of a high-voltage station emitting strong electromagnetic fields, which can significantly degrade GPS accuracy. In addition, the experiments were conducted on a university campus surrounded by tall buildings, further weakening the GPS signal. While the throttle method is somewhat sensitive to the vehicle's battery charging level, it remains the most suitable option, particularly because it provides more responsive measurements at a higher rate compared to GPS, which only outputs readings at a 1-second interval.

4.4. Perception module

For the implemented perception module, a layered pipeline composed of number of sequential stages has been structured with four consecutive stages. And as an initial step, the layered pipeline initializes the stereo camera, extracts, and stores its intrinsic parameters, such as the baseline, focal length, and principal points.

4.4.1. Stage 1: Object detection, classification, & tracking

In this stage, all surrounding objects are detected, bounded in 2D boxes, and classified with one of the trained classes. In this work, the architecture is only trained to limited classes relevant to the architecture, namely the "*car*" class. Furthermore, information about the class

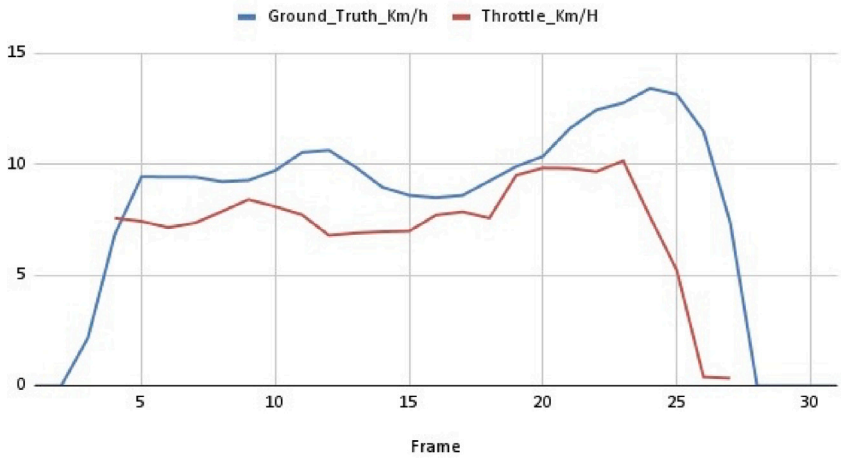


Fig. 7. TV's Speed Profile Obtained from Throttle-calibrated Measurement Vs. Mobile GPS Sensor.

Table 1
Detection Benchmark Summary (100-frame mean ± std). Latency is used as the main selection criterion; mean confidence is reported for descriptive purposes only.

Model	Latency (ms)	Avg. conf.
YOLOv8-n	18.4 ± 3.5	0.70 ± 0.05
Faster R-CNN	127.8 ± 3.7	0.22 ± 0.03
SSDLite320	101.2 ± 12.1	0.07 ± 0.01

scores and persistent ID is added as output from the tracking. In the implementation of this stage, three different benchmark methods have been investigated. These methods are YOLOv8-n [43], Faster Region-based Convolutional Neural Network (R-CNN) [44], and SSDLite (a lightweight version of the Single Shot MultiBox Detector, SSD) [45]. To quantitatively assess the detector, the three methods have been examined on over 100 frames from a ZED SVO, logging per-frame latency and mean confidence. Since the task in this work is limited to single-class car detection in relatively simple scenes, and all three detectors are state-of-the-art architectures with well-documented accuracy on standard benchmarks, we focus on their runtime behavior on our hardware rather than re-benchmarking accuracy on our small test set. As shown in Fig. 8 and Table 1, YOLOv8-n achieves the lowest average inference latency of ≈18.4 ms (≈54 FPS), comfortably above our 10 FPS real-time target whereas Faster R-CNN and SSDLite run noticeably slower (around 7.8 FPS and 9.9 FPS, respectively). YOLOv8-n is therefore clearly the fastest option in our setup while still providing stable detections in this scenario. The mean confidence values reported in Table 1 are included only as descriptive statistics of how each model scores its detections and are not used as a substitute for accuracy. On this basis, supported by the inspection of the detections quality on our sequences, we select YOLOv8-n in ByteTrack [46] mode as the detector for the subsequent experiments.

4.4.2. Stage 2: Depth estimation

The aim of this stage is to estimate the depth of the detected vehicles (the PV) relative to the TV. For this stage, two different state-of-the-art pipelines have been investigated.

Pipeline 1: RAFT-stereo depth estimation. The first investigated method is Recurrent All-Pairs Field Transforms (RAFT)-Stereo [47]. RAFT works directly with the left and right images from the stereo camera. It takes both images as input and runs for 24 iterations to produce a high resolution disparity map d. This disparity map is comprised of point pairs $d(u, v)$, where each value corresponds to the shift between the left and right images. This disparity is then converted to depth map Z in which each point depth $Z(u, v)$ is computed using the below formula:

$$Z(u, v) = \frac{f B}{d(u, v)} \tag{1}$$

where f is the focal length of the camera and B is its baseline. This depth $Z(u, v)$ represents the distance between the camera frame and the point in the disparity map $d(u, v)$. The obtained point depth is further calibrated to compensate the 15° tilt of the camera mounted on the vehicle and illustrated in Fig. 6. The computed depth map points values are then passed through several filters to smooth the returned values including median filtering, Gaussian smoothing, and Bilateral filtering for edge preservation. After estimating the depth of the PV's center point (u, v), the final vehicle's coordinates relative to the camera frame has been formulated:

$$X = \frac{(u - c_x)\, Z}{f_x}, \quad Y = \frac{(v - c_y)\, Z}{f_y}, \quad Z = \text{depth}. \tag{2}$$

where (c_x, c_y) are the principle point offsets and (f_x, f_y) are the camera's focal lengths. Finally, these coordinates are then translated from the camera frame to the vehicle frame to represent the position of the PV relative to the TV.

Pipeline 2: ROI-based segmentation depth estimation. For the second pipeline implementation, it is decided to work with segmented objects rather than just using bounding box as detection output. The main aim is to reduce the per-frame pixel throughput with the target of increasing the perception rate. Each detected box defines a Region-of-Interest (ROI), which is resized to the segmentation model's input (e.g. $512 \times 512px$). By focusing only on vehicle ROIs, the per-frame pixel throughput is reduced by up to 85%. Three different backbones are compared and evaluated which are DeepLabV3 with Residual Neural Network (ResNet)-50 and ResNet-101 encoders [48], Fully Convolutional Network (FCN)-ResNet-50/101 [49], and YOLOv8s-Seg variant [50]. DeepLabV3 incurs an average latency of approximately 11.2 ± 1.1 ms per ROI, FCN is the fastest at roughly 9.6 ± 0.9 ms, and the integrated YOLOv8s-Seg head runs at about 20 ± 4.3 ms. These results guide our choice of segmentation backbone under different real-time constraints: FCN for maximum throughput, DeepLabV3 for

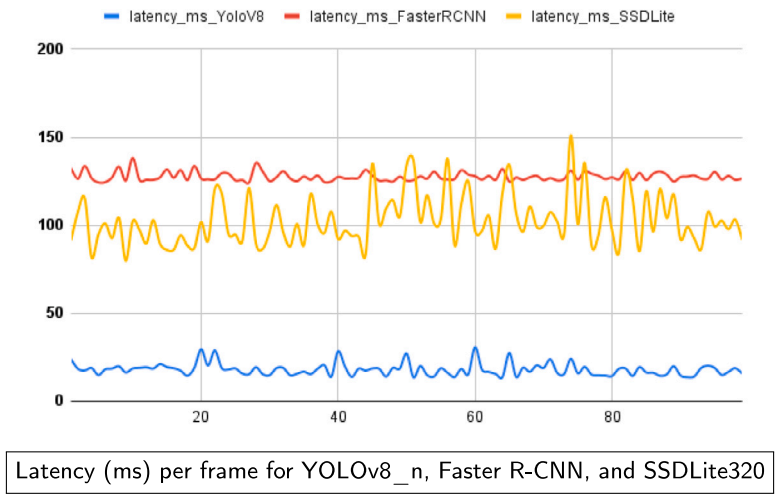
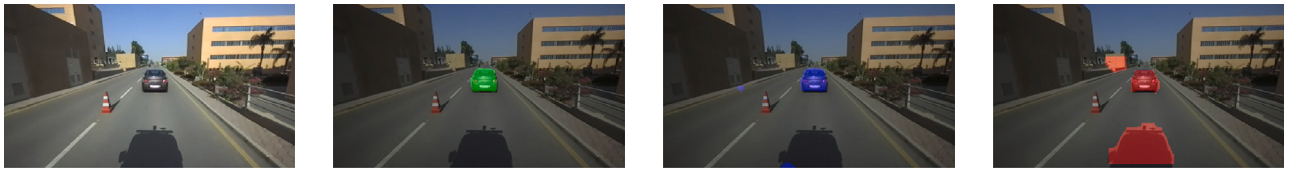


Fig. 8. Frame-by-frame inference latency for each detector.



(a) Input Frame. (b) DeepLabV3-ResNet50. (c) FCN-ResNet50. (d) YOLOv8s-Seg.

Fig. 9. Qualitative ROI-based segmentation outputs for the three backbones.

highest mask quality, and YOLOv8s-Seg when unified inference is preferred. The investigated segmentation models are evaluated based on qualitative mask quality and the per-frame latency on 100 frames of ZED 2 left-view video. Each model outputs a binary mask per ROI, which is threshold and upsampled to the original box size. Then, the per-model Intersection over Union (IoU) and segmentation latency are measured to guide our selection under the 10–30 FPS constraint. Fig. 9 presents a side-by-side comparison of the three segmentation backbones on a representative vehicle ROI. It can be observed that the green DeepLabV3 mask offers the most complete coverage of the vehicle's silhouette, effectively capturing the full extent of the car body. In contrast, the blue FCN mask adheres more closely to the car's geometric outline but suffers from over-segmentation of non-vehicle elements (e.g., scene clutter), leading to false positives in the background. Finally, the red YOLOv8s-Seg mask shows irregular, "jagged" contours and even misclassifies the vehicle's shadow and nearby wall as part of the vehicle, indicating both under- and over-segmentation artifacts. The per-frame inference time for each segmentation model was measured over 100 ZED frames, and the latency traces are shown in Fig. 10.

Based on these results, it is decided to proceed with the DeepLabV3-ResNet50 semantic segmentation model [51] to produce a binary mask for cars, and overlays the mask. Once the PV is being detected and segmented, its depth is being estimated however in this method, the disparity map d is obtained from the ZED's Software Development Kit (SDK) that computes d by matching each left-pixel to its counterpart in the right image. Afterwards using Eq. (1), the vehicle depth Z is computed.

