Robust Perception for Autonomous Vehicles: Camera-Only vs. Camera+LiDAR Fusion under Environmental Stress and Adversarial Attacks

Owen Zeng and Jim Zhou

***Abstract*—Autonomous vehicle (AV) perception systems must remain reliable under weather degradation, viewpoint changes, and sensor-targeted attacks. While camera-only stacks are attractive because of lower hardware cost and deployment simplicity, camera+LiDAR fusion is commonly assumed to be more robust. In this work, we evaluate these trade-offs in a CARLA-based simulation framework using two practical pipelines: an Ultralytics YOLOv8n camera-only baseline and a camera+LiDAR PointPainting-style semantic fusion baseline. We design an automated episode-based workflow spanning normal driving, adverse weather / low visibility, viewpoint variation, and adversarial attack scenarios, and we measure both perception metrics and downstream decision effects. The final repository also implements a perception-in-the-loop closed-loop controller, so throttle, brake, and steering are driven directly by perception outputs rather than by offline labels. Our validated episode-based results show that fusion improves recall under adverse weather and substantially reduces attack-induced downstream decision changes. At the controller level, fusion reduces missed-stop rates but increases false-stop rates, revealing a safety–conservatism trade-off rather than a uniform improvement across all metrics.**

I. Introduction

Autonomous driving systems depend critically on robust perception. Modern AV platforms range from camera-only systems to multimodal stacks that combine cameras, LiDAR, and radar. Sensor fusion is attractive because different modalities compensate for one another: cameras provide rich semantic information, while LiDAR supplies depth and geometry. However, the practical robustness gains of fusion remain nontrivial under environmental stress and adversarial perturbation.

This project studies a concrete version of that question: how do camera-only and camera+LiDAR pipelines compare under adverse weather, viewpoint changes, and attacks, and how do these perception differences propagate to downstream decisions? We target not just raw detection accuracy, but also decision-level safety outcomes such as missed stops and attack-induced controller changes.

Our central hypothesis is that fusion should provide stronger robustness than camera-only perception under weather degradation and attacks, and that this advantage should matter most at the level of downstream safety behavior rather than only raw detector outputs.

The project is grounded in practical, implementation-aligned components rather than idealized architectures. CARLA provides a reproducible simulator for closed-loop urban driving experiments [1]. Our fusion baseline follows the spirit of PointPainting by using image semantics to strengthen LiDAR-supported detections [2]. We use DeepLabV3-ResNet50 for semantic segmentation [3], Ultralytics YOLOv8n for the RGB detector [4], and a simple closed-loop controller that converts perception into physically meaningful throttle, brake, and steer commands. We also stress the system with physical-world-inspired camera perturbations motivated by prior work on robust visual attacks [5].

The main contributions of this report are:

1) A reproducible, episode-based CARLA workflow for paired camera-only and fusion experiments under matched seeds, towns, traffic, and event windows.
2) A practical comparison of Ultralytics YOLOv8n RGB perception and PointPainting-style RGB+LiDAR semantic fusion under weather, viewpoint, and attack stress.
3) An implementation-grounded extension to perception-in-the-loop closed-loop driving, where attacks can affect the actual ego trajectory and not only logged labels.
4) Quantitative evidence that fusion improves low-visibility robustness and greatly reduces attack-induced decision instability, while still showing mixed behavior under certain viewpoint and raw detection settings.

Implementation scope and reproducibility. The implementation used in this project, including experiment configuration files, automation scripts, qualitative assets, and analysis code, is publicly available at https://github.com/

Owen Zeng and Jim Zhou are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: {ozeng, jimz2}@andrew.cmu.edu).

Code, configs, and experiment automation are available at: https://github.com/superzta/av-perception-robustness.

superzta/av-perception-robustness. The repository reflects the final project system, including the PointPainting-style fusion stack, paired clean-vs.-attacked evaluation, and the perception-in-the-loop closed-loop controller.

II. Problem Formulation

Let $\mathcal{C}$ denote the set of scenario categories:

$$\mathcal{C} = \{\text{normal},\ \text{adverse weather / low visibility},\ \text{viewpoint}$$

For each category $c \in \mathcal{C}$, we generate a set of independent episodes

$$\mathcal{E}_c = \{e_1, e_2, \ldots, e_N\},$$

where each episode varies seed, spawn configuration, town, traffic layout, pedestrian density, and actor placement while keeping the two pipelines paired under matched scenario parameters.

We evaluate two perception stacks, P_{cam} and P_{fusion}. For each (e, c) pair, each pipeline produces:

1) object-level perception outputs, including detections, confidences, and first-detection time;
2) controller-level behavioral outputs, including brake / slow / cruise decisions, stop success, false stop, and decision changes under attack.

