Real-World Perturbation Testing of Autonomous Driving Systems

Stefano Carlo Lambertenghi
stefanocarlo.lambertenghi@tum.de
Technical University of Munich
fortiss GmbH
Germany

Matthias Weil
matthias.weil@tum.de
Technical University of Munich
Germany

Andrea Stocco
andrea.stocco@tum.de
Technical University of Munich
fortiss GmbH
Germany

ABSTRACT

Autonomous Driving Systems (ADS) must operate reliably under diverse conditions, yet representative data for rare or adverse scenarios is difficult to obtain. Perturbation-based testing is widely used to assess robustness, but most studies focus on offline datasets or simulation, leaving open questions about how such results translate to real-world driving. We present a large-scale study of 72 camera and LiDAR perturbations, evaluated across three testing modalities: offline model-level analysis, hardware-in-the-loop execution, and closed-loop system-level testing on a full-scale autonomous vehicle. The study covers both an end-to-end vision-based driving model and a modular LiDAR-based perception and planning stack.

Our results reveal a clear gap between testing levels. For camera-based systems, perturbations with limited offline impact can still induce unstable control and failures in real-world driving. For LiDAR-based systems, degradation is more consistent at the perception level but weakly predictive of system-level failures. Across both modalities, model-level metrics alone are insufficient to identify the most harmful perturbations. We further show that real-time feasibility is a key constraint in real-world testing, and that robustness observations obtained from recorded data do not consistently transfer to closed-loop behavior on a physical vehicle, highlighting the importance of complementary real-world, system-level evaluation.

1 INTRODUCTION

Autonomous Driving Systems (ADS) rely on perception pipelines built on heterogeneous sensors such as cameras and LiDAR to interpret dynamic environments and support downstream driving decisions [22, 41, 60, 67]. Ensuring robustness across diverse operating conditions remains difficult, since even small input perturbations can degrade perception and propagate to unsafe system-level behaviour, especially under out-of-distribution (OOD) conditions [14, 21, 27, 64].

One of the main limitations is the lack of representative data. Rare but safety-critical conditions, such as adverse weather, illumination changes, sensor noise, and environmental artifacts, are difficult to collect systematically and are often underrepresented in training and evaluation datasets [2, 34], limiting testing effectiveness.

To address this problem, prior work has widely explored *perturbation testing*, where controlled transformations are applied to sensor inputs to emulate challenging conditions [28, 29, 35, 47, 55]. In computer vision, benchmarks such as ImageNet-C, ImageNet-P, and MNIST-C established perturbation-based robustness evaluation as a standard offline practice [27, 28, 48], while augmentation methods such as RandAugment and AugMix improve robustness through adversarial training [12, 29]. In autonomous driving, perturbations have been applied to camera images [51, 61, 61, 70] and LiDAR point clouds [6, 15, 16], but these studies use static datasets, where downstream driving effects cannot be observed [26, 58, 60].

Recent work has integrated perturbations into simulation-based testing to enable closed-loop evaluation [5, 11, 36, 42–45, 49, 50, 57, 63, 66]. These approaches better capture temporal effects but remain constrained by simulation fidelity, as failures observed in simulation do not necessarily transfer to real-world deployments [36, 37, 69]. It therefore remains unclear whether perturbation-induced degradations observed offline translate into failures during real-world closed-loop driving [25, 26, 58], despite continued evidence of incidents highlighting the importance of real-world testing [8, 23, 31, 39, 54, 56].

To address this, we present a comprehensive evaluation of camera and LiDAR perturbations, benchmarking a wide range of techniques from the literature across two ADAS tasks of increasing complexity, a vision-based end-to-end driving model, and a modular LiDAR-based perception and planning stack, and three testing modalities: offline model-level evaluation on static datasets, hardware-in-the-loop (HiL) testing on the vehicle compute stack, and vehicle-in-the-loop (ViL) testing on a real autonomous vehicle.

