Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.6503

A Driver Assistance System Based on YOLO Object Detection: Development and Experimental Validation in the CARLA Simulator

Daniel Terra Gomes [**Universidade Estadual do Norte Fluminense Darcy Ribeiro** | **danielterra@pq.uenf.br**]
Annabell Del Real Tamariz [**Universidade Estadual do Norte Fluminense Darcy Ribeiro** | **annabell@uenf.br**]

Universidade Estadual do Norte Fluminense Darcy Ribeiro, Centro de Ciência e Tecnologia, Laboratório de Ciências Matemáticas, Av. Alberto Lamego, 2000 - Parque California, Campos dos Goytacazes - RJ, 28013-602, Brazil.

Received: 05 September 2025 • **Accepted:** 27 April 2026 • **Published:** 06 May 2026

Abstract. This work presents the development and validation of an integrated **Advanced Driver-Assistance System (ADAS)** combining YOLOv8-based computer vision with vehicle control in the CARLA simulator. The primary objective was to implement and validate the system's capability to detect stop signs and vehicles in real time, providing visual feedback to the driver and enabling the vehicle to respond to stop signs through appropriate deceleration and stopping behavior. The system employs a modular three-layer architecture: perception (YOLOv8), planning (finite state machine), and control (Proportional-Integral-Derivative (PID) longitudinal, Pure Pursuit lateral). Evaluation across six simulations under three weather conditions (clear noon, heavy rain sunset, heavy rain noon) demonstrated real-time processing at 17.13 FPS average, 59.4 ms detection time per frame, and 100% detection accuracy for stop signs and vehicles on the test route. Stop sign detection confidence remained above 0.73 across all conditions (coefficient of variation: 0.58%), and ANOVA revealed a significant effect of weather on detection time ($p = 0.025$) but no impact on detection confidence ($p = 0.651$), confirming perceptual reliability under adverse conditions. All routes were completed without collisions. Despite limitations inherent to simulated validation, the results empirically confirm that YOLO-based ADAS can provide reliable real-time visual feedback and proper behavioral responses for stop signs and vehicles under three distinct weather conditions, establishing methodological foundations for modular, distributed-processing architectures in autonomous vehicle applications.

Keywords: Self-driving car, Object Detection, Vehicle Control, Simulation, CARLA, YOLO

1 Introduction

Autonomous Vehicles (AVs) represent a transformative advancement in urban mobility and transportation safety [European Commission, Directorate-General for Mobility and Transport, 2021; Othman, 2021], with growing investment from companies like ZOOX, Waymo, and Lume Robotics driven by the expanding electric vehicle market [Sebo, 2024; Bratzel and Center of Automotive Management, 2022; Parekh *et al*., 2022]. A primary motivation for AV development is accident reduction: human error causes over 90% of traffic accidents, and large-scale AV adoption could prevent up to 585,000 fatalities over a decade [National Highway Traffic Safety Administration, 2018; Lanctot, 2017; Okpono *et al*., 2024]. Real-time object detection is fundamental to this goal, providing the perceptual backbone for safe autonomous navigation [Janai *et al*., 2020]. However, achieving full autonomy (SAE[1] Level 5) requires overcoming substantial challenges in computer vision [SAE International, 2021], and the high costs of real-world testing have led researchers to rely on simulators like CARLA[2] for algorithm development and validation [Dosovitskiy *et al*., 2017; Ahire *et al*., 2024]. While Advanced Driver-Assistance Systems (ADAS) represent significant progress, their effectiveness depends on the precision of their detection algorithms, which can degrade under varied environmental conditions [Ahire *et al*., 2024; Khan *et al*., 2022; Wachenfeld and Winner, 2016]. This underscores the need for solutions that integrate high-precision object detection with real-time processing in simulated environments. This context leads to our central research hypothesis: **A driver assistance system in autonomous cars can offer real-time visual feedback of stop signs and vehicles, making driving more reliable.** To validate this, we developed an autonomous system based on the YOLO computer vision algorithm, capable of detecting stop signs and vehicles and providing visual alerts to the driver, with validation performed in the CARLA simulation environment. The main objective of this work is to develop and validate an autonomous system for testing a computer vision solution that detects and recognizes stop signs and vehicles in real-time for AVs, using a YOLO-series algorithm and the CARLA simulation environment. The specific objectives are:

1. To implement a complete three-layer autonomous driving system (perception, planning, and control) capable of navigating a predefined urban route.
2. To develop a real-time object detection module using YOLO to identify stop signs with a processing rate greater than 10 FPS and provide visual feedback.
3. To implement a behavioral response system that enables the vehicle to react to detected stop signs by decelerating

[1] Society of Automotive Engineers (SAE) International: `https://www.sae.org/`.
[2] CARLA Simulator official website: `https://carla.org/`.

or stopping.
4. To establish a real-time visualization interface for monitoring detections, system performance metrics (FPS, latency), and vehicle status.
5. To experimentally validate the system, collecting quantitative metrics on detection rates, processing time, and the ability to complete the route without collisions.