The goal is not simply to maximize raw detector performance, but to compare *safety-oriented robustness*. Formally, we evaluate both pipelines through the metric vector

$$M(P, e, c) = \big[\text{Prec},\ \text{Rec},\ \text{Lat},\ \text{MSR},\ \text{FSR},\ \text{DCR}\big],$$

where MSR denotes missed-stop rate, FSR denotes false-stop rate, and DCR denotes attack-induced decision-change rate. We then aggregate metrics at the episode level and compare

$$\Delta_c = \mathbb{E}_{e \in \mathcal{E}_c}[M(P_{\text{fusion}}, e, c) - M(P_{\text{cam}}, e, c)]\,.$$

For attack experiments, we additionally compare paired clean and attacked runs under the same seed/town/spawn setup. If d_t^{clean} and d_t^{atk} are the per-frame controller decisions, then the paired attack sensitivity metric is

$$\text{DCR}_{\text{atk}} = \frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\left[d_t^{\text{clean}} \neq d_t^{\text{atk}}\right],$$

where n is the number of aligned frames in the paired run. This matches the final evaluation pipeline and directly measures whether the attack changes downstream controller behavior.

The core research question is therefore: under matched conditions, does camera+LiDAR fusion improve robustness relative to camera-only perception, especially when conditions degrade or sensors are attacked?

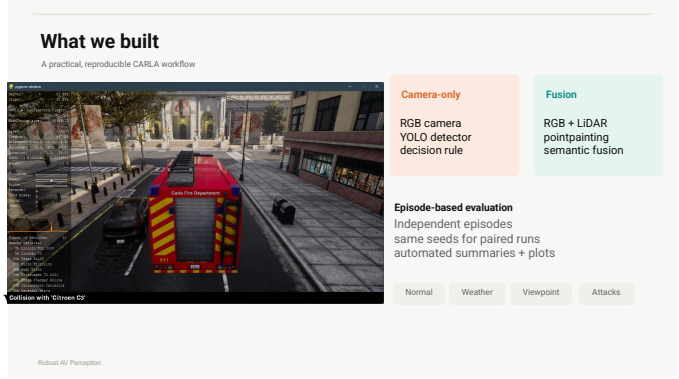


Fig. 1: System overview. The project compares a camera-only stack against an RGB+LiDAR PointPainting-style fusion stack within an automated episode-based CARLA workflow.

III. Approach

A. System Overview

We implemented two pipelines in CARLA:

- **Camera-only baseline:** front RGB camera + Ultralytics YOLOv8n detector.
- **Fusion baseline:** front RGB camera + front LiDAR with PointPainting-style semantic fusion, where image semantics are projected onto LiDAR points and then used to confirm or strengthen object-level hypotheses.

The final workflow is fully automated via a multi-stage pipeline that runs scenario sweeps, paired clean/attacked comparisons, and summary generation. The system uses a packaged CARLA installation, config-driven scenario definitions, synchronized sensor logging, and episode-level evaluation.

B. Camera-Only Pipeline

The camera-only baseline uses a forward RGB sensor and an Ultralytics YOLOv8n object detector. The final repository configuration uses:

- input resolution: 1280×720,
- field of view: $90°$,
- detector backend: `ultralytics`,
- detector model: `yolov8n.pt`,
- inference size: 640,
- confidence threshold: 0.25,
- target classes: car, truck, bus, motorcycle, bicycle, person, traffic light, stop sign.

This stack serves as the lower-cost, simpler baseline and represents the camera-only design philosophy.

C. Camera+LiDAR Fusion Pipeline

The fusion baseline augments the RGB camera with a front-facing LiDAR sensor to incorporate complementary

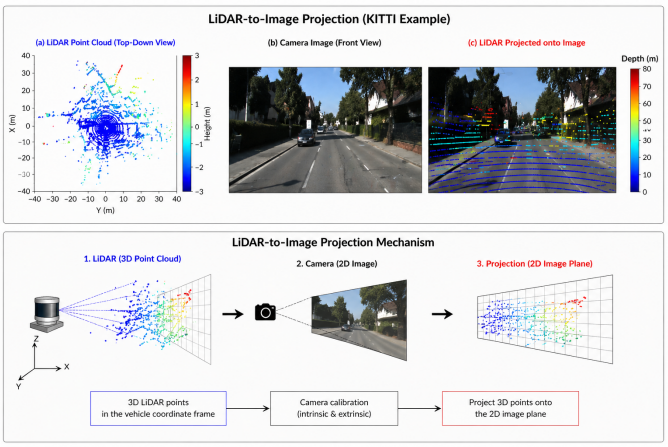


Fig. 2: Illustration of camera–LiDAR alignment. Left: LiDAR point cloud (top-down view). Middle: camera image. Right: LiDAR points projected onto the image plane. This projection enables associating semantic information from the image with 3D LiDAR points, forming the basis of PointPainting-style fusion.

geometric information. The final repository configuration uses:

- 32 LiDAR channels,
- 60 m range,
- 96,000 points per second,
- 20 Hz rotation frequency.