First, we perform model-level evaluation on recorded real-world datasets to quantify how perturbations affect model outputs and task-specific metrics. Second, we deploy perturbations in a HiL setup to assess real-time feasibility on automotive hardware. Third, we inject perturbations into live sensor streams during closed-loop driving on a full-scale autonomous vehicle to directly observe their impact on real driving behaviour. We also evaluate whether fine-tuning with perturbed data improves the robustness of the end-to-end driving model under real-world OOD conditions.

Our results show clear differences across modalities and testing levels. Camera perturbations often induce high-variance control deviations: low average offline errors can still lead to unstable driving. LiDAR perturbations instead produce more consistent degradation, mainly reducing detection retention while preserving comparatively stable localization. We also find limited transfer across testing levels: perturbations that appear mild offline can still trigger lane departures or collisions in closed-loop driving, while others that strongly affect model-level metrics have limited observable system-level impact. Most perturbations satisfy real-time constraints on the tested automotive hardware, and fine-tuning with perturbed data improves robustness under naturalistic perturbations such as rain, glare, and sensor artifacts while preserving nominal performance.

Our paper makes the following contributions:

Framework A ROS-based framework of 72 perturbations for conducting robustness testing across model-level, HiL, and ViL deployment for camera and LiDAR sensors.

Empirical Study To the best of our knowledge, this is the first software engineering paper to conduct cross-modality, cross-level robustness analysis in the context of real-world testing of ADS at the system-level with a real-world full-size vehicle, on end-to-end and modular driving models.

Findings We identify a gap between model-level robustness and system-level behavior, showing that perturbations with limited offline impact can still induce failures in closed-loop driving, and that real-time feasibility is a critical constraint for real-world system-level ADS testing.

2 IMAGE AND LIDAR PERTURBATIONS

We study 72 perturbations from the literature for both camera images and LiDAR point clouds, the two most important ADS sensors [60]. For simplicity of exposure, in this paper, we assign a compact identifier to each perturbation: camera perturbations are denoted as **C-X$_{\mathbf{Y}}$**, whereas LiDAR perturbations are denoted as **L-X$_{\mathbf{Y}}$**, where **X** identifies the perturbation category and **Y** an ordinal index within that category. Examples of camera and LiDAR perturbations are shown in Figure 1.

2.1 Camera Perturbations

For camera inputs, we adopt the perturbation suite from PerturbationDrive [36, 40], including 41 image corruptions that emulate environmental effects, sensor artifacts, and visual distortions. Each perturbation is applied with five severity levels following the configuration of the original paper. More in detail, we evaluate the following perturbations.

Noise Perturbations (A). Noise perturbations simulate sensor noise and signal interference: Gaussian (C-A$_{\mathrm{I}}$), Poisson (C-A$_{\mathrm{II}}$), impulse (C-A$_{\mathrm{III}}$), JPEG compression (C-A$_{\mathrm{IV}}$), and speckle noise (C-A$_{\mathrm{V}}$) [5, 14, 21, 28, 35, 45, 46, 48, 55].

Blur Perturbations (B). Blur perturbations model optical and motion artifacts: defocus (C-B$_{\mathrm{I}}$), glass (C-B$_{\mathrm{II}}$), motion (C-B$_{\mathrm{III}}$), zoom (C-B$_{\mathrm{IV}}$), Gaussian (C-B$_{\mathrm{V}}$), low-pass (C-B$_{\mathrm{VI}}$), and high-pass filtering (C-B$_{\mathrm{VII}}$) [21, 28, 45, 46, 48, 55].

Weather and Illumination Perturbations (C). These perturbations emulate environmental and illumination effects: frost (C-C$_{\mathrm{I}}$), snow (C-C$_{\mathrm{II}}$), fog (C-C$_{\mathrm{III}}$), brightness (Increase: C-C$_{\mathrm{IV}}$, Decrease: C-C$_{\mathrm{V}}$), contrast (C-C$_{\mathrm{VI}}$), rain (C-C$_{\mathrm{VII}}$), raindrops (C-C$_{\mathrm{VIII}}$), lightning (C-C$_{\mathrm{IX}}$), smoke (C-C$_{\mathrm{X}}$), sun glare (C-C$_{\mathrm{XI}}$), and snowflakes (C-C$_{\mathrm{XII}}$) [5, 12, 28, 45, 46, 55].