The main contributions of this work are: (i) the design and implementation of a modular three-layer autonomous driving architecture (perception, planning, and control) integrating YOLOv8 for real-time stop sign and vehicle detection and driver feedback; (ii) a novel distributed client-server processing solution to overcome Python version incompatibilities in the CARLA ecosystem; and (iii) a comprehensive experimental validation protocol demonstrating system robustness across multiple weather conditions with quantitative statistical analysis (ANOVA).

The remainder of this paper is organized as follows. Section 2 reviews the background and related work on SAE automation levels, the autonomous driving task hierarchy, simulation environments, and YOLO-based detection systems. Section 3 presents the system architecture and methodology. Section 4 details the implementation, including the technical environment, module design, and experimental setup. Section 5 presents the experimental results and discussion, including comparison with related approaches. Finally, Section 6 concludes the paper and outlines directions for future work.

2 Background and Related Work

The development of AVs is underpinned by key concepts and technologies, including standardized automation levels, a hierarchical task structure, and the use of simulation environments. This section reviews these foundational elements and contextualizes our work within the existing scientific literature.

2.1 SAE Levels of Driving Automation

The Society of Automotive Engineers (SAE) J3016 standard provides the most widely adopted taxonomy for classifying vehicle automation into six levels (0–5), based on which entity (driver or system) performs the Dynamic Driving Task (DDT), the monitoring of the driving environment, and the fallback response to critical situations [SAE International, 2021]. The six levels are formally defined as follows:

0. **No Automation:** The driver performs all aspects of the DDT. Momentary warning alerts do not constitute automation.
1. **Driver Assistance:** The system performs sustained steering *or* acceleration/deceleration, but the driver remains responsible for monitoring and all other DDT aspects (e.g., adaptive cruise control).
2. **Partial Automation:** The system performs sustained steering *and* acceleration/deceleration. The driver must constantly monitor the environment and be ready to retake control immediately.
3. **Conditional Automation:** The system performs all DDT aspects with the expectation that the driver will respond to intervention requests when the system's capabilities are exceeded.
4. **High Automation:** The system performs all DDT aspects and the fallback response within its Operational Design Domain (ODD). No driver intervention is expected within the ODD.
5. **Full Automation:** The system performs all DDT aspects under all conditions manageable by a human driver. There is no ODD limitation.

Currently, most commercial vehicles operate at Levels 1–2, classified as ADAS [Lage, 2019]. A significant milestone was reached in 2023 when Mercedes-Benz obtained certification for a Level 3 system (DRIVE PILOT) in the United States [Mitropoulos, 2023], while companies like Waymo operate limited Level 4 commercial services in specific geographic areas [Houser, 2023].

System targeting. The system developed in this work is designed as an **ADAS aligned with SAE Levels 1–2**, where the detection and classification of traffic signs serves as real-time visual feedback to support the driver, who retains full responsibility for the driving task. At the same time, the complete three-layer autonomous pipeline (perception, planning, and control) implemented in this work establishes a modular foundation that can be expanded toward higher automation levels.

2.2 The Hierarchical Task of Driving

The complex task of autonomous driving is typically decomposed into a three-layer hierarchical architecture: Perception, Planning, and Control [Khan *et al*., 2022].

- **Perception:** Interprets the environment using data from sensors like cameras, LiDAR, and RADAR to detect and classify static elements (e.g., traffic signs, lane markings) and dynamic elements (e.g., vehicles, pedestrians).
- **Planning:** Determines the vehicle's actions to navigate towards its goal. It is often subdivided into a mission planner (high-level route), a behavioral planner (tactical maneuvers like lane changes), and a local planner (generating precise, collision-free trajectories).
- **Control:** Executes the planned trajectories by sending commands to the vehicle's actuators (steering, throttle, brake). This involves longitudinal control (speed) and lateral control (steering).

Our proposed system implements this full hierarchy, using a camera and the YOLO algorithm for perception, a three-level planner for decision-making, and PID/Pure Pursuit controllers for vehicle control.