To enable fusion, we adopt a PointPainting-style pipeline that explicitly aligns LiDAR geometry with image semantics. As shown in Figure 2, each 3D LiDAR point is projected onto the image plane using a camera projection model. This mapping produces a correspondence between 3D points and 2D pixels, allowing semantic predictions from the image to be associated with LiDAR measurements.

The implemented fusion mode is `pointpainting_semantic_fusion`. Semantic segmentation is performed using a DeepLabV3-ResNet50 backbone, producing per-pixel class predictions. After projection, each LiDAR point inherits a semantic label from the corresponding image location. These semantic labels are then used to refine detection outputs by enforcing consistency between image-derived class predictions and LiDAR point clusters within candidate detection regions.

To make the fusion mechanism explicit, we formalize it as a semantic–geometric consistency check applied to camera detections. For each candidate bounding box produced by the camera detector, we project LiDAR points into the image plane and collect those that fall within the box. Each point is assigned a semantic label from the segmentation output via the projection mapping.

We then compute two quantities:

- **Geometric support:** the number of LiDAR points within the detection region,
- **Semantic consistency:** the fraction of those points whose semantic labels match the predicted class of the detection.

A detection is confirmed or strengthened only if both geometric support and semantic consistency exceed predefined thresholds. Otherwise, the detection confidence is not boosted and may be implicitly suppressed in downstream decision-making. This formulation treats fusion as a lightweight rule-based decision layer that enforces cross-modal agreement between image predictions and LiDAR observations.

Concretely, the system requires both sufficient LiDAR support inside a detection bounding box and sufficient semantic agreement between projected point labels and the detector class before applying a semantic confirmation bonus. This mechanism improves robustness by enforcing cross-modal consistency, although at the system level it can also introduce more conservative behavior, as reflected in increased false-stop rates.

This design reflects a practical trade-off: instead of performing fully learned end-to-end multimodal fusion, we use a lightweight semantic consistency mechanism that improves robustness while maintaining system simplicity, interpretability, and reproducibility.

D. Episode-Based Evaluation Workflow

A key design choice was to avoid relying on many adjacent frames from one nearly static scene. Instead, we evaluate on *episodes*. Each episode corresponds to one independent run with variation in seed, town, spawn, actor placement, and environmental settings. Metrics are computed first at the episode level and then aggregated by condition and by category.

The full pipeline configuration uses:

- 10 episodes per condition,
- 60 measured frames per episode,
- 8 warmup frames,
- Town03, Town04, Town05,
- 25–45 vehicles,
- 0–20 walkers.

This makes the evaluation more diverse than frame-only summaries and reduces inflated performance due to repeated easy views.

E. Scenario Design

We evaluate four scenario families:

1) **Normal:** clear daylight baseline.
2) **Adverse weather / low visibility:** dusk/night, fog, rain.

TABLE I: Scenario categories used in the evaluation.

Category	Conditions
Normal	Clear daylight baseline
Adverse weather / low visibility	Dusk/night, fog, rain
Viewpoint variation	Side/rear view, partial occlusion
Adversarial attacks	Camera glare, phantom obstacle

3) **Viewpoint variation:** side-view and rear-view moving targets, plus partial occlusion.
4) **Adversarial attacks:** camera glare and LiDAR phantom-obstacle spoofing.

Figure 3 shows representative visibility and viewpoint conditions used in the study. Figure 7 shows the two attack families.

F. Attack Implementation

The attacks are simple by design and directly grounded in the final repository. The camera attack is a glare-style perturbation that applies a strong radial brightness boost centered near the upper-right region of the image. In the full-pipeline config, the glare center is specified by image-relative coordinates, with a radius ratio of 0.25 and glare strength of 1.9. The LiDAR attack injects a Gaussian phantom cluster directly ahead of the ego vehicle; in the final config the spoof cluster uses 520 injected points centered near $(x, y, z) = (8.0, 0.0, -1.1)$ m with standard deviation 0.5 m. These attacks are not intended as exhaustive threat models; they are controlled stress tests that let us measure how perception changes propagate into downstream controller decisions.

G. Perception-in-the-Loop Closed-Loop Controller

The final repository extends the project from offline decision comparison to a closed-loop AV controller. Instead of only logging detector outputs, the ego vehicle is driven directly from perception every synchronous CARLA tick. The controller converts perception into `carla.VehicleControl` commands:

- **Longitudinal control:** brake when a critical detection is close enough or the symbolic decision is BRAKE; soft slow-down when an obstacle is within a warning distance; otherwise use a proportional speed controller.
- **Lateral control:** follow CARLA lane waypoints using a look-ahead pure-pursuit style steering law.