Distortion Perturbations (D). Distortion perturbations modify pixel geometry: elastic deformation (C-D$_{\mathrm{I}}$), pixelation (C-D$_{\mathrm{II}}$), sample pairing (C-D$_{\mathrm{III}}$), and sharpening filters (C-D$_{\mathrm{IV}}$) [5, 12, 29].

Graphic Pattern Perturbations (E). Graphic pattern perturbations overlay synthetic structures: splatter (C-E$_{\mathrm{I}}$), dotted lines (C-E$_{\mathrm{II}}$), zigzag patterns (C-E$_{\mathrm{III}}$), Canny edges (C-E$_{\mathrm{IV}}$), cutout (C-E$_{\mathrm{V}}$), and object overlays (C-E$_{\mathrm{VI}}$).

Color and Tone Adjustments (F). Color and tone perturbations modify chromatic properties: false color (C-F$_{\mathrm{I}}$), phase scrambling (C-F$_{\mathrm{II}}$), histogram equalization (C-F$_{\mathrm{III}}$), white balance (C-F$_{\mathrm{IV}}$), grayscale (C-F$_{\mathrm{V}}$), saturation (C-F$_{\mathrm{VI}}$), and posterization (C-F$_{\mathrm{VII}}$) [12, 21, 29].

2.2 LiDAR Perturbations

For LiDAR inputs, we adopt a total of 31 perturbation models from 3D-Corruptions-AD [15] and MultiCorrupt [6], which provide physically grounded and data-driven corruptions for point clouds. These methods cover weather effects, sensor artifacts, noise processes, and motion-related distortions. Each perturbation is applied with three severity levels following their original implementation.

In addition to these methods, we introduce lightweight approximations for selected perturbation types (i.e., weather effects and dynamic artifacts). These approximations are designed to capture the same qualitative degradation patterns as their simulation-based counterparts, while being computationally cheaper for real-world execution.

Weather and Environment Perturbations (A). They are modeled using both simulation-based and approximate methods. Snow includes two physically grounded snowfall models from 3D-Corruptions-AD (L-A$_{\mathrm{I}}$ and L-A$_{\mathrm{II}}$) and a variant from MultiCorrupt (L-A$_{\mathrm{III}}$). These approaches are based on light scattering formulations for LiDAR sensing [24]. In addition, we include a lightweight scattering approximation (L-A$_{\mathrm{IV}}$) that probabilistically attenuates and removes points based on distance and intensity. Rain includes a simulation-based rainfall model from 3D-Corruptions-AD (L-A$_{\mathrm{V}}$) and a lightweight attenuation model (L-A$_{\mathrm{VI}}$) that reduces return intensity and sparsifies distant points. Fog includes a physically based model from 3D-Corruptions-AD (L-A$_{\mathrm{VII}}$), a variant from MultiCorrupt (L-A$_{\mathrm{VIII}}$), and a lightweight attenuation approximation (L-A$_{\mathrm{IX}}$) based on exponential decay of signal strength with distance. Strong sunlight (L-A$_{\mathrm{X}}$), adopted from 3D-Corruptions-AD, introduces background noise and spurious returns.

Sensor and Field-of-View Artifacts (B). Density decrease includes global point thinning from 3D-Corruptions-AD (L-B$_{\mathrm{I}}$) and a stochastic reduction variant from MultiCorrupt (L-B$_{\mathrm{II}}$). Beam reduction (L-B$_{\mathrm{III}}$) simulates reduced LiDAR channels and is adopted from MultiCorrupt. Cutout (L-B$_{\mathrm{IV}}$) removes regions from the point cloud and is adopted from 3D-Corruptions-AD. Crosstalk noise (L-B$_{\mathrm{V}}$) and field-of-view filtering (L-B$_{\mathrm{VI}}$) are also from 3D-Corruptions-AD.

Global Noise Perturbations (C). They include Gaussian noise (L-C$_{\mathrm{I}}$), uniform noise (L-C$_{\mathrm{II}}$), and impulse noise (L-C$_{\mathrm{III}}$), all from 3D-Corruptions-AD.