2.3 Simulation in AV Development

Simulation is a cornerstone of AV research, providing a safe, scalable, and cost-effective platform for development and validation [Dosovitskiy *et al*., 2017]. Simulators like CARLA (Car Learning to Act) offer high-fidelity urban environments,

configurable sensors, and robust APIs for programmatic control, making them ideal for testing complex perception and control algorithms under a wide range of reproducible conditions, including adverse weather [Dosovitskiy *et al*., 2017]. Our choice of CARLA as the validation environment aligns with established best practices in the field [Gao *et al*., 2021; Surendra *et al*., 2023; Sánchez Juanola, 2019; Kim *et al*., 2023; Andrade, 2022].

2.4 Related Studies on YOLO in AV Simulation

The integration of YOLO with simulators for AV applications is an active area of research. Andrade [2022] developed a system using YOLOv4 in CARLA for object detection and distance estimation, demonstrating the viability of this integration. Sánchez Juanola [2019] created a system for detecting speed limit signs and providing driver feedback, similar in spirit to our work but without the integrated autonomous control component. Gao *et al*. [2021] performed a comparative analysis of YOLOv4, CenterNet, and Faster-RCNN within the CARLA simulator, identifying YOLOv4 as the most efficient detector without specific optimizations, which reinforces the suitability of the YOLO family for simulator-based AV research. Surendra *et al*. [2023] proposed an integrated system for lane detection (using SegNet) and traffic sign detection (using YOLO) in CARLA, achieving a mAP (mean Average Precision) of 93.67% for sign detection across 612 images under varied weather conditions. This work demonstrates the use of combined perception pipelines in similar simulated environments. Ahammed *et al*. [2024] developed a YOLO-based obstacle detection system for construction zones, achieving 94% accuracy with 1.6ms inference time, demonstrating the viability of real-time YOLO-based assistance in specific operational scenarios.

Other studies have focused on improving YOLO's robustness for traffic sign detection. Kim *et al*. [2023] analyzed the degradation of YOLO performance in adverse weather and proposed using simulators like CARLA to generate synthetic training data for more robust models. Architectural improvements to YOLO, such as those proposed by Wu and Cao [2022] and Li *et al*. [2023], have focused on enhancing the detection of small objects like traffic signs by modifying network layers and using optimized distance metrics on the TT100K benchmark dataset.

2.4.1 Justification for YOLOv8 Selection

While newer YOLO versions exist (e.g., YOLOv10, YOLOv11), our selection of YOLOv8 is deliberate and based on a systematic comparative analysis. Table 1 compares YOLOv8s against other contemporary architectures on standardized benchmarks.

YOLOv8s offers the best speed–accuracy tradeoff for our real-time requirements: its 45 FPS far exceeds the 10 FPS target, and its 11.2M parameters make it significantly more lightweight than alternatives like Faster R-CNN (42.0M) or DETR (41.3M). Although DETR achieves a higher mAP, its lower throughput and larger model size are less suitable for our distributed real-time architecture. Furthermore,

Table 1. Comparative analysis of object detection architectures (metrics from original publications, FPS normalized for NVIDIA V100 GPU).

Architecture	mAP@0.5	FPS	Params	Characteristics
YOLOv8s	58.5%	45	11.2M	Single-stage, anchor-free [Wang *et al*., 2024]
Faster R-CNN (R-50-FPN)	61.0%	26	42.0M	Two-stage, region proposal [Liu *et al*., 2020]
SSD (VGG16)	53.3%	22	24.9M	Single-stage, multi-scale [Liu *et al*., 2020]
RetinaNet (R-50-FPN)	57.5%	12	36.4M	Single-stage, focal loss [Lin *et al*., 2017]
DETR (R-50)	62.4%	28	41.3M	Transformer-based [Carion *et al*., 2020]

YOLOv8 presents superior maturity and stability compared to newer versions (YOLOv10/v11), with extensive community validation, comprehensive documentation, and a consolidated ecosystem of pre-trained models through the Ultralytics framework [Yaseen, 2024]. The marginal accuracy gains of newer versions do not justify the associated implementation risks for our system-integration-focused study. The COCO-pretrained YOLOv8s model natively includes the target classes required for our experiments ("car" and "stop sign"), enabling direct deployment without additional fine-tuning [Wang and Liao, 2024].

Table 2. Methodological comparison of related works.

Study	Main Goal	Algorithm(s)	Env.
Gao *et al*. [2021]	Comparative detector analysis	YOLOv4, CenterNet, Faster-RCNN	CARLA Simulator
Surendra *et al*. [2023]	Lane & sign detection pipeline	SegNet + YOLO	CARLA Simulator
Ahammed *et al*. [2024]	Obstacle detection at construction zones	YOLO	Real-world data
Wu and Cao [2022]	Small object detection speed	YOLOv4	TT100K Dataset
Li *et al*. [2023]	Small object accuracy	YOLOv7	TT100K Dataset
Sánchez Juanola [2019]	Speed sign detection & driver feedback	YOLOv3	CARLA Simulator
This Work	**End-to-end ADAS validation**	**YOLOv8, PID, Pure Pursuit**	**CARLA Simulator**

As summarized in Table 2, while these works address specific components of the autonomous driving task (e.g., perception accuracy, driver feedback), a significant gap exists in the literature regarding the end-to-end integration of a modern object detection system (like YOLOv8) with a complete, three-layer vehicle control architecture, validated systematically within a simulator. Our work aims to fill this gap by proposing a holistic solution that combines real-time perception, behavioral planning based on detections, and autonomous vehicle control, thereby contributing a comprehensive, validated system to the field.