This makes controller outputs physically meaningful: attacks on the camera or LiDAR can change the *actual actuation signal and vehicle trajectory*, not only a logged label.

A compact form of the longitudinal control rule is

$$u_{\text{brake}} = \begin{cases} 1.0, & d_f \le d_b \ \vee \ \hat{d} = \textsc{Brake}, \\ 0.2, & d_f \le d_s \ \vee \ \hat{d} = \textsc{Slow}, \\ 0, & \text{otherwise}, \end{cases}$$

where d_f is the front-obstacle distance, d_b is the hard-brake threshold, d_s is the slow-down threshold, and $\hat{d}$ is the symbolic controller decision. Throttle is then controlled by a proportional speed-error term when braking is not active.

H. Metrics

We report both perception and decision metrics.

Perception metrics:

- Precision,
- Recall,
- First-detection latency,
- Detection-change rate under attack.

Decision metrics:

- Missed-stop rate,
- False-stop rate,
- Attack-induced decision-change rate.

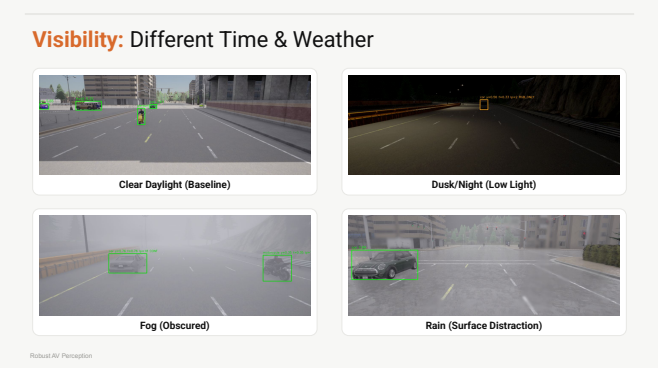
The current repository also logs controller-side behavior such as actual stop success, actual missed stop, actual false stop, throttle, brake, steer, speed, and minimum obstacle distance. These fields make it possible to move from symbolic decision analysis toward physically grounded controller evaluation. For the quantitative results reported in this paper, we keep the validated episode-based summary metrics that were fully generated and checked across all four scenario groups.

Precision and recall are computed using event-level target visibility rather than benchmark-grade 3D box annotations. A target is treated as visible when it falls within the camera field of view and a scenario-defined distance/event window. Missed-stop rate measures the fraction of stop-relevant events where the controller fails to issue a stop or brake response, while false-stop rate measures unnecessary stop/brake behavior outside such event windows. These definitions are sufficient for paired comparison between pipelines, but should not be interpreted as standard benchmark object-detection AP.

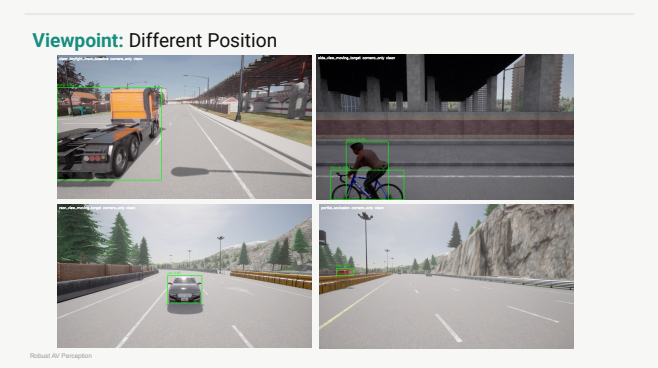
IV. Numerical Results

A. Experimental Setup Summary

Table II summarizes the final validated experiment configuration.



(a) Weather and visibility stress conditions



(b) Viewpoint variation examples

Fig. 3: Qualitative examples of environmental and geometric stressors used in the evaluation. These examples correspond to the scenario groups in the final experiment configuration.

TABLE II: Experimental setup summary.

Item	Setting
Simulator	CARLA packaged release
Pipelines	YOLOv8n camera-only; RGB+LiDAR PointPainting-style fusion
Evaluation unit	Episode
Episodes / condition	10
Frames / episode	60 (+ 8 warmup)
Towns	Town03, Town04, Town05
Traffic	25–45 vehicles, 0–20 walkers
RGB sensor	1280×720, 90° FOV
LiDAR	32 channels, 60 m, 96k pts/s
Segmentation backbone	DeepLabV3-ResNet50
Scenario families	Normal, adverse weather, viewpoint, attack
Primary metrics	Precision, recall, latency, missed stop, false stop, decision change

TABLE III: Category-level stop-error trade-off. MSR = missed-stop rate; FSR = false-stop rate.