Local Corruptions (D). They include density reduction (L-D$_{\mathrm{I}}$), cutout (L-D$_{\mathrm{II}}$), and localized Gaussian (L-D$_{\mathrm{III}}$), uniform (L-D$_{\mathrm{IV}}$), and impulse noise (L-D$_{\mathrm{V}}$), all from 3D-Corruptions-AD.

Affine Transformations (E). They include shear (L-E$_{\mathrm{I}}$), scaling (L-E$_{\mathrm{II}}$), and rotation (L-E$_{\mathrm{III}}$), from 3D-Corruptions-AD.

Motion and Alignment Perturbations (F). Motion blur (L-F$_{\mathrm{I}}$) introduces temporal smearing of points and is adopted from MultiCorrupt. Moving-object perturbations include dynamic noise injection from 3D-Corruptions-AD (L-F$_{\mathrm{II}}$) and ghost point generation implemented by us (L-F$_{\mathrm{III}}$). Spatial misalignment (L-F$_{\mathrm{IV}}$) applies rigid transformations and is from MultiCorrupt.

3 EVALUATION FRAMEWORK

Our evaluation framework includes a real autonomous driving vehicle, a Volkswagen Passat Variant GTE plug-in hybrid [3],[1] along with a software library that unifies camera and LiDAR perturbations

[1] Reference anonymized for double-blind review.

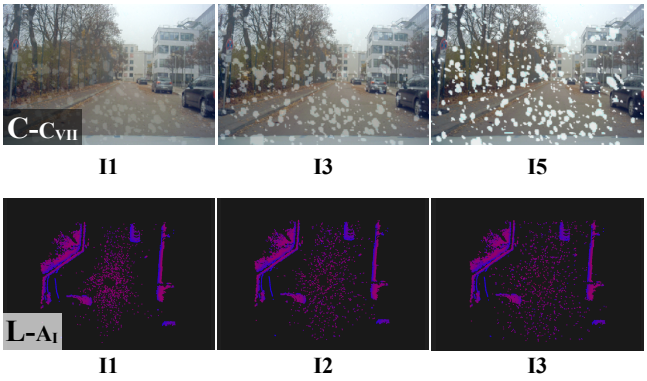

Figure 1: Examples of camera (top) and LiDAR (bottom) perturbations illustrating the Snow effect at different intensities.

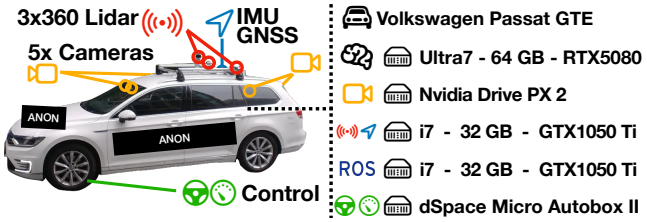

Figure 2: Autonomous driving platform used in this work: sensors (left) and onboard computing hardware (right).

from multiple sources (e.g., PerturbationDrive, 3D-Corruptions-AD, MultiCorrupt) under a unified execution interface, described next.

3.1 Vehicle Platform

As illustrated in Figure 2 (left), the vehicle is equipped with a sensor suite providing a 360° field of view. The setup includes five Sekonix automotive cameras (2.3 MP), consisting of two front-facing cameras (60° and 120°), two side-facing cameras (120°), and one rear camera (120°). The vehicle is also equipped with three Velodyne LiDAR sensors mounted on a roof rack, including a central 32-layer VLP-32C (range up to 200 m) and two tilted VLP-16 sensors covering the vehicle's sides. For localization, the platform uses an iMAR iNAT FSSG 1 GNSS INS system with RTK, providing positioning accuracy of approximately 2 cm, used both by modular driving pipelines and for recording high-precision telemetry during experiments.

The onboard computing infrastructure (Figure 2 right) includes two CAR PCs (Intel Core i7, 32 GB RAM, NVIDIA GTX 1050 Ti), an NVIDIA Drive PX2 used for perception processing, and a high-performance computing unit dedicated to onboard intelligence and perturbation execution, equipped with an Intel Ultra 9 processor, an NVIDIA RTX 5080 GPU, and 64 GB of memory. Low-level vehicle control is handled by a dSPACE MicroAutoBox [17], which interfaces with the vehicle CAN bus through a secure gateway to control steering and acceleration actuators in real time. All computing units are interconnected through a secure Ethernet local network.