3 System Architecture and Methodology

Building on the identified gaps in the literature—particularly the lack of end-to-end system validation combining YOLO-based perception with planning and control in a simulated environment—this section presents our proposed architecture. Our system is designed with a hierarchical three-layer architecture—Perception, Planning, and Control—that mirrors the functional decomposition of the autonomous driving task. Figure 1 illustrates the overall architecture and data flow.

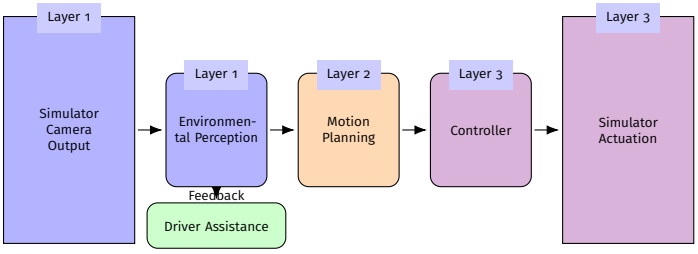

Figure 1. Proposed three-layer software architecture for the autonomous driving system.

3.1 Layer 1: Perception

The perception layer is responsible for interpreting the vehicle's environment. It uses a virtual RGB camera in CARLA as its primary sensor. The raw image data is processed by the YOLOv8 object detection algorithm to identify and classify relevant objects, specifically stop signs and vehicles. In addition to providing data for the planning layer, this layer generates real-time visual feedback for the driver, including bounding boxes around detected objects, classification labels, and confidence scores, directly addressing our primary hypothesis.

3.2 Layer 2: Planning

The planning layer receives structured data from the perception layer and is responsible for generating safe and feasible trajectories. It follows a hierarchical structure:

- **Mission Planner:** Defines the high-level route using a predefined sequence of waypoints, each with a target velocity.
- **Behavioral Planner:** Uses a Finite State Machine (FSM) to make tactical decisions based on environmental context. The FSM transitions between states like `FOLLOW_LANE`, `DECELERATE_TO_STOP`, and `STAY_STOPPED` in response to stop signs.
- **Local Planner:** Generates smooth, kinematically feasible, and collision-free paths to execute the behavioral planner's decisions. It uses a Conformal Lattice Planner to create multiple candidate paths and selects the optimal one based on safety, comfort, and efficiency criteria.

3.3 Layer 3: Control

The control layer executes the trajectory generated by the local planner by sending precise commands to the vehicle's actuators in the CARLA simulator. It is composed of two independent controllers based on the kinematic bicycle model:

- **Longitudinal Control:** A Proportional-Integral-Derivative (PID) controller manages the vehicle's speed. It calculates the error between the current speed and the target speed from the trajectory's velocity profile and applies throttle or brake commands to minimize this error. The PID control law is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad (1)$$

  where $u(t)$ is the control signal, $e(t)$ is the velocity error, and K_p, K_i, K_d are the proportional, integral, and derivative gains, respectively.
- **Lateral Control:** A Pure Pursuit controller manages steering. It identifies a target point on the reference path ahead of the vehicle (at a "lookahead distance") and calculates the steering angle required to navigate the vehicle along a circular arc to intercept that point. The steering angle δ is calculated as:

$$\delta(t) = \arctan\left(\frac{2L\sin\alpha(t)}{l_d}\right) \quad (2)$$

  where L is the vehicle's wheelbase, $\alpha(t)$ is the angle to the lookahead point, and l_d is the lookahead distance, which is adapted based on the vehicle's speed.

4 Implementation Details

This section describes the concrete realization of the architecture presented in Section 3, detailing the technical environment, module implementation, and experimental setup. The system was implemented using Python, leveraging the CARLA simulator (version 0.8.4) for the environment and vehicle dynamics.

4.1 Technical Environment

A key technical challenge was the incompatibility between the CARLA simulator client, which requires Python 3.6, and modern deep learning libraries like PyTorch (for YOLOv8), which require Python 3.8+. To overcome this, we designed a distributed client-server architecture (Figure 2).