4.4.3. Stage 3: Object tracking & kinematic estimation

This stage is important to track the historical kinematical estimations of the object. This helps in extracting more kinematical properties as the object speed and yaw angle. An object tracker class is created to store the history of each object detected for a few frames, even after its disappearance from the camera view, to enhance the tracking of the objects. For this work, the most critical kinematical feature to be extracted is the PV speed, which is crucial for the safety features extraction. Two different trackers and kinematical estimation techniques are used depending on the implemented pipeline discussed in Section 4.4.2. For Pipeline 1, after computing the PV depth, its speed is calculated using the saved historical kinematical information as follows:

$$v_{\text{obj}} = -\frac{v_x X + v_z Z}{\sqrt{X^2 + Z^2}} \tag{3}$$

where v_x and v_z is the change of the PV's position in relation to the time in the x and z directions respectively, while the X and Z are its coordinates in the 3D camera frame.

For Pipeline 2, the implemented tracker class tracks per-object depth and velocity with Exponential Moving Average (EMA) smoothing technique. For each track, it maintains a rolling window of recent depth measurements and computes a smoothed depth as the mean over that window. An anchor-based velocity is calculated across the window, then an exponential moving average with coefficient $\alpha = 0.3$ at 20 FPS is applied.

4.4.4. Stage 4: Safety features extraction

After estimating the depth and speed of the PV, TTC and THW are computed as follows:

$$\text{TTC} = \frac{|d|}{|v_{\text{rel}}|}, \quad \text{THW} = \frac{d}{v_{\text{TV}}}, \tag{4}$$

where d is the relative distance between the TV and PV, $v_{\text{rel}} = v_{\text{TV}} - v_{\text{PV}}$ is their relative velocity, and v_{TV} is the TV's velocity. TTC represents the time remaining before a potential collision if both vehicles maintain their current speeds, while THW indicates the temporal gap between the two vehicles. These values are transmitted via Transmission Control Protocol/Internet Protocol (TCP/IP) to the communication module and subsequently relayed to the prediction module.

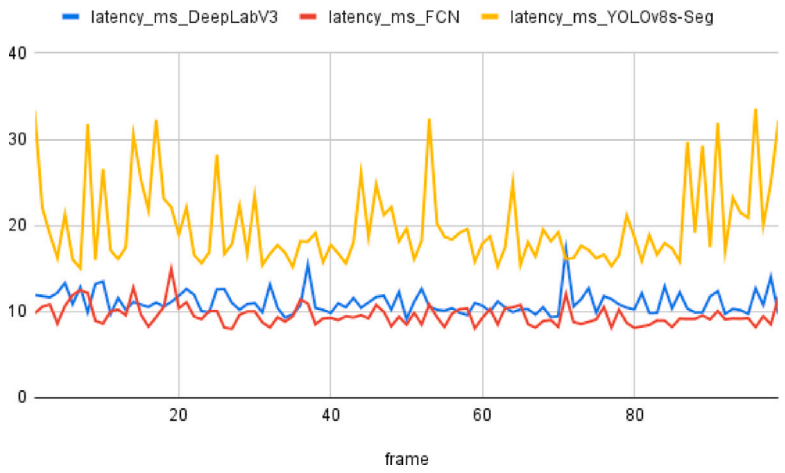


Fig. 10. Frame-by-frame inference latency for DeepLabV3, FCN, and YOLOv8s-Seg (100 frames).

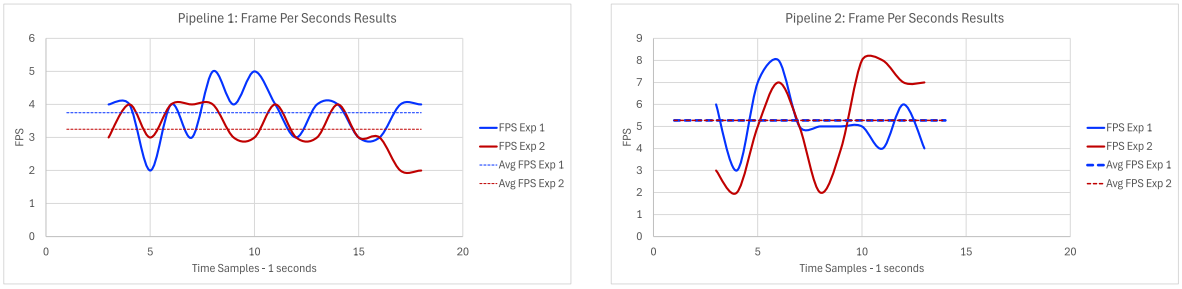
4.5. Results

The performance of the perception module is assessed under both investigated pipelines. Both pipelines have been experimented twice in the full integrated system with the prediction module. Both experiments are basically under the same scenario. However, it is repeated to prove the repeatability and robustness of the overall system. It is important to note that even if the experiment is basically the same, it is impossible to yield the same exact results since 2 of the 3 vehicles are human-driven. That means, they will have different velocity and acceleration profile, hence slightly different results. In this section, we will mainly focus on illustrating the quantitative assessment of the perception module only. First in terms of the speed of the model, Fig. 11 shows the variation in FPS throughout the experiment, which lasted approximately 15–17 s. It can be seen that pipeline 2 is running with higher FPS with maximum of 8FPS and average of 5.3FPS compared to maximum of 5FPS and average of 3.75FPS by pipeline 1. However, it is important to note that these results are recorded for the online testing of the experiments with the laptops onboard of the vehicle with no power charging. These average FPS values significantly increase in the offline testing to reach 10 – 15FPS which was considered as a selection criteria for the used models. It is important to note that in both experiments for pipeline 2, the average FPS value is consistent unlike the values obtained by pipeline 1. From this we can conclude that pipeline 2 is faster than pipeline 1 due to the lightweight models and the reduction of the detected object by using the ROI-Segmentation model. To assess the estimators and the features extracted from both pipelines, two experiments with similar experimental setups were conducted. By comparing the estimated distances in experiment 1, it was found that the depth values from both pipelines were close to each other over the experiment duration. As shown in Fig. 12(a), the vehicles maintained an approximate gap of 10 m.

Moving to analyzing the estimated PV velocity using both trackers, Fig. 12(b) shows that in pipeline 1 the vehicles moved with higher speeds than in pipeline 2. Still, in both cases the vehicles kept almost the same average gap of about 10, m. This led to a riskier driving profile in pipeline 1, as confirmed by the lower TTC values in Fig. 12(c) compared to pipeline 2, where the lower speeds indicated less risk. It can also be seen from Fig. 12(c) that the minimum TTC occurred at the end of the experiment, which corresponds to the moment when the PV was braking and the TV was making the risky maneuver to avoid a collision.

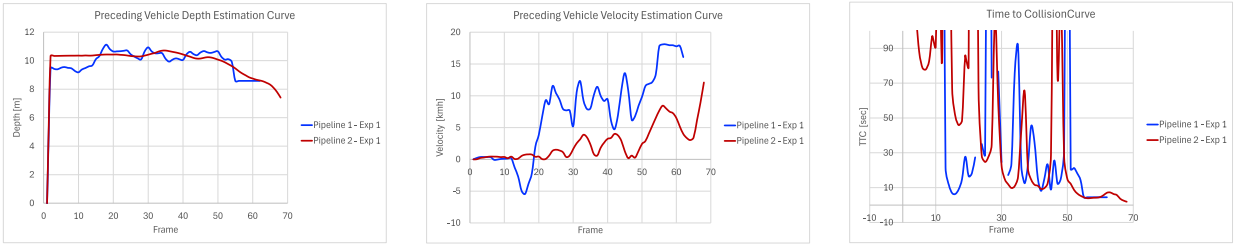
4.6. Findings

- **Computational Performance:**
  - RAFT-Stereo required significant processing power, achieving average 3.75FPS during the online experiment, while ROI-based segmentation was more efficient average 5.3FPS. That confirms the importance of working with specified ROI.
  - Offline runs confirmed that hardware load was the bottleneck, as both pipelines reached 10 – 15FPS with stable power.
  - This highlights the importance of hardware-aware pipeline design for scalable real-time deployment.
- **Depth Estimation Observations:**
  - The ZED 2 stereo camera delivered reliable depth within the 0.3 – 12 m range, with high accuracy (<1% error at 3 m).
  - Beyond 12 m, measurements became increasingly noisy, reinforcing the need to prioritize mid-range detection in cooperative perception.
- **Environmental Insights:**
  - Lighting strongly influenced stereo matching: shaded or evenly lit scenes produced stable results, while direct sunlight and glare introduced distortions as shown in Fig. 13. This will be considered as a challenge specifically for sunny countries to experiment during day light and of course it is always a challenge to experiment at night using vision-based perception. This only gave us the sunset time to be the most suitable time for experimentation.
  - GPS readings were occasionally disturbed by electromagnetic interference like in the near high-voltage stations areas or narrow urban areas with high buildings, making throttle-based velocity estimation a more practical alternative in such contexts. Also, taking direct measurement is also a challenge since it is always affected by the battery charging level.
- **Camera and Sensor Behavior:**
  - ZED 2 performance was sensitive to focus drift and calibration issues in extreme lighting conditions.
  - Maintaining stable results required scenario-aware deployment to avoid scenarios as reflective ground surfaces and testing during mild-light periods.
- **Pipeline-Level Insights:**
  - Pipeline 2 demonstrated more consistent FPS and smoother perception under resource-constrained settings.



(a) Pipeline 1. (b) Pipeline 2.

Fig. 11. Comparison between the investigated pipelines Performance in terms of the FPS during online Testing.



(a) PV Depth. (b) PV Velocity. (c) TTC.

Fig. 12. Perception Pipeline 1 and 2 results (a) The Depth estimation of the PV using RAFT vs ZED SDK, (b) The Velocity Estimation of the PV based on the estimated depth and relative velocity, and (c) The final estimated Time-to-Collision between the TV and PV.

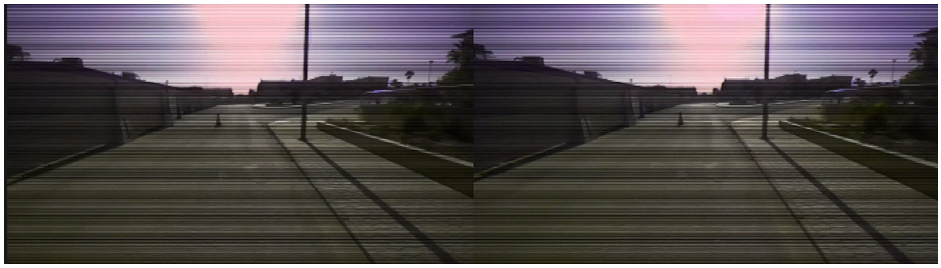


Fig. 13. Distorted camera view.