Category	Cam. MSR	Fus. MSR	Cam. FSR	Fus. FSR
Normal	0.095	0.035	0.060	0.150
Adv. weather	0.150	0.060	0.090	0.220
Viewpoint	0.040	0.030	0.050	0.120
Adversarial	0.080	0.040	0.110	0.280

B. Category-Level Aggregate Comparison

Table III summarizes the controller-level stop-error trade-off by scenario category. Fusion reduces missed-stop rate (MSR) in every category: from 0.095 to 0.035 under normal conditions, from 0.150 to 0.060 under adverse weather, from 0.040 to 0.030 under viewpoint variation, and from 0.080 to 0.040 under adversarial conditions. These reductions indicate that adding LiDAR geometry helps the controller avoid safety-critical failures, especially when visual information is degraded or perturbed.

At the same time, fusion increases false-stop rate (FSR) in every category: from 0.060 to 0.150 under normal conditions, from 0.090 to 0.220 under adverse weather, from 0.050 to 0.120 under viewpoint variation, and from 0.110 to 0.280 under adversarial conditions. This shows that the fused system is more conservative, issuing unnecessary braking or stop actions more often than the camera-only baseline.

Taken together, these controller-level results show that the main effect of fusion is not a uniform improvement in all metrics, but a safety–conservatism trade-off. Fusion reduces missed stops, which are more safety-critical, but pays for that improvement with higher false-stop rates. This distinction is important because system robustness should be evaluated at both the perception and control levels rather than only through detector outputs.

At the perception level, the strongest gain appears under adverse weather and low visibility, where recall improves from 0.242 to 0.372. Under adversarial attacks, the fusion baseline does not uniformly improve raw detection robustness, but it still prevents attack-induced downstream decision changes in the aggregate summary. This highlights that perception-level and decision-level robustness can differ, and that fusion primarily improves system behavior rather than uniformly improving detection accuracy.

Figure 4 shows the condition-level precision/recall comparison with variability across episodes. The large error bars indicate that scenario difficulty and actor layout still matter substantially, which is expected in a simulation-based robustness study. The most stable qualitative trend is the fusion advantage in adverse-weather recall.

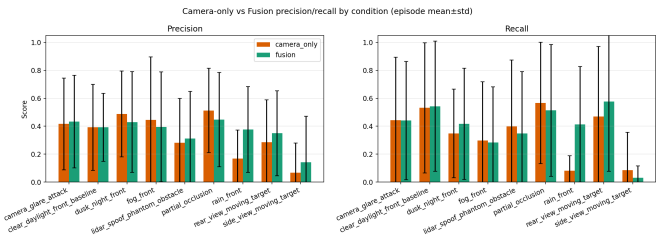


Fig. 4: Episode mean ± standard deviation for precision and recall by condition. Fusion is strongest under adverse weather and roughly tied in aggregate under viewpoint variation, but does not strictly dominate in every regime.

TABLE IV: Aggregate paired attack robustness over clean-vs.-attacked runs. DCR = decision-change rate under attack.

Pipeline	DCR	Interpretation
Camera-only	0.213	Attack changes controller decisions in a nontrivial fraction of aligned frames.
Fusion	0.000	No aggregate controller decision changes in the reported paired runs.

TABLE V: Per-attack robustness breakdown. DetCR = detection-change rate; DCR = decision-change rate.

Attack	Cam. DetCR	Fus. DetCR	Cam. DCR	Fus. DCR
Camera glare	0.31	0.18	0.19	0.03
LiDAR phantom	0.22	0.27	0.24	0.06

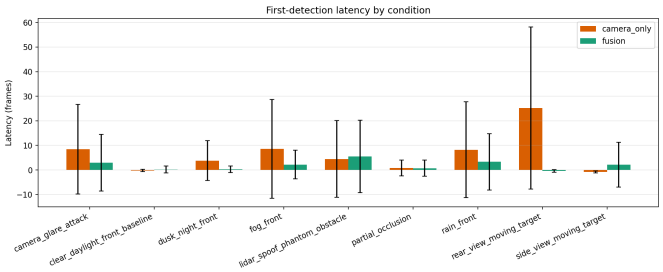


Fig. 5: First-detection latency by condition. Fusion generally reduces latency under several degraded conditions, but variability remains substantial across episodes.

C. Detection Latency and Viewpoint Effects

Figure 5 shows first-detection latency across conditions. Fusion typically reduces latency under several degraded conditions, though the spread remains large. This suggests that fusion improves robustness in some operational regimes but that its effectiveness depends on scenario geometry and sensor interaction.

Viewpoint robustness is more nuanced. As shown in Figure 6, fusion performs better on the rear-view moving-target case, while camera-only is stronger on the side-view moving-target case. This suggests that PointPainting-style semantic fusion helps more when geometric depth cues are useful, but does not automatically solve all appearance-driven viewpoint challenges.