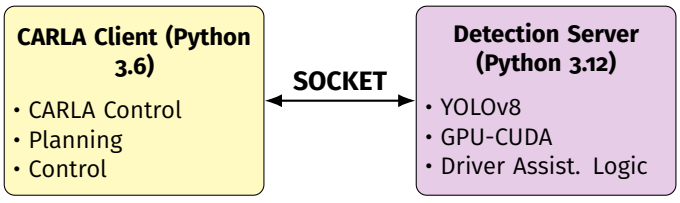

Figure 2. Distributed client-server architecture to overcome Python version incompatibility.

- **Client (Python 3.6):** Runs within the CARLA environment. It is responsible for vehicle control, planning, and capturing camera frames. It serializes and sends frames to the server via a TCP/IP socket.

- **Server (Python 3.12):** Runs in a separate Conda environment. It receives frames, performs object detection using YOLOv8 with GPU acceleration (NVIDIA RTX 2060/4060), and sends the detection results (bounding boxes, classes, confidence scores) back to the client.

Communication was optimized using the MessagePack binary serialization protocol to minimize latency. To ensure real-time performance and prevent the main control loop from being blocked by the detection process, an asynchronous, multi-threaded approach was implemented. A producer thread captures frames and places them in a queue, while a consumer thread sends them to the detection server, ensuring the control loop remains responsive.

4.2 Module Implementation

- **Perception Module:** The YOLOv8s (small) model, pre-trained on the COCO dataset (which includes the `car` and `stop sign` classes), was deployed on a GPU for real-time inference. Visual feedback—bounding boxes, labels, and persistent warnings—was rendered onto the camera feed using OpenCV on the client side.
- **Planning Module:** The behavioral planner was implemented as a simple FSM in the `behavioural_planner.py` script. It transitions between states based on vehicle speed and proximity to virtual fences representing stop signs, whose locations were pre-loaded from a configuration file. The local planner, implemented in `local_planner.py`, generates seven candidate paths using parametric spirals and selects the best collision-free path. Collision checking is performed against static obstacles (parked cars) defined in a configuration file.
- **Control Module:** Both the PID longitudinal controller and the Pure Pursuit lateral controller were implemented within the `controller2d.py` script. The PID gains ($K_p = 0.5, K_i = 0.3, K_d = 0.13$) and the Pure Pursuit lookahead distance (2.0 meters, adapted with speed) were tuned empirically within the CARLA simulation to achieve stable and responsive vehicle behavior.

4.3 Experimental Setup and Data Generation

Validation was performed in the CARLA simulator (version 0.8.4) on its publicly available `Town01` map. Instead of using a pre-existing benchmark dataset, this study's "dataset" consists of quantitative metrics generated from a custom, predefined experimental route. This route was designed to test the system's core functionalities, including straight sections, a 90-degree turn, and a mandatory stop sign interaction.

Figure 3 illustrates the diversity of simulation environments available in CARLA, including varied weather conditions and urban maps. Our experiments were conducted on the `Town01` map under three of these weather configurations.

Six independent simulation runs were conducted under three different weather conditions: *Clear Noon*, *Hard Rain Sunset*, and *Hard Rain Noon*, with two runs per condition to assess result reproducibility. These three conditions were deliberately selected to span a practical range of visibility and environmental challenge: *Clear Noon* provides a baseline of optimal daylight and visibility; *Hard Rain Sunset* introduces reduced lighting combined with heavy precipitation; and *Hard Rain Noon* represents severe precipitation under full daylight—together covering the most operationally relevant adverse scenarios for real-world urban driving. A comprehensive metrics collection system (`performance_metrics.py`) was developed to log data on detection time, FPS, detection confidence, and vehicle dynamics. The specific route configuration, simulation scripts, and all raw data generated and analyzed for this study are openly available, as detailed in the Data Availability Statement. This allows for full reproducibility of our experimental validation.

Figure 3. CARLA simulator environments: four different urban maps under varied weather and lighting conditions, demonstrating the simulator's capability for diverse scenario generation [Dosovitskiy *et al.*, 2017]. Our experiments used the `Town01` map under Clear Noon, Hard Rain Sunset, and Hard Rain Noon conditions.

5 Results and Discussion

This section presents the quantitative and qualitative results obtained from the experimental setup described in Section 4.3, organized by system layer. The system successfully completed the entire predefined trajectory in all six simulation runs across all weather conditions, without any collisions.

5.1 Perception System Performance

The distributed architecture proved highly effective. The system achieved an average processing rate of **17.13 FPS**, significantly exceeding the target of 10 FPS. The average detection time per frame was **59.4 ms**. A key finding was the system's robustness to adverse weather. While the processing time was statistically significantly affected by weather conditions (ANOVA, p=0.025), with performance dropping to 14.41 FPS in heavy noon rain (likely due to the high computational cost of rendering heavy particle effects), the system remained well within real-time operational limits.