- Pipeline 1 produced richer kinematic details but was more vulnerable to dropped frames and delays in sudden object motion.
- Both pipelines successfully enabled computation of TTC and THW, supporting cooperative perception for safety-critical applications. However, they are also function in the accuracy of the estimations which encounter challenges already.

5. Communication module

Reliable communication between perception and prediction modules is essential for real-time cooperative driving. In this study, we evaluated two strategies: direct Vehicle-to-Vehicle (V2V) links for minimal latency, and a relay-based architecture for improved modularity and scalability.

5.1. Direct V2V communication

The first strategy was based on direct V2V communication, where the perception module, running on a laptop *(Device 1)* in the TV that sends the extracted features directly to the prediction module hosted on either a Jetson Nano or a Raspberry Pi in the EV. In this configuration, only the communication delay was measured, while the prediction time was excluded. Although V2V communication allowed for direct exchange of perception information, it revealed several limitations. The perception and prediction modules were tightly coupled, such that any modification in the perception pipeline required corresponding adjustments in the prediction module. This limitation becomes even more critical if the concept is generalized to a larger number of vehicles: imagine a situation with many perception vehicles transmitting to many prediction devices. A single change in the perception pipeline would then propagate to all connected prediction modules, creating significant overhead and limiting flexibility. Furthermore, the direct wireless link was not always reliable. When the distance between vehicles increased, the Wi-Fi connection occasionally dropped, leading to communication interruptions and loss of data exchange.

5.2. Relay-based communication

To address these issues, a second strategy was implemented using a relay-based architecture, where a laptop *(Device 2)* acted as an RSU. In this case, the perception module *(Device 1)* transmitted

numerical features to the RSU *(Device 2)*, which handled preprocessing and conversion into linguistic categories before forwarding the reformatted data to the prediction device. These linguistic categories (e.g., Low/Medium/High risk derived from TTC and THW) evolve at a slower time scale than the normal numerical signals, since they only change when the overall traffic situation transitions between different configurations. In our recorded scenarios, these categories often remained unchanged for hundreds of milliseconds to several seconds, and only updated when the surrounding traffic changed significantly (for example, when the PV started braking). As a result, the prediction module does not need new messages every few milliseconds, but only when the underlying risk or contextual state changes. This relay-based design decoupled perception from prediction, allowing each module to focus solely on its intended function: the perception module concentrated on sensing and feature extraction, the relay processed and formatted the data, and the prediction device executed inference. The modular structure ensured that changes in one module did not propagate to others, improving system maintainability. More importantly, this modularity scales better to multi-vehicle environments. Instead of requiring updates across a large number of prediction modules whenever the perception pipeline changes, only the relay would need to be updated, while all connected vehicles continue to receive standardized information. In this way, perception and prediction are fully decoupled, and modifications in one do not force changes in the other. Furthermore, placing a relay between vehicles stabilized the communication link by reducing the likelihood of dropped connections, since the relay acted as a fixed and reliable communication point. Finally, this approach offers a path toward scalability. In a multi-vehicle scenario, it would be impractical for a single perception module to establish and maintain direct links with multiple prediction devices. Instead, the relay can process the data once and distribute it to the appropriate vehicles. While our experiments involved only one perception and one prediction vehicle, we expect that this design would scale more naturally to larger traffic environments as shown in [52,53].

5.3. Experimental setup and delay measurement

All communication was established over the TCP/IP protocol using the available Wi-Fi interfaces on each device. The Raspberry Pi 3B+ is equipped with an onboard Broadcom Wi-Fi chipset limited to 802.11n at 2.4 GHz. The Jetson Nano 2 GB employs an external Universal Serial Bus (USB) Wi-Fi dongle supporting 802.11ac at 5 GHz, which enables faster and more stable links. The client device *(Device 1)* is a laptop running Windows, while the server laptop *(Device 2)* runs Linux; both laptops feature integrated dual-band adapters with 802.11ac/ax capability at 5 GHz. All communication in this work was established using the TCP/IP protocol. To quantify the communication delay, a Round-Trip Time (RTT) measurement procedure was employed. In this approach, a device sends a signal to the other device, which immediately echoed it back. The measured RTT in that case represents the total travel time of the message between the two devices in both directions, and the one-way communication time was obtained by dividing this value by two. This procedure was applied consistently across all tests. It was used for the direct V2V communication between the perception and prediction modules, as well as for the relay-based strategy, which includes both the communication from the perception laptop *(Device 1)* to the relay *(Device 2)* and the communication from the relay to the prediction module.

5.4. Results

Table 2 summarizes the measured communication overhead between the different device pairs in the experiment. The first column corresponds to the communication between the laptop client *(Device 1)*, which serves as the perception module, and the Raspberry Pi. This link required 4.0 ms per iteration. The second column represents the communication between *Device 1* and the Jetson Nano, which took 3.25 ms per iteration. Both of these cases fall under the direct V2V strategy, where the Jetson achieved a 0.75 ms per-iteration advantage compared to the Raspberry Pi. In the relay-based strategy, the process changes. The perception laptop *(Device 1)* first communicates with the Linux laptop *(Device 2)*, which acts as the relay (3.5 ms per iteration). After processing and formatting the data, the relay forwards it to the Jetson Nano (3.25 ms per iteration). The total communication overhead for this two-step process is 7.25 ms per iteration. Although this is nearly double the latency of the direct V2V case, the relay-based approach (as previously mentioned) provides important benefits in terms of modularity and scalability, making it more suitable for multi-vehicle environments.

Table 2
Measured communication overhead between device pairs. Direct V2V communication required 4.0 ms (*Device 1*–Pi) and 3.25 ms (*Device 1*–Jetson) per iteration, while the relay-based path (*Device 1*–*Device 2*–Jetson) increased the time to 7.25 ms. Although the relay nearly doubles the delay, it provides the advantages of modularity and scalability.

Approach	*Device 1* - Pi	*Device 1* - Jetson	*Device 1* - *Device 2*
time ms/itr	4	3.25	3.5

To assess the impact of additional communication delay on the cooperative behavior, we introduced artificial delays of 100 ms and 200 ms in the communication link between the relay and the prediction device. In both cases, the outcome did not change with respect to the nominal case (7.25 ms average latency). This indicates that the system behavior is robust to moderate increases in communication delay on the order of 100–200 ms, which is consistent with the relatively slow evolution of the linguistic contextual categories discussed in Section 5.2.

5.5. Findings

- Direct V2V links achieved the lowest latency (3.25–4.0 ms), but tightly coupled perception and prediction, making the system inflexible to updates or changes. The real bottleneck is not milliseconds of delay but the system's scalability and flexibility.
- Wireless reliability in direct V2V was unstable; links dropped when the vehicle distance increased, making it unsuitable for consistent outdoor operation. Connection stability is therefore more critical than raw latency in real-world deployments.
- The relay-based approach introduced slightly higher latency (~7.25 ms) but enabled full decoupling of perception and prediction, which is critical for modularity and maintainability.
- Relay communication scales better to multi-vehicle environments: only the relay requires updates when perception changes, avoiding the rapid growth of direct links as the number of vehicles increases.
- Direct V2V links were unstable over distance, with frequent connection drops. The relay, acting as a fixed RSU, provided more robust and reliable communication.
- For multi-vehicle environments, the relay design is preferable since only the relay must be updated when the perception pipeline changes, avoiding the impracticality of maintaining many direct links.
- Device capabilities influenced communication performance: the Jetson Nano with 802.11ac (5 GHz) outperformed the Raspberry Pi with 802.11n (2.4 GHz), showing that communication quality is shaped not only by architecture but also by Wi-Fi hardware and frequency bands.
- Despite nearly doubling the delay, the relay kept communication overhead within practical real-time limits. The extra relay overhead is negligible in practice given the safety margins gained

from modularity and stability. In conclusion, the relay-based approach offers the most practical balance of scalability, robustness, and modularity, making it better suited than direct V2V for cooperative perception in real-world deployments.

- Since the relay exchanges low-rate linguistic categories rather than high-rate raw sensor streams, the prediction module does not require updates every few milliseconds. The offline sensitivity analysis with additional delays of 100 ms and 200 ms showed that the predicted maneuver and the ego-vehicle behavior remained practically identical to the nominal case (7.25 ms average latency), confirming that the cooperative lane-change behavior is robust to moderate increases in communication delay on the order of 100–200 ms (25x compared to nominal case).

6. Prediction module

6.1. KGE training

A knowledge graph is a structured representation of information where entities are expressed as nodes and their semantic relations are expressed as edges, typically in the form of triples of the type <subject, relation, object>. In the context of vehicle intention prediction, such a representation allows numerical data describing vehicle motion to be translated into semantic facts, such as "the vehicle has high-risk TTC with the preceding car" or "the lateral acceleration is close to zero". Unlike raw numerical values, these semantic descriptors can be directly interpreted by humans and can serve as the foundation for reasoning systems. In order to use these graphs for prediction tasks, knowledge graph embeddings are introduced. Knowledge graph embedding models project the entities and relations of the graph into a continuous low-dimensional vector space while preserving semantic proximity. Entities and relations that are strongly associated in the data are placed closer together in the embedding space, whereas unrelated elements are placed further apart. This transformation makes it possible to apply machine learning and probabilistic reasoning on top of symbolic knowledge.

The prediction pipeline builds on twelve input features that describe the kinematic and contextual state of the scene. These include the lateral velocity and acceleration of the TV; TTC with the preceding, left-preceding, right-preceding, left-following, and right-following vehicles; lane identifier; position within the lane; THW with the PV; the lane with the highest frontal gap; and the lane with the highest attraction score. Because these variables are raw numerical quantities and not inherently interpretable, they are first converted into linguistic categories using thresholding criteria. Interpretability here refers to how easily a user can understand why the model produced a specific result based on the input. For example, a model receives numerical inputs, such as a lateral velocity of 0.1 m/s and a TTC of 4 s, and it predicts a left lane change. In another case, the input is linguistic, stating that the vehicle is moving straight. The time to collision is marked as high risk, and the model predicts a left lane change. The second example is more interpretable. It is easier for a human to follow up and understand why the model made that prediction, and that is because the inputs are more understandable. So, interpretability means making the input–output relationship easier to follow for users. This conversion from numerical values to linguistic categories relies on simple threshold checks and has negligible computational (almost zero) cost. These threshold values are obtained according to two principles. For features such as lateral velocity, the distribution of the data was analyzed, the observed values followed a normal distribution, and by applying the mean and standard deviation, the values were divided into three categories: *moving left*, *moving straight*, and *moving right*. Other features, such as TTC, were categorized into *high risk*, *medium risk*, and *low risk* according to thresholds reported in the literature. After this conversion, the categories are reified into triples (<subject, relation, object>) and stored in a tabular form in a Comma-Separated Values (CSV) file. Each row of the file corresponds to a fact that describes the scene. For example: "vehicle, LATERAL_ VELOCITY_ IS, movingLeft", "vehicle, TTC_ WITH_ PRECEDING_ VEHICLE_ IS, highRisk", and "vehicle, INTENTION_ IS, leftLaneChange". These triples are constructed according to a formal ontology that defines all entities and possible relations. Further details on the knowledge graph construction and complete description of the ontology and its instances are available in [36]. For embedding, the TransE model was used via the AmpliGraph library. The embedding size was set to 100, the optimizer was Adam with a learning rate of 0.0005, and training used a batch size of 10,000. A self-adversarial loss function was applied with negative sampling at a $5:1$ ratio, and early stopping monitored using the Mean Reciprocal Rank (MRR).