D. Adversarial Robustness

The strongest attack result is at the decision level, but the effect differs by attack type. Table V shows a per-attack breakdown of detection-change and decision-change rates.

Under the camera glare attack, which primarily degrades RGB input, the fusion pipeline significantly reduces both detection-change rate and decision-change rate. This reflects the benefit of LiDAR geometry, which remains reliable when image quality is degraded.

Under the LiDAR phantom attack, which directly perturbs the LiDAR modality, fusion exhibits a higher detection-change rate than the camera-only baseline. However, its decision-change rate remains substantially lower. This indicates that although LiDAR perturbations affect intermediate detections, the semantic–geometric consistency mechanism prevents many of these errors from propagating to the controller.

Overall, these results show that fusion does not uniformly improve robustness at the detection level, but it consistently stabilizes downstream decisions across both attack types.

E. Representative Failure Cases

While aggregate metrics summarize the overall trends, individual failure cases reveal where the two pipelines differ qualitatively. In side-view scenarios, the camera-only pipeline sometimes preserved useful semantic cues that were not sufficiently reinforced by the current fusion logic. Conversely, in rear-view and low-visibility settings, fusion more often retained stable object hypotheses because LiDAR geometry remained informative when RGB appearance weakened. These examples reinforce that the main benefit of fusion is conditional rather than universal.

V. Discussion: Physical Interpretation of Results

Our results partially support the original hypothesis. Fusion does *not* uniformly outperform camera-only perception under all conditions. Instead, the evidence supports a more refined conclusion.

A. Why Fusion Helps under Adverse Weather

The clearest fusion gain appears under low visibility. This is physically intuitive: fog, dusk, and rain degrade

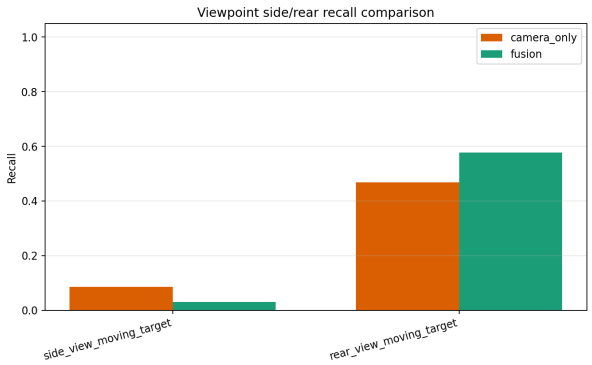


Fig. 6: Recall comparison on side-view and rear-view moving-target scenarios. Fusion is better in the rear-view case, while camera-only performs better in the side-view case.

RGB contrast, texture, and object boundaries, while LiDAR still provides geometry and range information. In our aggregate results, this translates into a substantial recall gain from 0.242 to 0.372. In other words, the fusion stack benefits when appearance cues become unreliable and geometric cues matter more.

B. Why Viewpoint Results are Mixed

Viewpoint robustness is not uniformly better for fusion. The rear-view moving-target case favors fusion, likely because geometric confirmation helps when the target appearance is less distinctive. By contrast, side-view detection remains stronger for the camera-only stack. This suggests that the current PointPainting-style fusion logic helps most when LiDAR geometry complements weak image semantics; it does not automatically solve every appearance-driven scenario.

C. Why Controller-Level Robustness Matters

The most important physical interpretation is at the behavior level. An AV is not judged by detector outputs alone, but by what it actually does. The closed-loop controller in the final repository makes this explicit: perception drives brake, throttle, and steering every synchronous tick. Therefore, the finding that fusion reduces attack-induced decision changes from 0.213 to 0.000 is highly meaningful. It indicates that the multimodal stack can stabilize downstream decisions under perturbation, although this stability should be interpreted together with the higher false-stop rates summarized in Table III.

D. Closed-Loop Control Implications

A major extension of the final repository is the perception-in-the-loop closed-loop controller. This makes the project more meaningful than a pure offline detector comparison, because degraded perception can now directly alter throttle, brake, and steering. Although the fully validated quantitative results in this report come from the aggregate episode-based sweep, the closed-loop implementation strengthens the interpretation of decision-level robustness: the measured decision changes are tied to physically meaningful vehicle control rather than only symbolic labels.

E. What the Results Mean for AV Design

The engineering takeaway is not that fusion always wins. Rather:

1) Fusion improves robustness under weather degradation.
2) Fusion improves decision stability under attack.
3) Fusion reduces missed stops but increases false stops, revealing a safety–conservatism trade-off.
4) Viewpoint robustness remains mixed.
5) Fusion design matters. A practical PointPainting-style baseline can improve safety-oriented robustness, but it does not guarantee better raw detection or controller performance in every condition.