Figure 4 shows example detection results from the system, illustrating bounding boxes and confidence scores rendered on the camera feed during simulation runs.

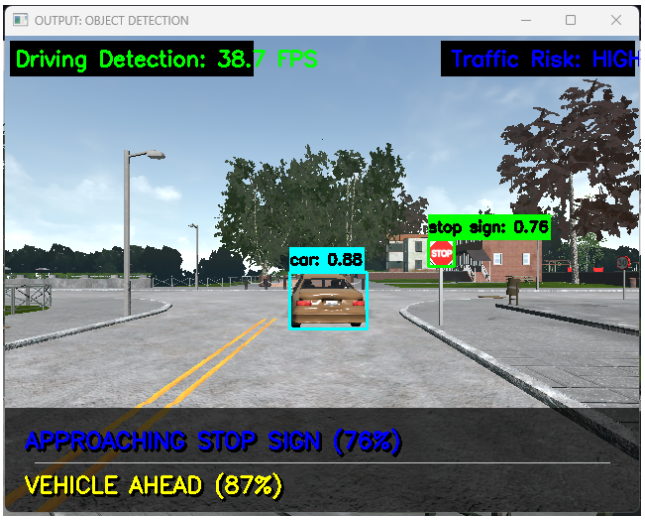

Figure 4. Example YOLOv8 detection results showing real-time visual feedback with bounding boxes, class labels, and confidence scores for stop signs and vehicles under different weather conditions.

Crucially for our hypothesis, the confidence of the detections remained remarkably stable. As shown in Table 3, the average confidence for stop sign detection was **0.738**, with a coefficient of variation of only **0.58%** across all weather conditions. An ANOVA test confirmed that there was no statistically significant difference in stop sign detection confidence between the weather conditions ($p=0.651$), demonstrating the system's reliability. On the predefined experimental route, the system achieved a **100% detection rate** for the target objects (all stop signs and vehicles) present in the trajectory.

Table 3. Key performance metrics across different weather conditions.

Metric	Clear Noon	Rain Sunset	Rain Noon	Average
Avg. FPS	17.74	19.23	14.41	**17.13**
Avg. Detection Time (s)	0.0566	0.0521	0.0694	**0.0594**
Avg. Stop Sign Conf.	0.734	0.743	0.737	**0.738**

5.2 Planning and Control Performance

The planning and control modules performed flawlessly. The behavioral planner's FSM correctly transitioned states upon approaching the stop sign in every run. Figure 5 shows a representative velocity profile, clearly illustrating the deceleration for the stop sign, the brief period of being stationary, and the subsequent re-acceleration.

The lateral control was also successful. The vehicle maintained its lane on straight sections and executed the 90-degree turn smoothly. The local planner successfully generated collision-free paths around the parked vehicle obstacle in all simulations.

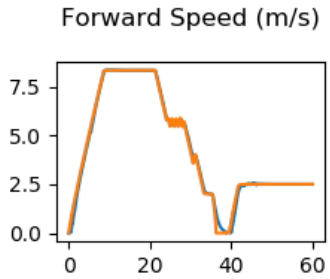

Figure 5. Vehicle's forward speed profile during a simulation run, showing the stop maneuver. The vertical axis (Y) represents the forward speed in m/s, while the horizontal axis (X) represents simulation time steps. The profile shows the vehicle decelerating to zero upon stop sign detection, remaining stationary during the required stop period, and subsequently re-accelerating to resume the route.

5.3 Discussion and Comparison with Related Work

The results provide strong empirical validation for our hypothesis. The system consistently demonstrated its ability to:

1. **Detect stop signs and vehicles in real-time:** Achieved with an average of 17.13 FPS.
2. **Provide reliable feedback:** Stop sign detection confidence remained high (0.738) and stable across adverse conditions, and visual feedback was generated in 100% of cases.
3. **Enable reliable driving:** The vehicle successfully used this information to perform a mandatory stop and completed all trajectories without incident.

It is important to contextualize the 100% detection rate. This is not a generalized accuracy metric (e.g., mAP) on a large-scale benchmark dataset, but rather a system validation metric specific to our controlled, predefined experimental route. This result confirms the system's ability to complete its intended task in a known environment, which was the objective of the study. This focus on system-level validation differentiates our work from studies that focus purely on perception-level optimization.