6.2. Bayesian inference

Once the graph is constructed and embedded, Bayesian inference is carried out on top of the learned embeddings to perform the prediction. The task is to compute the probability of each possible maneuver hypothesis H given the observed evidence E formed by the twelve linguistic inputs. This follows Bayes' theorem:

$$P(H \mid E) = \frac{P(H) \cdot P(E \mid H)}{P(E)} \tag{5}$$

Here, $P(H \mid E)$ is the posterior probability of a maneuver given the evidence. $P(H)$ is the prior probability of the maneuver, reflecting its baseline likelihood before considering current evidence, and is informed by statistical regularities captured in the embeddings. $P(E \mid H)$ is the likelihood, i.e., the probability of observing the evidence if the maneuver were true. Finally, $P(E)$ normalizes the result so that the posteriors over all hypotheses sum to one.

An example that illustrates the reasoning can be like the following. Suppose the system considers three hypotheses: lane keeping, left lane change, and right lane change. Initially, the prior favors lane keeping, since it is generally more common. If the TTC with the PV is very small (high risk), the likelihood of lane keeping decreases, while the likelihood of either lane change increases. The posterior $P(H \mid E)$ then shifts accordingly. Next, if the TTC with the right-following vehicle is also high risk, while that with the left-following vehicle is low risk, the evidence weakens the probability of a right lane change and strengthens that of a left lane change. At this point, the posterior favors the left lane change hypothesis. Through this sequential updating process, all available inputs (lateral motion,TTC values, THW, lane gaps, and attraction scores) contribute to refining the probability of each maneuver. The final posterior distribution represents the combined effect of all evidence, and the maneuver with the highest posterior probability is selected as the predicted intention of the TV. The strength of this approach lies not only in its predictive ability but also in its interpretability. Because each step of the inference process is explicit, one can always trace back how the evidence led to the final decision: the prediction of a left lane change can be justified by pointing to the high-risk TTC with the PV, which reduced the probability of lane keeping, and to the low-risk conditions on the left, which increased the probability of a leftward maneuver. In this way, the Bayesian framework ensures that predictions are both accurate and transparent, qualities that are essential for safety-critical applications in autonomous driving.

6.3. Results

The model was trained on both the highD dataset (safe lane changes) and the CRASH dataset (risky lane changes). It achieved an f1-score of 95% and 90.0% for safe maneuvers with two and four-second anticipation horizon respectively, and 95% and 91.5% for risky maneuvers under the same horizons. When both safe and risky maneuvers were combined in a mixed setting, the model reached an f1-score of 95% and 89.4%, showing consistent performance across scenarios.

Compared with prior studies, our results are competitive with state of the art deep learning models that predict only safe lane changes. For example, transformer-based approaches [19] on highD reached 98%–95% f1-scores, and XGBoost model used by [] on highD reached 99%–92% f1-scores. Both models predicted short horizons (0.5–2 s), lacked interpretability and transparency, and did not consider risky situations and did not extend their horizon to 4 s. Our model, while slightly lower at very short horizons, maintains high accuracy up to four seconds, provides traceable Bayesian reasoning, and uniquely addresses risky near-crash lane changes. These results demonstrate that the approach not only performs well in standard safe scenarios but also extends predictive capability to high-risk conditions, making it more practical for safety-critical driving applications.

6.4. Hardware validation

In terms of hardware validation, the model was evaluated on three different computing devices: a Lenovo laptop equipped with an NVIDIA RTX 4090 GPU (16 GB VRAM, 32 GB RAM, and a 1 TB SSD), a Raspberry Pi 3 Model B+, and an NVIDIA Jetson Nano with 2 GB of memory. On the laptop, the model was tested in an online setting and achieved a speed of approximately three to four frames per second. However, it was not feasible to run the online version of the model directly on the Raspberry Pi or the Jetson Nano due to their more limited computational capacity.

To overcome this limitation, we exploited the fact that the model operates on linguistic variables. Instead of executing the full model online, we precomputed all feasible combinations of the twelve input features and their corresponding predictions. For example, the feature "TTC with the PV" can be classified into three categories: low risk, medium risk, or high risk. Considering similar categories for all twelve features, a large set of possible input combinations can be generated. Some of these combinations, however, are not physically feasible. For instance, if the TV is already in the leftmost lane, it is not possible to assign the category "left lane has the highest frontal gap", because no left lane exists. After removing all such infeasible cases, approximately 212,000 valid combinations remained. These combinations were first stored in a CSV file and loaded into a Pandas DataFrame. In this setup, when a prediction is required, the system searches by filtering row by row: each of the twelve inputs is matched against its corresponding column (for example, *Input1 == x, Input2 == y, ... Input12 == z)* until the single row that satisfies all twelve conditions is found, and the corresponding prediction is returned. This type of search behaves like an *O(n)* operation, where *n* is the number of rows in the file, because as the CSV grows larger, more records must be checked before the match is found. This is why prediction with CSV search slows down as the number of stored combinations increases. The first row in Table 3 reports the computational time, in seconds per iteration, of the three devices when using this CSV file approach. As expected, the laptop achieved the lowest computational time (0.075 s per iteration) due to its superior hardware, but this entry is mainly included for comparison. Since the final system must run on an embedded platform, the relevant comparison is between the Raspberry Pi and the Jetson Nano. From the results, it is clear that the Jetson Nano provides better computational performance than the Pi when using the CSV-based search, Jetson outperformed the Pi by about 31%. To improve efficiency, the combinations were also stored in a lookup table, implemented as a dictionary structure. In this representation, all twelve inputs are concatenated into a single string key at the time of storage. At inference time, the twelve current linguistic inputs are concatenated in the same way, and the dictionary is queried directly using this key to return the prediction. This avoids the need for sequential filtering across twelve separate columns, replacing it with a single direct lookup. From a computational standpoint, the dictionary approach has constant-time complexity *O(1)*, meaning that the search time does not increase with the number of stored combinations. The results of the lookup approach, shown in Table 3, indicate much better computational time across all devices. The laptop reduced its computational time from 0.075 s to $1.4e^{-7}$ s, corresponding to a speed-up of ($10^5\times$). The Raspberry Pi dropped from 1.16 s to $9.4e^{-6}$ s, and the Jetson Nano from 0.80 s to $2.4e^{-6}$ s. This means the lookup approach provided a speed-up of nearly ($10^5\times$) on the Pi and over ($3 * 10^5\times$) on the Jetson, confirming that lookup tables make real-time deployment feasible. Moreover, the Jetson consistently outperforms the Pi, making it the preferred option for in-vehicle integration. To further analyze scalability, the experiment was repeated with a reduced dataset representing only two-lane roads, resulting in approximately 18,000 feasible combinations instead of 212,000. In this smaller configuration, the CSV search became faster than in the previous case as shown in the 3rd row of Table 3, as expected due to the reduced number of rows to filter. However, the lookup table continued to outperform the CSV approach and, as anticipated, maintained nearly identical performance to the larger configuration as shown in the 4th row of the table. This confirms that lookup-based prediction provides stable speed, while CSV search depends heavily on dataset size.

Table 3
Computational time (seconds per iteration) for CSV and lookup search across devices. Lookup tables consistently achieve much lower latency than CSV search, and maintained performance regardless of dataset size.

Number of combinations	Approach	Laptop	Pi	Jetson
212K	CSV	0.075	1.16	0.80
	LOOKUP	$1.4e^{-7}$	$9.4e^{-6}$	$2.4e^{-6}$
18K	CSV	0.0067	0.11	0.070
	LOOKUP	$1.4e^{-7}$	$9.4e^{-6}$	$2.4e^{-6}$

6.5. Findings

- Unlike many black-box deep learning approaches, the use of linguistic information along with Bayesian reasoning ensures interpretability and transparency, making it suitable for safety-critical driving applications.
- The laptop offered the highest efficiency (fastest execution), but this serves only as a comparison baseline and is not intended for real in-vehicle deployment.
- Among embedded devices, the Jetson Nano consistently outperformed the Raspberry Pi (by ≈31% under CSV search and by orders of magnitude under lookup table), making it the preferred option for onboard integration.
- The CSV-based search was computationally heavy (up to 1.16 s/iteration on the Pi). By contrast, lookup tables reduced latency by $\approx 10^5\times$ to microseconds, achieved constant-time performance ($O(1)$), and maintained stable inference regardless of dataset size (tested at ∼212k vs. ∼18k combinations). This enables real-time and scalable cooperative prediction.
- With prediction active, the EV anticipated the TV's maneuver ∼4 s before the lane change, yielding smoothly and avoiding abrupt braking. Without prediction, the TV was forced into sharp braking, confirming that predictive reasoning directly improves safety and comfort in cooperative driving.
- In conclusion, the Jetson Nano with lookup tables offers the best trade-off for real-time, interpretable prediction on embedded systems, directly supporting safer and smoother cooperative lane-change interactions.

7. Planning and control module

This section addresses the implementation details related to the control of the EV. In the upcoming subsections we would discuss the motion planning details, afterwards the sensors used in the EV setup will be discussed and finally the control implementation details are presented.

7.1. Motion planning

Vehicular motion planning is required to be done for both longitudinal (velocity) and lateral (steering) motions. In this study, the scope of the EV motion considered requires it to be in a specific direction in its lane, without the need to shift lane. Hence, for a state-feedback lateral control was implemented maintaining the vehicle's heading constant in the same direction. To that end, constant feedback of the heading of the vehicle is provided, based on which minor incremental steering angles could be applied (when needed) to maintain the heading constant.