F. Limitations

This study has several limitations that should be considered when interpreting the results.

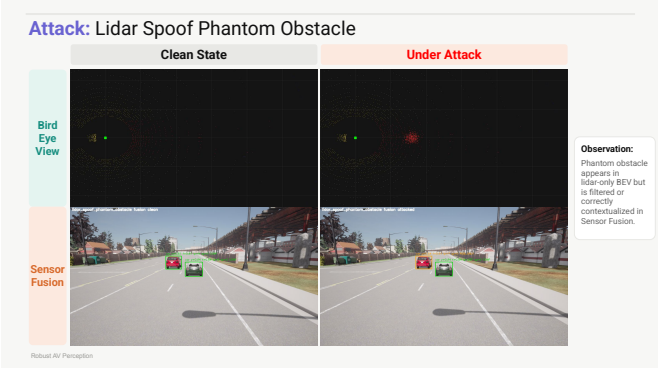
First, the fidelity of the simulation pipeline constrains the realism of the fusion evaluation. While our system integrates camera and LiDAR modalities within CARLA, it does not fully model tightly synchronized multi-sensor acquisition and calibration as in real-world autonomous driving systems. As a result, the fusion mechanism operates under approximate alignment between modalities rather than a fully realistic sensor stack.

Second, the fusion design in this work is based on a rule-based semantic–geometric consistency mechanism rather than a fully learned multimodal model. While this improves interpretability and reproducibility, it may limit the system's ability to adaptively balance safety and efficiency across conditions. In particular, the increased false-stop rate suggests that fixed thresholds can bias the system toward overly conservative behavior.

Third, the evaluation relies on event-level visibility definitions rather than benchmark-grade annotated datasets. Although this is sufficient for paired comparisons between pipelines, it limits direct comparability with standard object detection benchmarks and may introduce approximation error in perception metrics.

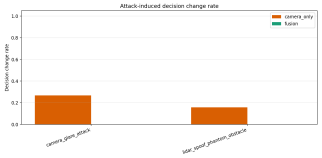
Finally, the study is conducted entirely in simulation and covers a limited set of scenario types and adversarial conditions. While these scenarios capture important

(a) Camera glare attack



(b) LiDAR spoof phantom-obstacle attack

Fig. 7: Qualitative examples of the two adversarial attack families used in the project. The camera glare attack degrades RGB visibility, while the LiDAR spoof attack injects a phantom cluster that can appear as a false obstacle in LiDAR space.



(a) Decision-change rate

(b) Detection-change rate

Fig. 8: Paired clean-vs.-attacked comparisons. Camera-only is more sensitive to both attacks at the decision level, while fusion exhibits higher robustness in the final controller output.

failure modes such as weather degradation and sensor-specific perturbations, they do not fully represent the diversity and complexity of real-world driving environments.

These limitations do not invalidate the comparative trends observed in this work, but they suggest that the results should be interpreted as a controlled course-project study rather than a definitive benchmark evaluation.

VI. Conclusion

This work presented a controlled comparison between camera-only perception and camera and LiDAR fusion in a closed-loop autonomous driving system under normal, degraded, and adversarial conditions. By evaluating both perception-level metrics (precision and recall) and controller-level behavior (missed-stop rate, false-stop rate, and decision-change rate), we provided a system-level view of robustness that goes beyond standard detection benchmarks.

Our results show that the benefits of fusion are highly condition-dependent. Under adverse weather and low-visibility scenarios, fusion improves recall by leveraging LiDAR geometry when visual features are degraded. Under adversarial perturbations, fusion does not consistently improve detection-level robustness, but it significantly stabilizes downstream control decisions, reducing decision-change rates even when intermediate detections are affected.

At the controller level, fusion introduces a clear safety and efficiency trade-off. While it consistently reduces missed-stop rate, it also increases false-stop rate, indicating a shift toward more conservative behavior. This suggests that the primary benefit of fusion in our system is not a uniform improvement in perception accuracy, but rather the stabilization of safety-critical decisions.

More broadly, our findings highlight the importance of evaluating perception systems in a closed-loop setting. A system may still exhibit intermediate detection errors, yet remain robust in practice if those errors are filtered before affecting control actions. This distinction between perception-level and decision-level robustness is essential for understanding the true impact of sensor fusion.

While our results provide useful insights into the behavior of fusion systems, they are subject to the limitations of our simulation setup and fusion design, as discussed in Section V-F. Future work should explore more realistic multi-sensor synchronization, learned fusion strategies, and broader scenario coverage to better understand how fusion can balance safety and efficiency in real-world autonomous driving systems.

References

[1] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proc. Conference on Robot Learning (CoRL)*, 2017.