For instance, while our work focuses on the end-to-end integration of a complete autonomous pipeline, other studies have focused on optimizing the accuracy of the perception module itself. Wu and Cao [2022] and Li *et al*. [2023] propose architectural improvements to YOLOv4 and YOLOv7, respectively, to enhance the detection of small traffic signs, achieving high mAP scores on the challenging TT100K benchmark dataset. Our contribution is complementary: we demonstrate that a modern, off-the-shelf detector like YOLOv8 is sufficiently robust for real-time integration into a full Perception-Planning-Control loop, maintaining stable detection confidence even under adverse simulated weather.

One limitation observed was the occurrence of some false positive detections with high confidence (e.g., classifying parts of a car as a "motorcycle"). This highlights a known challenge in deep learning models and suggests that for safety-critical applications, further model fine-tuning or supplementary validation logic would be necessary. However, since the primary focus was on stop sign detection, which was highly reliable, this did not compromise the validation of our core hypothesis.

For complete transparency and reproducibility, all raw experimental data, source code, and configuration files underlying these results are openly available in the project repository, as detailed in the Availability of Data and Materials section.

6 Conclusion and Future Work

This work successfully developed and validated an integrated driver assistance system based on YOLOv8 object detection within the CARLA simulator. We empirically confirmed our hypothesis that such a system can provide reliable, real-time visual feedback and proper behavioral responses for stop signs and vehicles, validated under three distinct weather conditions, contributing to safer and more dependable driving. The project yielded several key contributions, including a modular and functional three-layer autonomous driving architecture, a novel distributed processing solution to overcome software compatibility challenges, and a structured multi-condition validation protocol with quantitative statistical analysis. The results demonstrated that the system met or exceeded all predefined performance criteria. It operated in real-time (avg. 17.13 FPS), achieved 100% accuracy in detecting both stop signs and vehicles on its predefined path, and demonstrated robust detection confidence across the three evaluated weather conditions (Clear Noon, Hard Rain Sunset, and Hard Rain Noon). The integrated planning and control systems enabled the vehicle to autonomously navigate its route and correctly respond to traffic regulations, completing all test runs without collision.

Despite these successes, the study has limitations, primarily its reliance on a simulated environment and its validation on a single, predefined route with a limited set of objects (stop signs and cars). Future work should focus on bridging the gap between simulation and the real world. A key direction is to enhance the communication architecture; migrating from TCP/IP sockets to an inter-process shared memory system could drastically reduce latency and increase throughput, enabling the integration of multiple sensors. Further research should also expand the system's capabilities by training and validating the detection of a wider variety of traffic signs, pedestrians, and cyclists. Finally, integrating data from complementary sensors, such as LiDAR and RADAR, would create a more robust perception system through sensor fusion, further improving reliability in all environmental conditions.

Declarations

Authors' Contributions

D.T.G. designed the study, performed the experiments, analyzed the data, and was the main writer of the manuscript. A.D.R.T. supervised the research, provided critical feedback, and reviewed the manuscript. Both authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

I express my deep gratitude to all who contributed significantly to this research. To my advisor, Prof. Dr. Annabell Del Real Tamariz, for her rigorous scientific guidance and continuous dedication. To the Universidade Estadual do Norte Fluminense Darcy Ribeiro (UENF) and the Laboratório de Ciências Matemáticas, especially the P5 Laboratory, for providing the academic support and computational resources essential for the experiments. To my family, Carlos and Delma, and my fiancée, Maria, for their unconditional support and encouragement. To my colleagues for enriching scientific discussions and to the researchers whose work provided the theoretical foundation for this investigation.

Funding

This research was supported by a scientific initiation scholarship from the National Council for Scientific and Technological Development (CNPq) through the Institutional Program for Scientific Initiation Scholarships (PIBIC/CNPq).

Availability of data and materials

The complete source code, experimental data, configuration files, and automation scripts generated and analyzed during the current study are openly available in the following GitHub repository: `https://github.com/ARRETdaniel/CARLA_simulator_YOLO-openCV_realTime_objectDetection_for_autonomousVehicles/tree/main/CarlaSimulator/PythonClient/FinalProject`.

References

Ahammed, A. S., Hossain, M. S. A., and Obermaisser, R. (2024). A computer vision approach for autonomous cars to drive safe at construction zone.

Ahire, P., Naidu, S., Varpe, S., Nadarge, S., and Patil, A. (2024). Simulating vehicle driving using carla. *Journal of Electrical Systems*, 20(10s):44–52.

Andrade, G. G. d. (2022). Uma Proposta para Detecção de Objetos e Estimação de Distância em um Simulador de Veículos Autônomos. Trabalho de Conclusão de Curso (Graduação) – Universidade de Brasília.