However, in this study, the longitudinal motion of the EV is necessary to be controlled in an online manner, which is achieved via manipulating the speed profile of the vehicle. The EV used in this study is a scaled 1:4 vehicle following the body dimensions of a BMW x3 vehicle. The vehicle achieves a maximum speed of 1.5 m/s (around 5.4 km/h). The speed of the vehicle is controlled via manipulating the Pulse-Width-Modulation (PWM) method available on the low-level controller onboard, from 0%–100%. For the purpose of the experiments in this study, it is required to control the motion of the vehicle based on one of 3 states. In each time step, the vehicle receives the desired state from the prediction module, and updates the desired PWM accordingly. Firstly, the (accelerate) state, for which the vehicle's velocity PWM is increased incrementally by 4% each sampling time (100 ms). Secondly, the state (decelerate), in which the vehicle is required to slow down gradually. In this case the deceleration is done by decrementing the PWM values by 8% each time step. This is done in order to increase safety. Finally, the (stop) state aiming to terminate the vehicle's speed instantaneously in case of emergency, in which the vehicle decelerates by its maximum possible value by setting the desired PWM value to 0%.

7.2. Onboard sensors

To be able to control the vehicle's motion to follow the desired motion planned, a set of onboard sensors were used. For the longitudinal motion, velocity feedback is obtained from a single optical encoder coupled with the vehicle's rear wheel axis. This 600-pulse incremental rotary encoder, operating at 5 V, provides feedback to a low-level onboard processor dedicated to speed control. From the pulses generated by this encoder, both the vehicle's speed and traveled distance can be calculated and accordingly controlled. For the lateral motion, an onboard IMU sensor version MPU6050 is used providing feedback only for the yaw angle rate. If the value of the yaw angle increased beyond a predefined margin of ±2 degrees from its original value at the beginning of the experiment, the steering wheel motor (servo motor) is activated to compensate this deviation. It is worth to mention that effect of the vibration from the road on the deviation of the vehicle is amplified for scaled vehicles than real-sized ones, and thus proper control of heading in case of other experiments that include lateral motion is a must. The readings of the IMU is provided to another low-level processor dedicated for the heading control.

7.3. EV control

As mentioned in the previous subsections, lateral control in this study is done using a simple state-feedback control law aiming to maintain the heading of the vehicle constant. On the contrary, in this experiment, longitudinal control is essential to make sure that the vehicle is following the desired velocity profile. This is achieved by using a tuned PID control for speed control of the vehicle based on the following set of equations. The velocity error in each iteration is calculated using the following equation:

$$e(k) = v_d(k) - v_a(k) \tag{6}$$

where $e(k)$ is the velocity error in the current iteration, while the $v_d(k)$ and $v_a(k)$ are the desired and the actual velocity of the EV in the current iteration respectively. Accordingly, the proper action from the PID controller is calculated using the following equation:

$$x(k) = k_p e(k) + k_i \sum_{i=0}^{k} \frac{e(k) + e(k-1)}{2} T_s + k_d \frac{e(k) - e(k-1)}{T_s} \tag{7}$$

where $x(k)$ is the calculated PID output (before mapping), k_p, k_i and k_d are the proportional, integral and derivate PID gains that tune the effect of each error component. $e(k)$ and $e(k-1)$ present the velocity error in the current iteration and the previous iteration respectively, and finally T_s present the control loop sampling time (100 ms) in this study. After the PID control action is calculated, proper mapping is required for it to convert it to the PWM acceptable range of value [0-255] and then sent to the speed controller. The desired speed is defined from the motion planning module based on the previously explained states (accelerate, decelerate or stop).

7.4. Observations and findings

From the experiments conducted, the following observations could be extracted for the scaled EV:

- Lateral control performance was completely different between in lab experiments and the real road experiment, this was attributed to the vibration from the asphalt road which as propagated heavily to the IMU sensor and altered its readings drastically. Accordingly, proper damping for the IMU sensor is required for outdoor experiments.
- Localization sensors commonly used for outdoor experiments were not suitable for the EV localization. Sensors such as GPS were not able to accurately localize the scaled vehicle as the velocity range and distance traveled were small. Accordingly, utilization of other localization methods/sensors was needed. In this study we relied upon encoders however, it is possible also to rely on visual odometry (future work).
- During testing, the communication channel between the EV and other vehicles/devices was lost specially when the distance between them increased. At which case, the EV drifted during its motion, as it was not able to identify the proper course of action to follow. This was also observed for communication between onboard devices. These observations are highly impacted by many factors such as the presence of nearby devices, the distance from the RSU router, the charging level of the batteries supplying the devices, and others. To that end, having a fault-tolerant system for such modules is essential for better implementation of the demonstrated approaches specially in a Cooperative driving scenario.
- Another interesting observation was the crashing of some onboard processing units suddenly during the experiment when conducted during day time. After lots of investigation, it was attributed to the effect of high temperature. The thermal effect on the vehicles' sensors and processors have a huge effect on the stability of the systems, this has a huge impact on the future of implementing the approaches on vehicles operating in high temperature countries, such as the GUlf or Middle East and North Africa (MENA) region.
- Despite being a scaled vehicle, the EV successfully participated in an experiment with real-sized vehicles. Thus the scaled vehicles utilization provide a suitable platform that can be extended to demonstrate different cooperative vehicles behaviors interacting together in the future.

8. System level integration

Before moving on to the main experiment results, it is important to quantify how each design choice contributes to the end-to-end performance, Fig. 14 summarizes the latency budget for different configurations from the investigated perception, communication, and prediction

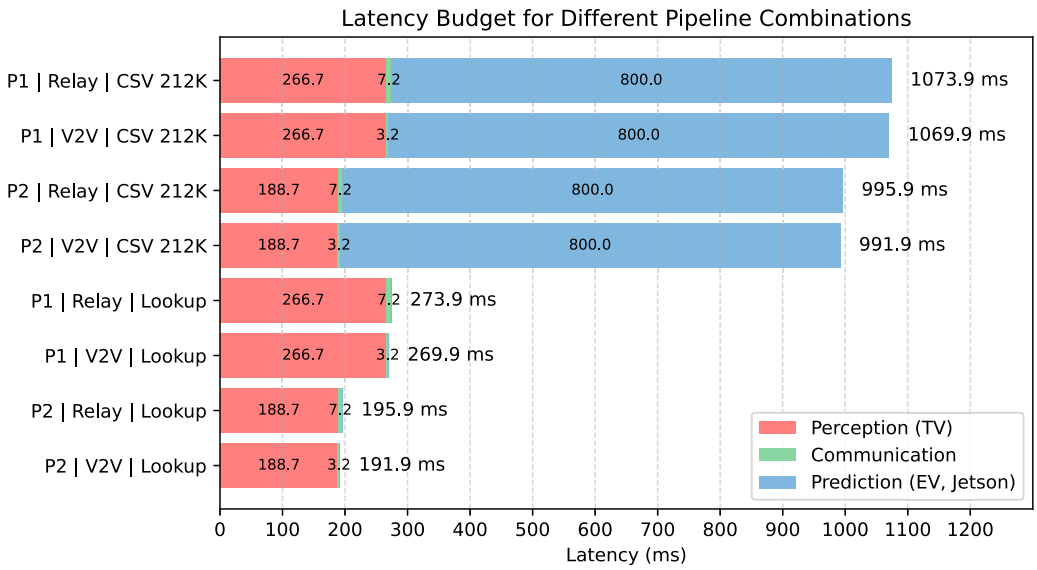


Fig. 14. Latency budget of the cooperative perception–communication–prediction pipeline for different design combinations. Each stacked horizontal bar corresponds to one configuration that combines Pipeline 1 (P1: RAFT) or Pipeline 2 (P2: ROI-based segmentation) on the TV, either direct V2V or relay-based communication, and either CSV search over 212k combinations or lookup-table prediction on the Jetson Nano. The bars show the average contribution of perception, communication, and prediction to the total latency. Lookup-based prediction runs in only $2.4 * 10^{-6}$ s (≈0.002 ms) on the Jetson, so its segment is not visible at this scale.

modules in this study. The perception stage compares Pipeline 1 (RAFT) with Pipeline 2 (ROI-based segmentation), the communication module compares direct V2V links with the relay-based architecture, and the prediction module contrasts lookup-table inference with CSV search over 212k combinations on the Jetson Nano. The four configurations using lookup tables achieve total latencies between approximately 192 ms and 274 ms, where the delay is dominated by perception, communication adds only about 3–7 ms, and the prediction time is negligible ($2.4 * 10^{-6}$ s ≈ 0.002 ms). In contrast, the four CSV-based configurations reach total latencies close to 1 s (0.99–1.07 s), clearly dominated by the prediction stage. Based on these results and the communication insights discussed in Section 5.5, the final integrated experiments were conducted with Pipeline 2 + relay-based communication + lookup-table prediction, which offers one of the lowest total latencies (196 ms) while benefiting from the modularity, scalability, and improved reliability of the relay architecture, even though direct V2V links are a few milliseconds faster.

After integrating all the modules, the two scenarios discussed in Section 3 were tested. In the first scenario, the prediction module was not integrated, while in the second scenario the prediction module was active in the system. To demonstrate the behavior of the integrated system, experimental trials were recorded and are available from multiple viewpoints. Table 4 provides links to the two scenarios (with and without prediction), captured from different camera perspectives. These videos clearly show the different interaction outcomes depending on whether the prediction module is active. More views can be accessed from the following playlist link: https://www.youtube.com/playlist?list=PLAeK3AuwxenFqIeAnKa9BLB8bDqEk8dNH

As described in Section 3, each of the two system-level scenarios (with and without prediction) was executed four times under the same conditions: two preliminary runs were used to verify the setup and communication, and two additional runs were recorded and used for analysis, giving eight executions in total. The curves reported in Fig. 15 and Fig. 16 correspond to one representative recorded run for each scenario. The remaining runs exhibited the same qualitative behavior, and the second recorded run showed similar quantitative profiles in terms of velocity and acceleration curves, with only small variations due to the human-driven PV and TV. These observations support that both the individual modules (as analyzed in the previous sections) and the integrated system behave consistently across executions, and that the effect of the prediction module on the interaction outcome is reproducible. The acceleration and velocity of both the EV and the TV were recorded, as shown in Figs. 15 and 16. For consistency, $t = 0$ corresponds to the moment when the TV crosses the lane marking, negative values indicate moments before the lane change, and positive values indicate moments after. When the prediction module was not integrated (dashed curves), the EV failed to anticipate the left-lane change maneuver of the TV. The EV continued straight ahead, which is reflected in the nearly constant acceleration of the red dashed ego-vehicle curve. This behavior resulted in a very small gap between the two vehicles, which did not permit the target to complete the lane change. Consequently, the TV was forced to brake sharply, visible as the sudden drop in the blue dashed target-vehicle curve close to the crossing moment. Such abrupt braking introduces potential safety risks (as the TV was about to collide with the PV) and discomfort for passengers. In contrast, when the prediction module was integrated (solid red curve for the EV and solid blue curve for the TV), the system anticipated the target's left-lane change about four seconds before the TV reached the lane marking. The EV then began to yield smoothly, visible in the gradual decrease of its acceleration and velocity curves. As the EV comes to a stop, its acceleration gradually returns to zero, confirming a controlled and safe maneuver. This behavior created enough space for the TV to merge without braking. Accordingly, the TV's acceleration and velocity remained nearly constant, showing no sudden deceleration. The velocity profiles confirm this behavior. Without prediction (dashed curves), the EV maintained a nearly flat velocity profile, while the TV's velocity dropped abruptly when it could not merge. With prediction (solid curves), the EV smoothly reduced its velocity to zero, as seen in the steady downward slope of the solid red curve, while the TV maintained its velocity without a sharp decrease, indicating that it could merge safely.