3.2 Testing Modalities

3.2.1 Model-Level. Perturbations are applied offline to recorded sensor data, i.e., without ROS or vehicles integration. Camera perturbations modify image frames before inference, while LiDAR perturbations affect point clouds prior to perception processing, enabling batch execution and isolating input-level effects.

3.2.2 Hardware-in-the-Loop. Recorded ROS sensor messages are replayed on the vehicle hardware, and perturbations are applied online within the same ROS pipeline used in the live system. This enables evaluation of runtime performance under realistic conditions and filtering of perturbations that violate real-time constraints.

3.2.3 Vehicle-in-the-Loop. Perturbations are injected online while the systems under test run on the real vehicle. ROS nodes subscribe to raw sensor topics, apply perturbations in real time, and republish modified data to downstream modules. The same nodes used in HiL are reused on live sensor streams. Perturbations are injected seamlessly at the sensor interface via Ethernet, requiring no modifications to the ADS.

3.3 Perturbation Library

We extended PerturbationDrive [36] beyond its original image-based, simulation-driven setting by incorporating LiDAR perturbations and enabling execution across model-level, HiL, and real-world ViL environments.

Camera perturbations. Camera perturbations operate on image tensors. In offline execution, they are applied before inference; in ROS-based settings, a node subscribes to image topics, applies transformations, and republishes frames while preserving message structure and timestamps. For dynamic effects (e.g., rain, smoke), spatial masks and particle structures are pre-computed and reused to reduce runtime overhead.

LiDAR perturbations. In offline execution, they are applied directly to recorded data; in ROS-based settings, a node processes streaming point clouds, converts them to internal representations, applies perturbations, and republishes the results. To handle high data rates, the node adopts a decoupled producer–consumer design: incoming messages are buffered via non-blocking callbacks, while perturbations are applied in a separate processing loop using vectorized operations, preventing latency propagation. In particular, physics-based perturbations from MultiCorrupt are adapted to the Velodyne VLP-32C [10] by reconfiguring sensor-specific parameters (e.g., beam count, vertical field of view, angular resolution, range limits) and regenerating scattering lookup tables.

Preliminary experiments showed that full physics-based models incur prohibitive overhead for real-time execution. We therefore adopt phenomenological approximations of weather effects that preserve key properties (e.g., distance-dependent attenuation and near-field backscatter) while reducing computation to constant time per point. Atmospheric attenuation is modeled using a stochastic formulation derived from the Beer–Lambert law [59]. For a point at distance d, the survival probability is: $P_{\text{surv}}(d) = e^{-\alpha d}$, where α controls perturbation severity. Points are stochastically discarded according to P_{surv}, simulating distance-dependent signal loss observed in fog, rain, and snow. Synthetic back-scatter noise is injected in the near-field region to emulate particle reflections,

reproducing characteristic ghost returns. This formulation enables execution above 5Hz without per-ray simulation.

The LiDAR node supports both global and actor-specific perturbations. For actor-specific perturbations, the node first identifies points associated with dynamic objects using 3D bounding boxes generated in real time from the unperturbed stream, and then applies the corruption only to those points. This mode is used to enable *Local Corruptions (D)* and moving-object perturbations in *Motion and Alignment Perturbations (F)*. As defined in the original libraries, camera perturbations use five severity levels, whereas LiDAR perturbations provide either three or five; for consistency, we unify all LiDAR perturbations to a three-level scale.