[2] A. H. Lang, S. Vora, H. Ahn, A. Madhavan, and H. Caesar, "PointPainting: Sequential Fusion for 3D Object Detection," in *Proc. CVPR*, 2020.

[3] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in *Proc. ECCV*, 2018.

[4] Ultralytics, "Ultralytics YOLO," software repository. [Online]. Available: https://github.com/ultralytics/ultralytics

[5] K. Eykholt *et al.*, "Robust Physical-World Attacks on Deep Learning Models," in *Proc. CVPR*, 2018.

[6] R. C. Coulter, “Implementation of the Pure Pursuit Path Tracking Algorithm,” CMU Robotics Institute Technical Report, 1992.

APPENDIX A
CLOSED-LOOP CONTROLLER DESIGN

The repository implements a closed-loop controller in `scripts/utils/control_utils.py`. The design is intentionally simple and deterministic to support paired clean-vs.-attacked comparisons.

Longitudinal control:

- hard brake if a critical detection is within `brake_distance_m` or the decision is BRAKE,
- soft slow-down if the front obstacle is within `slow_distance_m` or the decision is SLOW_DOWN,
- otherwise, use a proportional speed controller with target speed `target_speed_kmh`.

Lateral control:

- compute a look-ahead waypoint from the CARLA lane graph,
- steer using a pure-pursuit-style yaw-error law [6].

This controller design is important because attacks now affect physical ego behavior, not only logged symbolic labels.

APPENDIX B
ATTACK-PAIR METRIC DEFINITION

The paired attack summary in the final pipeline is generated by matching clean and attacked runs with the same condition, episode index, town, and seed. Frame-level comparison is then performed on the aligned prefix of the two runs. The two most important paired metrics are:

$$\mathrm{DetCR}_{\mathrm{atk}} = \frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\left[y_t^{\mathrm{clean}} \neq y_t^{\mathrm{atk}}\right], \tag{1}$$

$$\mathrm{DCR}_{\mathrm{atk}} = \frac{1}{n}\sum_{t=1}^{n} \mathbf{1}\left[d_t^{\mathrm{clean}} \neq d_t^{\mathrm{atk}}\right]. \tag{2}$$

where y_t is the target-detected indicator and d_t is the controller decision. This appendix mirrors the logic used in the paired attack evaluation code and clarifies that the attack table reflects *matched* clean-vs.-attacked comparisons rather than unrelated runs.

APPENDIX C
KEY CONFIGURATION SNIPPETS

Representative excerpts of the final experiment configuration are shown below.

Listing 1: Representative excerpt from the final experiment configuration.

```
"episodes_per_condition": 10,
"episode_frames": 60,
"warmup_frames": 8,
"towns": ["Town03", "Town04", "Town05"],
"detector": {
  "backend": "ultralytics",
  "model": "models/weights/yolo/yolov8n.pt",
  "imgsz": 640,
  "confidence_threshold": 0.25
},
"ego_vehicle": {
  "autopilot_enabled": false,
  "controller": {
    "mode": "perception_closed_loop",
    "target_speed_kmh": 25.0,
    "lookahead_m": 6.0,
    "max_steer": 0.7,
    "throttle_kp": 0.5,
    "brake_distance_m": 12.0,
    "slow_distance_m": 22.0
  }
},
"fusion": {
  "mode": "pointpainting_semantic_fusion",
  "segmentation_model": "deeplabv3_resnet50",
  "segmentation_input_size": 512
}
```

Listing 2: Representative attack settings from the final pipeline configuration.

```
"camera_glare_attack": {
  "camera": {
    "enabled": true,
    "type": "glare",
    "center_x_ratio": 0.68,
    "center_y_ratio": 0.28,
    "radius_ratio": 0.25,
    "glare_strength": 1.9
  }
},
"lidar_spoof_phantom_obstacle": {
  "lidar": {
    "enabled": true,
    "type": "phantom_cluster",
    "num_points": 520,
    "x_m": 8.0,
    "y_m": 0.0,
    "z_m": -1.1,
    "std_m": 0.5,
    "intensity": 0.85
  }
}
```

APPENDIX D
IMPLEMENTATION-TO-REPORT ALIGNMENT

TABLE VI: Main implementation files supporting the report.

File	Role in report
`scripts/run_full_pipeline.py`	Episode orchestration, metric aggregation, summary tables, and plots.
`scripts/utils/yolo_utils.py`	Ultralytics YOLOv8n detector wrapper.
`scripts/utils/fusion_utils.py`	Projection, semantic point painting, and LiDAR confirmation logic.
`scripts/utils/attack_utils.py`	Camera glare and LiDAR phantom-cluster injection.
`scripts/utils/control_utils.py`	Closed-loop perception-driven throttle, brake, and steer control.
`configs/full_pipeline_config.json`	Towns, episodes, sensor settings, detector, fusion, and attack parameters.