9. Discussion and conclusion

9.1. Discussion

9.1.1. Role of cooperative prediction

This study examined a cooperative perception–prediction architecture for lane-change anticipation through real hardware deployment.

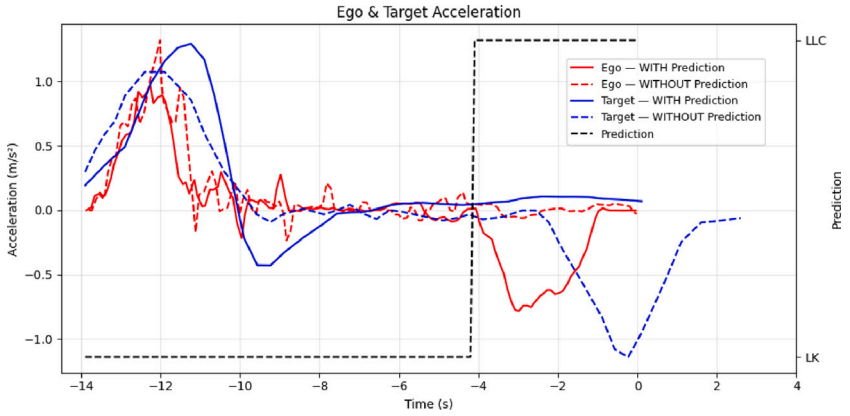


Fig. 15. Acceleration profiles of the EV (red) and the TV (blue) with and without the prediction model. The black dashed line indicates the maneuver prediction, and the gray vertical dashed line denotes the crossing moment at $t = 0$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

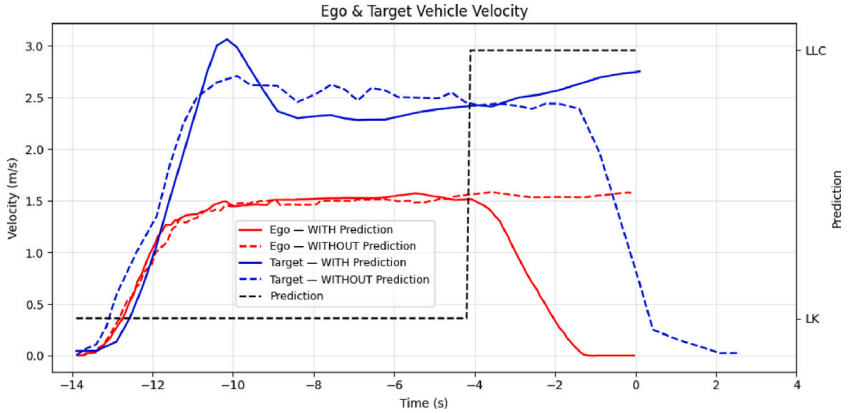


Fig. 16. Velocity profiles of the EV (red) and the TV (blue) with and without the prediction model. The black dashed line indicates the maneuver prediction, and the gray vertical dashed line denotes the crossing moment at $t = 0$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Video links showing hardware validation results, with and without integrating the prediction module.

Scenario View	Link
WITH Prediction (View 1)	https://youtu.be/p4LKoaUsZpc
WITHOUT Prediction (View 1)	https://youtu.be/BFVqhoT5Lck
WITH Prediction (View 2)	https://youtu.be/8Dc6Agxgn8M
WITHOUT Prediction (View 2)	https://youtu.be/U7PYoEeVnqY

The system was studied in real-world conditions with heterogeneous vehicles and embedded devices. By integrating sensing, communication, prediction, and control across different vehicles, the experiments demonstrated how cooperative information exchange can change the outcome of risky interactions. The experiments demonstrated that prediction changes the interaction outcome: when active, the ego vehicle smoothly yielded and enabled a safe merge, whereas without prediction the target vehicle was forced into abrupt braking.

This study was designed to show a cooperative architecture in which the TV, as the vehicle with the most informative view of the critical event, shares contextual information with the EV. For this reason, we did not implement a configuration where the EV handles its own perception and relies on its local sensing. Instead, the hardware experiments are intended to illustrate how cooperation can change the outcome of an interaction in a situation where local perception alone would have limited visibility of the underlying risk. These arguments are expected to become even more relevant in more complex layouts. For example, in a three-lane highway scenario where the TV is in the center lane, the EV is in the left lane, and another vehicle is present in the right lane, an EV relying solely on its own sensors would have limited visibility of the interaction between the TV and the vehicle in the right lane, and would struggle to assess the associated risk. By contrast, cooperative perception allows the TV to share a more complete and accurate description of its surroundings, including vehicles that are partially or fully occluded from the EV's point of view. Even in the two-lane configuration studied in this work, the TV has a better position point to measure the distance and TTC to the PV directly ahead in its lane than an EV that is already ahead in the adjacent lane at the beginning of the scenario as shown in Fig. 3. An EV relying only on its own sensors would first need to infer the geometry and gap between the TV and PV indirectly, which is more complex and less accurate than using the TV's own measurements. In the studied cooperative architecture, the TV senses the surrounding traffic and transmits the resulting contextual information to the EV, which can react based on a clearer and earlier understanding of the risk. This is consistent with previous cooperative perception studies, which have shown that information sharing between vehicles can mitigate occlusions, extend the effective field of view, and improve safety compared to purely vehicle-centric sensing in multi-lane traffic [52,53]. In practice, cooperative perception is expected to complement rather than replace local sensing, providing earlier and richer context in situations where local sensors have limited visibility or are affected by occlusions.

9.1.2. System and module levels design insights

At the system level, several deep insights and bottlenecks emerged:

- **Hardware Constraints:** Perception pipelines such as RAFT-stereo provided detailed kinematics but were limited by low FPS. This highlights the trade-off between model richness and real-time feasibility.
- **Communication Reliability:** Direct V2V links offered the lowest delay but were unreliable over distance. The relay-based design, while adding 3.5 ms latency, proved essential for modularity, scalability, and robust operation in multi-vehicle settings. The experiments also show that the cooperative prediction remains stable even when the effective relay delay is increased by up to 200 ms (about 25 times the nominal 7.25 ms), with no change in the qualitative outcome of the scenario, indicating a comfortable timing margin with respect to moderate variations in communication latency.
- **Prediction Scalability:** Lookup tables transformed inference into constant-time complexity, making real-time deployment feasible even on resource-limited devices. This design insight is crucial for scaling cooperative architectures.
- **Environmental Factors:** Lighting variability, GPS interference, and thermal effects on embedded boards were persistent issues, showing that perception and computation pipelines must be tailored to deployment environments.
- **System Integration:** The cooperative prediction fundamentally changed the traffic interaction outcome. Anticipation at a ∼4 s horizon enabled smoother and safer behavior, providing a level of foresight not achievable with reactive control alone.

9.1.3. Application of the research results

Beyond these observations, the results provide concrete guidance for designing prediction modules in real systems. First, the use of linguistic categorical inputs instead of continuous numerical features shows that the prediction problem can be reformulated in a hardware-friendly way. When the key variables are discretized into a finite set of linguistic categories, the input space becomes finite and the prediction outcomes can be precomputed and stored in a lookup table. This enables an offline prediction model that can be executed at runtime as a constant-time table lookup, avoiding the need for online evaluation of complex numerical models and reducing the computational requirements of the prediction module. In practice, this means that the prediction module can be deployed on low-cost embedded boards rather than high-end GPUs, while keeping the learning phase on external workstations or cloud resources. At the communication level, the experiments suggest some practical choices for cooperative systems. In our setup, short messages containing only a few categorical values were sufficient to keep the vehicles coordinated, and the wireless link was stable enough for the prediction to arrive in time for the ego vehicle to react. The use of a relay node between the vehicles proved to be helpful when direct links were unreliable, showing that roadside or intermediate units can play a useful role in maintaining connectivity in more complex environments. Combined with the observed 3–4 s anticipation horizon, these results indicate that, if communication delays are kept reasonably small compared to this horizon, the ego vehicle has enough time to decelerate smoothly and open a sufficient gap for the target vehicle to maneuver safely. The findings on environmental and hardware robustness are also useful for deploying similar systems in hot and sunny regions. The thermal issues observed on the embedded boards during hot-weather tests indicate that active cooling, careful placement of processing units away from direct sunlight, and thermal monitoring with safe fallback modes should be considered in real vehicles. Likewise, the sensitivity of the stereo camera to strong sunlight and glare suggests that camera may need basic protection from direct sunlight, such as a small hood, an appropriate lens filter, or adjusted exposure settings, and that testing and calibration should be scheduled at times and locations with more stable illumination when possible. Finally, the disturbances observed on the GPS signal near electrical installations highlight the need to either select test areas with limited interference or to complement GNSS with additional localization sources when deploying cooperative perception systems in dense urban environments.

9.1.4. Edge scenarios and prediction stability

The hardware validation in this article is limited to a single cooperative left lane-change scenario in which the TV will change lanes to avoid a braking PV. The main objective of the study is not to perform a broad evaluation of all possible lane-change situations, but to study in detail one representative cooperative lane-change scenario and to extract the main bottlenecks and deployment issues that arise when implementing the pipeline on real hardware. These insights are intended to guide a broader experimental campaign in future work, where more lanes, additional traffic configurations, and a wider set of corner cases will be considered, taking into account the difficulties and lessons learned in this focused experiment. The stability and generalization of the prediction module itself have already been studied more broadly in simulation studies, where the model was exposed to a variety of lane-keeping and lane-change situations. Based on those studies, we expect that in a braking-only case where the TV approaches the PV and decelerates without changing lanes, the TTC between them becomes very small and the model reacts by increasing the probability of a lane change or cut-in, causing the EV to decelerate as a conservative response. Once the TV has slowed down and the TTC increases again, the model reduces the lane-change probability and returns to a lane-keeping prediction, so that the EV can resume its motion. A similar transient pattern is expected when a lane change is initiated and then aborted: the lane-change probability rises while the TV exhibits lateral motion and drops again when the vehicle recenters in its lane. These reactions have been observed in simulation studies of the prediction model, but they have not yet been tested on the real hardware and will be considered in the broader experimental campaign planned as future work.