4 EMPIRICAL STUDY

4.1 Research Questions

RQ$_1$ (model-level testing): *How do camera and LiDAR perturbations affect the outputs of ADS during model-level testing?*
RQ$_2$ (hardware-in-the-loop feasibility): *Can perturbation nodes meet the real-time constraints of the vehicle platform during hardware-in-the-loop testing?*
RQ$_3$ (system-level testing): *How do perturbations affect driving behavior when injected into live sensor streams during ViL testing?*
RQ$_4$ (perturbation-based fine-tuning): *Does training an end-to-end driving model on perturbed data improve robustness under real weather conditions?*

RQ$_1$ isolates the direct impact of input perturbations by applying them offline to recorded sensor data, enabling controlled, repeatable evaluation of model robustness independent of system-level dynamics. RQ$_2$ evaluates whether perturbations that are effective offline remain feasible under real-time constraints on automotive hardware, a prerequisite for hardware-in-the-loop and on-vehicle testing. RQ$_3$ assesses whether perturbation effects observed in controlled settings translate to real-world driving behavior. While model- and hardware-level evaluations capture functional and timing aspects, only closed-loop, on-vehicle execution reveals their impact on system-level performance under realistic dynamics and feedback. RQ$_4$ evaluates whether perturbations can be used not only for testing but also to improve robustness. By fine-tuning on perturbed data and assessing performance under real weather conditions, this RQ examines the transferability of synthetic perturbations to real-world robustness gains.

4.2 Driving Scenario

Experiments are conducted in **anonymized**, a public urban street (450 m, two-way, 30 km/h, Figure 3) with sidewalks, no lane markings, parked vehicles partially occupying the right driving lane, and three curves, but no intersections or pedestrian crossings. This yields a controlled yet non-trivial scenario, as the vehicle must infer the drivable corridor without lane markings while handling road curvature and lateral constraints induced by parked vehicles. We define three routes: Route A (full route) is used for training, dataset collection, and fine-tuning tests (RQ$_1$, RQ$_4$). Route B starts 70 m after Route A and includes two turns; it is used for camera-based experiments (RQ$_3$). Route C shares the start of Route A and ends at Route B's endpoint; it is used for LiDAR-based experiments (RQ$_3$).

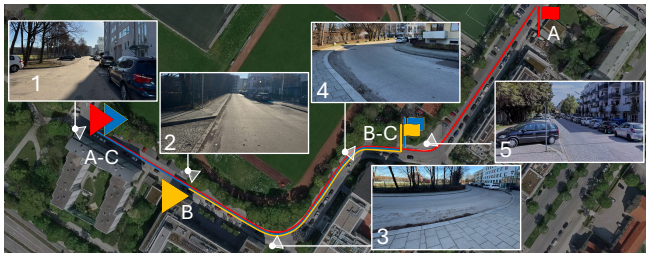

Figure 3: Real-world testing scenario.

4.3 Objects of Study

4.3.1 End-to-end Driving Model. The system under test is a vision-based driving model inspired by DAVE-2 [7] that predicts continuous steering commands from RGB images and comprises five convolutional layers followed by three fully connected layers, with Batch Normalization and Dropout to improve training stability and generalization. Input frames (800 × 503) are cropped by 204 pixels from the top and 35 from the bottom, yielding 800 × 264 images focused on the road surface. The training dataset includes 150 driving runs and 292k image-command pairs, and is augmented via horizontal flipping with inverted steering commands, following existing guidelines [7]. Data is split into 80% training and 20% validation with fixed seeds for reproducibility. The model is trained with Adam [33], a learning rate of 1e−4, batch size 16, and MSE loss, for up to 100 epochs with early stopping (patience 10).

4.3.2 Modular Perception Model. The system uses the LiDAR-based Autoware Mini pipeline [62], a Python port of the Autoware stack [19], deployed both on a Lexus RX450h at the University of Tartu [4] and adapted to our vehicle. Perception is based on geometric clustering: ground points are removed, and the point cloud is segmented using DBSCAN clustering [18], with clusters converted into 3D bounding boxes. The ADS includes perception, global and local planning, and control. Global planning uses a Lanelet2 map [52], while local planning adapts trajectories online and supports street-level navigation around parked vehicles. Trajectories are tracked using a Pure Pursuit controller.

4.4 Procedure and Metrics

4.4.1 RQ$_1$. We evaluate perturbation impact on the two ADS offline using a testing dataset. A separate recording on Route A was performed under nominal driving conditions. The recording consists of a 128-second cloudy-weather sequence with 3840 images and 640 point clouds. For each perturbation, perturbed versions of the dataset are generated at all intensity levels.