Bratzel, S. and Center of Automotive Management (2022). Die Zukunft der Mobilität – Die Zukunftstrends in den Bereichen Elektromobilität, Connected Car und Mobilitätsdienstleistungen.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the Conference on Robot Learning*, pages 1–16. PMLR. DOI: 10.48550/arXiv.1711.03938.

European Commission, Directorate-General for Mobility and Transport (2021). Safer roads for all: EU Road Safety Policy Framework 2021-2030.

Gao, W., Tang, J., and Wang, T. (2021). An object detection research method based on CARLA simulation. *Jour-*

nal of Physics: Conference Series, 1948(1):012163. DOI: 10.1088/1742-6596/1948/1/012163.

Houser, K. (2023). Mercedes-Benz wins race to bring level 3 autonomous cars to US. Freethink Media.

Janai, J., Güney, F., Behl, A., and Geiger, A. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends in Computer Graphics and Vision*, 12(1–3):1–308. DOI: 10.1561/0600000079.

Khan, M. A., Sayed, H. E., Malik, S., Zia, T., Khan, J., Alkaabi, N., and Ignatious, H. (2022). Level-5 autonomous driving—are we there yet? a review of research literature. *ACM Computing Surveys*. DOI: 10.1145/3485767.

Kim, T., Jeon, H., and Lim, Y. (2023). Challenges of yolo series for object detection in extremely heavy rain: Calra simulator based synthetic evaluation data set. *arXiv preprint arXiv:2312.07976*. DOI: 10.48550/arXiv.2312.07976.

Lage, C. A. (2019). Quatro cenários para os veículos autônomos no mundo ocidental. Repositório Institucional da UnB.

Lanctot, R. (2017). Accelerating the Future: The Economic Impact of the Emerging Passenger Economy.

Li, S., Wang, S., and Wang, P. (2023). A small object detection algorithm for traffic signs based on improved yolov7. *Sensors*, 23(16). DOI: 10.3390/s23167145.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988. DOI: 10.1109/ICCV.2017.324.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2):261–318. DOI: 10.1007/s11263-019-01247-4.

Mitropoulos, A. (2023). Mercedes-Benz erhält als weltweit erstes Automobilunternehmen Zertifizierung für SAE Level 3-System für US-Markt.

National Highway Traffic Safety Administration (2018). Critical reasons for crashes investigated in the national motor vehicle crash causation survey: Dot hs 812 506 a brief statistical summary.

Okpono, J., Asedegbega, J., Ogieva, M., and Sanyaolu, T. O. (2024). Advanced driver assistance systems road accident data insights: Uncovering trends and risk factors. *The International Journal of Engineering Research*, 11(9):a141–a152. DOI: 10.5281/zenodo.13817419.

Othman, K. (2021). Multidimension analysis of autonomous vehicles: The future of mobility. *Civil Engineering Journal*, 7(7):71–93. DOI: 10.28991/CEJ-SP2021-07-06.

Parekh, D., Poddar, N., and Chahal, M. (2022). A review on autonomous vehicles: Progress, methods and challenges. *Electronics*, 11(14):2162. DOI: 10.3390/electronics11142162.

SAE International (2021). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE J3016_202104*. DOI: 10.4271/J3016_202104.

Sebo, D. (2024). Impact of electric vehicle market growth on automotive industry transformation: trends, potentials, and challenges analysis. In *Economic and Social Development (Book of Proceedings), 106th International Scientific Conference on Economic and Social Development*.

Surendra, H. *et al*. (2023). Lane detection and traffic sign detection using deep learning and computer vision for autonomous driving research using CARLA simulator. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11(10):2062. DOI: 10.17762/ijritcc.v11i10.8891.

Sánchez Juanola, M. (2019). Speed Traffic Sign Detection on the CARLA Simulator Using YOLO. Master's thesis, Universitat Pompeu Fabra.

Wachenfeld, W. and Winner, H. (2016). The Release of Autonomous Vehicles. In Maurer, M. *et al*., editors, *Autonomous Driving: Technical, Legal and Social Aspects*, pages 425–449. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-662-48847-8_21.

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., and Ding, G. (2024). Yolov10: Real-time end-to-end object detection.

Wang, C.-Y. and Liao, H.-Y. M. (2024). Yolov1 to yolov10: The fastest and most accurate real-time object detection systems. *APSIPA Transactions on Signal and Information Processing*, 13(1):1–30. DOI: 10.1561/116.20240058.

Wu, X. and Cao, H. (2022). Traffic sign detection algorithm based on improved yolov4. *Journal of Physics: Conference Series*, 2258(1):012009. DOI: 10.1088/1742-6596/2258/1/012009.

Yaseen, M. (2024). What is yolov8: An in-depth exploration of the internal features of the next-generation object detector.