9.1.5. Dynamic gap

There is also a dynamic gap between the scaled vehicles used in our experiments and full-size passenger vehicles. The small scaled platforms operate at low speeds and have different mass and thrust-to-weight characteristics, which leads to shorter braking distances and faster responses to control commands. For this reason, the absolute values of acceleration reported in our experiments are specific to this testbed and should not be interpreted as safety margins for real vehicles. The use of scaled vehicles as testbeds for autonomous driving research is in line with existing platforms such as MiniCity and F1TENTH [54,55], which are designed to study perception–planning–control pipelines and system integration under realistic but scaled dynamics, while full-scale safety assessment is left to real vehicles. In the same manner, we interpret our results mainly through time-based quantities such as TTC and a 3–4 s anticipation horizon, which provide a meaningful indication of how early the system reacts, while the exact stopping distance and deceleration levels must be re-tuned and validated on full-size vehicles. In our scenario, a 3–4 s anticipation window was sufficient for the scaled ego vehicle to decelerate smoothly and create a safe gap. This interpretation is consistent with our previous work in the CARLA simulator [36], where the same prediction logic was evaluated with full-scale vehicle models and realistic dynamics. In that study, using TTC as a risk indicator and providing a prediction horizon on the order of 3–4 s also allowed the ego vehicle to brake earlier and avoid near-crash situations when compared with purely reactive control. Although those simulation results are not part of the experiments reported in this article and do not replace full-scale on-road tests, they provide additional qualitative support that a few seconds of anticipation can be beneficial under more realistic vehicle dynamics. As an additional step, future work could incorporate dynamics-mimicking controllers to make the scaled vehicles more closely match the longitudinal behavior of a full-size vehicle as presented in [56] which applied Buckingham's pi theorem to obtain a scaled car whose longitudinal and power-train dynamics are similar to those of a full-size vehicle.

9.2. Conclusion

This work has demonstrated a cooperative perception–prediction pipeline deployed on real hardware in mixed traffic, combining a human-driven PV and TV with an automated scaled EV. The experiments show that sharing contextual information from the TV and anticipating the TV's lane change 3–4 s in advance enables the EV to decelerate early and open a safe gap for the TV to maneuver, whereas without prediction the same interaction leads to a sharper and riskier maneuver. At the system level, the study highlights that the main bottlenecks lie at the perception interface, while the use of linguistic categorical inputs and lookup-table inference keeps the prediction module both interpretable and computationally negligible on embedded hardware. Relay-based communication adds only a few milliseconds to the latency budget but improves modularity and robustness. Future work will extend the experimental campaign to additional scenarios and corner cases, and investigate dynamics-mimicking controllers to confirm these findings under more realistic conditions.

CRediT authorship contribution statement

Mohamed Manzour: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Catherine M. Elias:** Writing – original draft, Validation, Software, Resources, Methodology, Investigation, Formal analysis. **Omar M. Shehata:** Writing – original draft, Validation, Software, Resources, Methodology, Investigation, Formal analysis. **Rubén Izquierdo:** Writing – review & editing, Supervision. **Miguel Ángel Sotelo:** Writing – review & editing, Supervision, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Y. Zhang, X. Shi, S. Zhang, A. Abraham, A xgboost-based lane change prediction on time series data using feature engineering for autopilot vehicles, IEEE Trans. Intell. Transp. Syst. 23 (10) (2022) 19187–19200.

[2] R. Syama, C. Mala, An ensemble model for lane change intention inference for autonomous driving, in: 2022 International Conference on Computing, Communication, Security and Intelligent Systems, IC3SIS, IEEE, 2022, pp. 1–6.

[3] Y. Ma, S. Song, L. Zhang, L. Xiong, J. Chen, Lane change analysis and prediction using mean impact value method and logistic regression model, in: 2021 IEEE International Intelligent Transportation Systems Conference, ITSC, IEEE, 2021, pp. 1346–1352.

[4] J. Li, B. Dai, X. Li, X. Xu, D. Liu, A dynamic Bayesian network for vehicle maneuver prediction in highway driving scenarios: Framework and verification, Electronics 8 (1) (2019) 40.

[5] M. Liu, Y. Li, Q. Yang, W. Wu, Peril set-aided lane-change intention inference on highway, IEEE Internet Things J. 11 (9) (2024) 16647–16659.

[6] Y. Xia, Z. Qu, Z. Sun, Z. Li, A human-like model to understand surrounding vehicles' lane changing intentions for autonomous driving, IEEE Trans. Veh. Technol. 70 (5) (2021) 4178–4189.

[7] Z. Li, Y. Wang, Z. Zuo, Z. Liu, Y. Chen, H. Li, Early intention prediction of lane-changing based on dual Gaussian-mixed hidden Markov models, in: 2023 62nd IEEE Conference on Decision and Control, CDC, IEEE, 2023, pp. 4081–4086.

[8] R. Izquierdo, A. Quintanar, I. Parra, D. Fernández-Llorca, M. Sotelo, Experimental validation of lane-change intention prediction methodologies based on CNN and LSTM, in: 2019 IEEE Intelligent Transportation Systems Conference, ITSC, IEEE, 2019, pp. 3657–3662.

[9] K. Liang, J. Wang, A. Bhalerao, Lane change classification and prediction with action recognition networks, in: European Conference on Computer Vision, Springer, 2022, pp. 617–632.

[10] D. Fernández-Llorca, M. Biparva, R. Izquierdo-Gonzalo, J.K. Tsotsos, Two-stream networks for lane-change prediction of surrounding vehicles, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC, IEEE, 2020, pp. 1–6.

[11] M. Biparva, D. Fernández-Llorca, R.I. Gonzalo, J.K. Tsotsos, Video action recognition for lane-change classification and prediction of surrounding vehicles, IEEE Trans. Intell. Veh. 7 (3) (2022) 569–578.

[12] R. Izquierdo, Á. Quintanar, J. Lorenzo, I. García-Daza, I. Parra, D. Fernández-Llorca, M.Á. Sotelo, Vehicle lane change prediction on highways using efficient environment representation and deep learning, IEEE Access 9 (2021) 119454–119465.

[13] S. Su, K. Muelling, J. Dolan, P. Palanisamy, P. Mudalige, Learning vehicle surrounding-aware lane-changing behavior from observed trajectories, in: 2018 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2018, pp. 1412–1417.

[14] L. Amer, C.M. Elias, Enhancing highway safety with LSTM-based vehicle intention prediction (HVIP), in: 2024 International Conference on Computer and Applications, ICCA, IEEE, 2024, pp. 1–6.

[15] Q. Shangguan, T. Fu, J. Wang, S. Fang, L. Fu, A proactive lane-changing risk prediction framework considering driving intention recognition and different lane-changing patterns, Accid. Anal. Prev. 164 (2022) 106500.

[16] O. Scheel, N.S. Nagaraja, L. Schwarz, N. Navab, F. Tombari, Attention-based lane change prediction, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 8655–8661.

[17] O. Scheel, N.S. Nagaraja, L. Schwarz, N. Navab, F. Tombari, Recurrent models for lane change prediction and situation assessment, IEEE Trans. Intell. Transp. Syst. 23 (10) (2022) 17284–17300.

[18] O. Laimona, M.A. Manzour, O.M. Shehata, E.I. Morgan, Implementation and evaluation of an enhanced intention prediction algorithm for lane-changing scenarios on highway roads, in: 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference, NILES, IEEE, 2020, pp. 128–133.

[19] K. Gao, X. Li, B. Chen, L. Hu, J. Liu, R. Du, Y. Li, Dual transformer based prediction for lane change intentions and trajectories in mixed traffic environment, IEEE Trans. Intell. Transp. Syst. 24 (6) (2023) 6203–6216.

[20] X. Guo, H. Jia, Q. Huang, Q. Luo, N. Wang, Z. Mao, A vehicle driving intention prediction method based on gated dual tower transformer model for autonomous driving, Expert Syst. Appl. (2025) 128000.

[21] Y. Lu, Z. Zhang, W. Wang, Y. Zhu, T. Chen, Y.D. Al-Otaibi, A.K. Bashir, X. Hu, Knowledge-driven lane change prediction for secure and reliable internet of vehicles, IEEE Trans. Intell. Transp. Syst. (2025).

[22] M. Peng, X. Guo, X. Chen, K. Chen, M. Zhu, L. Chen, F.Y. Wang, Lc-llm: Explainable lane-change intention and trajectory predictions with large language models, Commun. Transp. Res. 5 (2025) 100170.

[23] Y. Lu, P. Xu, X. Jiang, A.K. Bashir, T.R. Gadekallu, W. Wang, X. Hu, Lane change prediction for autonomous driving with transferred trajectory interaction, IEEE Trans. Intell. Transp. Syst. (2025).

[24] P. Li, G. Shen, Q. Pan, Z. Liu, X. Kong, Social behavior-aware driving intention detection using spatio-temporal attention network, in: China National Conference on Big Data and Social Computing, Springer, 2023, pp. 121–134.

[25] M. Manzour, A. Ballardini, R. Izquierdo, M. Sotelo, Vehicle lane change prediction based on knowledge graph embeddings and bayesian inference, in: 2024 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2024, pp. 1893–1900.

[26] M.M. Hussien, A.N. Melo, A.L. Ballardini, C.S. Maldonado, R. Izquierdo, M.A. Sotelo, Rag-based explainable prediction of road users behaviors for automated driving using knowledge graphs and large language models, Expert Syst. Appl. 265 (2025) 125914.

[27] S. Chen, L. Piao, X. Zang, Q. Luo, J. Li, J. Yang, J. Rong, Analyzing differences of highway lane-changing behavior using vehicle trajectory data, Phys. A 624 (2023) 128980.

[28] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis, An ensemble deep learning approach for driver lane change intention inference, Transp. Res. Part C: Emerg. Technol. 115 (2020) 102615.

[29] A. Jain, H.S. Koppula, B. Raghavan, S. Soh, A. Saxena, Car that knows before you do: Anticipating maneuvers via learning temporal driving models, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3182–3190.