For camera inputs, perturbations are first applied before DAVE-2 inference, after which steering predictions are compared against the unperturbed baseline. We quantify their effect using Mean Squared Error (MSE), reporting the average and maximum across severity levels. MSE is a standard steering-angle regression metric and is also used in the original DAVE-2 implementation [7]. We additionally report the maximum steering deviation, i.e., the maximum absolute difference from the baseline prediction.

For LiDAR inputs, perturbations are first applied to point clouds before perception, after which detections are compared against the

unperturbed baseline. We quantify the effect using Retention Rate (Recall) of baseline detections [20, 32, 65], reporting the average and minimum across severity levels, and Average Translation Error (ATE) [9, 32], reporting its maximum across severity levels. Recall is the percentage of baseline detections that remain detectable after perturbation, while ATE measures translation error between matched bounding boxes in perturbed and baseline point clouds.

4.4.2 RQ_2. We integrate the perturbation library on the vehicle's onboard hardware within the ROS perception pipeline. Each perturbation is benchmarked over a 10-second window, recording perturbation latency and ROS node overhead for each frame or point cloud. We measure average latency (mean processing time), maximum latency (worst case across severity levels), and node overhead, i.e., the additional latency introduced by the ROS node.

This separation lets us assess whether middleware contributes to real-time violations. Latency is computed per severity level and aggregated per perturbation as the mean across severity levels and the maximum observed value. For ViL evaluation in RQ_3, camera perturbations must remain below 33,ms and LiDAR perturbations below 200,ms; others are excluded.

4.4.3 RQ_3. Perturbations are evaluated starting from the highest severity level (5). Upon failure, lower severities ([4..1]) are tested. This prioritizes failure-inducing conditions and reduces real-world testing cost. All experiments are conducted at a fixed speed of 20 km/h. All trajectory-based metrics are computed at the run level and aggregated per perturbation using only non-failing runs. In addition, for each perturbation, we summarize the run outcome separately for each tested intensity level.

Each experiment is classified as *success* or *failure*, with results summarized over five severity levels for the camera-based system and three representative levels for the LiDAR-based system. For all runs, we report completion rate, i.e., the percentage of the scenario completed before failure [38]. Failures are further categorized into five types. Erratic start (**E**) refers to unstable behavior immediately after initialization, characterized by large steering oscillations (> 0.3 between consecutive frames) that prevent starting experiments. Understeer (**U**) denotes insufficient steering during turns, causing the vehicle to drift outward. Oversteer (**O**) refers to excessive steering leading to sharp deviations from the intended trajectory. Out of road (**R**) describes lane departure during straight segments due to accumulated lateral error. Collision (**C**) corresponds to contact with static obstacles (e.g., parked vehicles), typically due to perception or planning failures.

We also compute cross-track error (CTE) [30, 57], defined as the per-frame lateral distance between the driven trajectory and a reference region constructed from three nominal trajectories. CTE is computed over the full trajectory for non-failing runs and up to the failure point for failing runs. In the latter case, the final d meters ($d = 5$m) before failure are excluded, and at least $d + 1$m of valid trajectory must remain after truncation, in addition to a minimum completion threshold. This ensures that CTE reflects stable driving behavior rather than terminal instability immediately before failure.

For the end-to-end model, control stability is evaluated using steering jitter, i.e., the mean absolute change between consecutive steering commands. This metric has been used to measure driving stability in end-to-end systems [13, 38], including PerturbationDrive [1]. For the LiDAR-based ADS, we measure near-field detection coverage, i.e., the average proportion of vehicle detections maintained within 5,m during closed-loop operation. This distance slightly exceeds our vehicle length (4.7,m), covering the immediate obstacle-avoidance region.

4.4.4 RQ_4. We fine-tune DAVE-2 on an augmented training set obtained by applying camera perturbations to the original images. For each sample, one perturbed variant is generated by randomly sampling a perturbation type and severity level from those used in RQ_1–RQ_3. The augmented data is combined with the original dataset, approximately doubling its size while retaining clean samples to preserve nominal performance. The model architecture, training procedure, and data splits remain unchanged to ensure a fair comparison.