[30] H. Berndt, J. Emmert, K. Dietmayer, Continuous driver intention recognition with hidden markov models, in: 2008 11th International IEEE Conference on Intelligent Transportation Systems, IEEE, 2008, pp. 1189–1194.

[31] K. Li, X. Wang, Y. Xu, J. Wang, Lane changing intention recognition based on speech recognition models, Transp. Res. Part C: Emerg. Technol. 69 (2016) 497–514.

[32] J. Li, T. Jiang, H. Liu, Y. Sun, C. Lv, Q. Li, G. Yin, Y. Liu, Lane changing maneuver prediction by using driver's spatio-temporal gaze attention inputs for naturalistic driving, Adv. Eng. Informatics 61 (2024) 102529.

[33] Z. Rastin, D. Söffker, Multi-level feature extraction and classification for lane changing behavior prediction and POD-based evaluation, Automation 5 (3) (2024) 310–323.

[34] X. Chen, S. Wu, C. Shi, Y. Huang, Y. Yang, R. Ke, J. Zhao, Sensing data supported traffic flow prediction via denoising schemes and ANN: A comparison, IEEE Sensors J. 20 (23) (2020) 14317–14328.

[35] L. Li, W. Zhao, C. Xu, C. Wang, Q. Chen, S. Dai, Lane-change intention inference based on RNN for autonomous driving on highways, IEEE Trans. Veh. Technol. 70 (6) (2021) 5499–5510.
[36] M. Manzour, A. Ballardini, R. Izquierdo, M.A. Sotelo, Explainable lane change prediction for near-crash scenarios using knowledge graph embeddings and retrieval augmented generation, in: 2025 IEEE Intelligent Vehicles Symposium, IV, 2025, pp. 1538–1545, http://dx.doi.org/10.1109/IV64158.2025.11097512.
[37] L. Costabello, S. Pai, C.L. Van, R. McGrath, N. McCarthy, P. Tabacof, AmpliGraph: a Library for Representation Learning on Knowledge Graphs, 2019, http://dx.doi.org/10.5281/zenodo.2595043.
[38] B. Morris, A. Doshi, M. Trivedi, Lane change intent prediction for driver assistance: On-road design and evaluation, in: 2011 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2011, pp. 895–901.
[39] A. Doshi, B. Morris, M. Trivedi, On-road prediction of driver's intent with multimodal sensory cues, IEEE Pervasive Comput. 10 (3) (2011) 22–34.
[40] M. Manzour, C.M. Elias, O.M. Shehata, R. Izquierdo, M. Sotelo, Real-world deployment of a lane change prediction architecture based on knowledge graph embeddings and Bayesian inference, 2025, arXiv preprint arXiv:2506.11925.
[41] R.I. Gonzalo, C.S. Maldonado, J.A. Ruiz, I.P. Alonso, D.F. Llorca, M.Á. Sotelo, Testing predictive automated driving systems: lessons learned and future recommendations, IEEE Intell. Transp. Syst. Mag. 14 (6) (2022) 77–93.
[42] M.A. Manzour, C.M. Elias, E.I. Morgan, O.M. Shehata, Development of a modular ROS-enabled pedestrian intention prediction architecture for AVs maneuvering control, IEEE Trans. Intell. Transp. Syst. (2024).
[43] Ultralytics, YOLOv8: The latest in real-time object detection, 2025, https://yolov8.com/. (Accessed 05 July 2025).
[44] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[45] A. Howard, M. Sandler, G. Chu, L.C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al., Searching for mobilenetv3, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1314–1324.
[46] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, X. Wang, ByteTrack: Multi-object tracking by associating every detection box, 2022, URL https://arxiv.org/abs/2110.06864, arXiv:2110.06864.
[47] L. Lipson, Z. Teed, J. Deng, RAFT-stereo: Multilevel recurrent field transforms for stereo matching, in: International Conference on 3D Vision, 3DV, 2021.
[48] L.C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 801–818.
[49] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.
[50] G. Jocher, U. Team, YOLOv8: Real-time multi-task vision models from ultralytics, 2023, https://github.com/ultralytics/ultralytics. (Accessed 24 May 2025).
[51] L.C. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking atrous convolution for semantic image segmentation, 2017, arXiv preprint arXiv:1706.05587, arXiv:1706.05587.
[52] Y. Wang, W. Sun, C. Liu, Z. Cui, M. Zhu, Z. Pu, et al., Cooperative perception of roadside unit and onboard equipment with edge artificial intelligence for driving assistance, 2021.
[53] Q.I. Ali, H.M. Mohammed, Enhancing road safety and network intelligence through vehicleto-everything (V2X) communication: Architectures, models, and performance analysis, Transp. Dev. Res. 3 (1) (2025) 1–13.
[54] N. Buckman, A. Hansen, S. Karaman, D. Rus, Evaluating autonomous urban perception and planning in a 1/10th scale minicity, Sensors 22 (18) (2022) 6793.
[55] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y.V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, et al., F1/10: an open-source autonomous cyber-physical platform, 2019, arXiv preprint arXiv:1901.08567.
[56] R. Verma, D. Del Vecchio, H.K. Fathy, Development of a scaled vehicle with longitudinal dynamics of an HMMWV for an ITS testbed, IEEE/ASME Trans. Mechatronics 13 (1) (2008) 46–57.

Mohamed Manzour Hussien obtained his bachelor's degree in mechatronics from the German University in Cairo (GUC) in 2019. During his undergraduate studies, he had the opportunity to conduct his bachelor's thesis in the field of machine learning at the IFS (Institut für Schienenfahrzeuge) institute in Germany. After that, he worked as a lecturer assistant in the GUC till 2022. During this period, he completed his master's degree in the field of Intelligent Transportation Systems (ITS) at the Multi-Robot Systems (MRS) research group in the GUC in 2022. He focused on pedestrian behavior prediction. In 2023, he started his Ph.D. journey in the field of ITS at the INVETT (INtelligent VEhicles and Traffic Technologies) research group, University of Alcala, Spain. He is focusing on vehicle behavior prediction. In 2023, he won the IEEE ITSS student competition in the track of Driver Decision Prediction.

Catherine M. Elias is a lecturer in the Computer Science and Engineering Department, Faculty of Media Engineering and Technology (MET), German University in Cairo (GUC). She received her Ph.D. degree in Mechatronics Engineering from the GUC in Dec. 2022 in the field of Cooperative Architecture for Transportation Systems. She received her M.Sc. degree in Mechatronics Engineering from the GUC in Nov. 2017 in the field of cooperative control of multi-robot systems. She is currently the director of the Cognitive Driving Research in Vehicular Systems (C-DRiVeS Lab), which explores autonomous driving stack modules with a special focus on driving behavior in Egyptian traffic dynamics. The lab develops and curates region-specific datasets to capture the complexity of local road environments, aiming to build perception and decision-making models that reflect real-world driving conditions in Egypt and comparable emerging regions. She serves as a Board of Governors (BoG) member in the IEEE ITS Society during the interval 2023-2025, the 2023-2025 Co-chair of the committee on Diversity, Equity, and Inclusion in ITS committee chair.

Omar M. Shehata is an Associate Professor of Mechatronics and Robotics Engineering in the German University in Cairo (GUC), Egypt. He obtained his BSc in Mechatronics Engineering in 2010 from Ain Shams University (ASU). He obtained his 1st MSc from ASU in 2014 in Multi-Vehicle Coordination in intersections, and another MSc in 2015 from GUC in micro-robots control. In 2018 he finished his PhD from the Mechatronics Engineering Dept, ASU. In 2019, he finished his MBA from the school of Business, specializing in Strategic Planning. He is also the director of the Multi-Robot Systems (MRS) Research Group focusing on addressing the challenges related to the coordination of multi-robot and multi-vehicular systems. MRS application areas extend across different fields including robotics manipulators, legged robots, connected vehicles including the aerial and ground. His research interests include robot cooperation, computer vision and optimization techniques. Also, he is the founder and Head of the DRIVe-IT Hub, which is the central hub bridging challenges between Africa & MENA region countries and the international Intelligent Systems (ITS) community.

Rubén Izquierdo received the Bachelor's degree in electronics and industrial automation engineering in 2014, the M.S. in industrial engineering in 2016, and the Ph.D. degree in information and communication technologies in 2020 from the University of Alcalá (UAH). He is currently Assistant Professor at the Department of Computer Engineering of the UAH. His research interest is focused on the prediction of vehicle behaviors and control algorithms for highly automated and cooperative vehicles. His work has developed a predictive ACC and AES system for cut-in collision avoidance successfully tested in Euro NCAP tests. He was awarded with the Best Ph.D. thesis on Intelligent Transportation Systems by the Spanish Chapter of the ITSS in 2022, the outstanding award for his Ph.D. thesis by the UAH in 2021. He also received the XIII Prize from the Social Council of the UAH to the University–Society Knowledge Transfer in 2018 and the Prize to the Best Team with Full Automation in GCDC 2016.

Miguel Ángel Sotelo received the degree in Electrical Engineering in 1996 from the Technical University of Madrid, the Ph.D. degree in Electrical Engineering in 2001 from the University of Alcalá (Alcalá de Henares, Madrid), Spain, and the Master in Business Administration (MBA) from the European Business School in 2008. He is currently a Full Professor at the Department of Computer Engineering of the University of Alcalá (UAH). His research interests include Self-driving cars, Prediction Systems, and Traffic Technologies. He is author of more than 300 publications in journals, conferences, and book chapters. He has been recipient of the Best Research Award in the domain of Automotive and Vehicle Applications in Spain in 2002 and 2009, and the 3M Foundation Awards in the category of eSafety in 2004 and 2009. Miguel Ángel Sotelo has served as Project Evaluator,

Rapporteur, and Reviewer for the European Commission in the field of ICT for Intelligent Vehicles and Cooperative Systems in FP6 and FP7. He was Editor-in-Chief of the IEEE Intelligent Transportation Systems Magazine (2014–2016), Associate Editor of IEEE Transactions on Intelligent Transportation Systems (2008–2014), member of the Steering Committee of the IEEE Transactions on Intelligent Vehicles (since 2015), and a member of the Editorial Board of The Open Transportation Journal (2006–2015). He has served as General Chair of the 2012 IEEE Intelligent Vehicles Symposium (IV'2012) that was held in Alcalá de Henares (Spain) in June 2012. He was recipient of the IEEE ITS Outstanding Research Award in 2022, the IEEE ITS Outstanding Application Award in 2013, and the Prize to the Best Team with Full Automation in GCDC 2016. He is a Former President of the IEEE Intelligent Transportation Systems Society.