The fine-tuned model is deployed on the vehicle and evaluated using the same closed-loop driving setup as in RQ_3. In contrast to RQ_3, which uses reduced routes, the evaluation in RQ_4 is conducted on the full driving scenario (Route A) to assess robustness under more diverse and extended driving conditions. First, both models are tested under nominal conditions to verify that the fine-tuned model maintains comparable performance in the absence of perturbations. Second, the models are evaluated under naturalistic environmental disturbances, including mud on the windshield, raindrops, sunlight glare, wet road surfaces, and heavy cloud coverage.

Each condition is evaluated in three runs on Route A. The evaluation follows the same metrics as in RQ_3, including run outcome, completion rate, and failure type. For successful runs, we report cross-track error and steering jitter as defined in RQ_3 to assess trajectory similarity and control stability.

4.4.5 Summary. Our experiments comprise more than 740 test executions: 298 at the model level, 298 in the HiL setup, 119 in the ViL setup, and 30 for fine-tuning. In total, the study covers more than a million ADS predictions and over 48 hours of real-world driving conducted across multiple months to capture diverse weather conditions, with collected ROS data exceeding 600 GB. Across model-level, HiL, and ViL experiments, the total runtime exceeds 80 hours.

4.5 Results

4.5.1 Model-level testing (RQ_1). The first four columns of Table 1 report model-level robustness metrics for camera and LiDAR perturbations. For the camera model, 32/42 perturbations show a clear monotonic increase in error with severity. The remaining cases follow a few recurring exception patterns: *C-C_{VI}*, *C-B_{IV}*, and *C-B_{VII}* remain nearly flat across severity, *C-E_{IV}* shows an inverted trend, and *C-E_{III}* and *C-C_{IX}* peak at intermediate severities. The remaining four, such as *C-F_I*, are broadly increasing but locally irregular. In terms of magnitude, only 8/42 camera perturbations exceed 1.0 in Avg. MSE, whereas 14/42 exceed 1.0 in Max. MSE, showing that peak degradation is more common than sustained average degradation. The strongest cases are *C-C_{XII}*, *C-D_{III}*, *C-F_I*, *and C-E_V*, all with high average error (Avg. MSE > 1.6) and high peak error (Max. MSE > 3.0). By contrast, most perturbations remain below 1.0 in Avg. MSE (34/42) and Max. MSE (28/42). A large gap between Avg. MSE

Table 1: Combined camera and LiDAR robustness, latency, and on-vehicle impact summary.

Camera	Trend	Avg. MSE	Max. MSE	Max. dev.	Avg.	Max.	Completion rate	Failure type	Avg CTE	Fail CTE	Jitter
C-C-XII		2.66	5.78	6.45	6.03	7.36		[✓,E,E,E,E]	0.35	–	6.06
C-D-III		2.08	4.52	6.02	0.63	1.08		[-,✓,E,E,E]	0.51	–	5.47
C-F-I		1.76	3.09	6.75	0.76	8.94		[-,✓,C,C,C]	0.62	0.61	5.93
C-E-V		1.62	3.22	5.68	0.16	0.35		[-,✓,U,E,E]	0.75	0.91	10.52
C-F-II		1.40	2.73	5.55	123.99	131.88		–	–	–	–
C-C-XI		1.27	2.27	6.40	6.73	8.72		[-,✓,U,U,U]	0.83	0.95	8.43
C-C-VII		1.26	1.64	5.43	39.77	138.36		–	–	–	–
C-C-I		1.11	2.61	5.28	15.08	22.64		[✓,O,C,C,C]	0.77	0.86	7.92
C-C-VIII		0.80	1.88	3.78	6.18	7.77		[-,-,-,-,✓]	1.11	–	3.32
C-E-IV		0.65	0.81	3.17	2.57	11.45		[-,-,-,-,✓]	0.72	–	5.25
C-F-III		0.65	1.30	2.96	1.39	10.28		[-,-,-,-,✓]	1.27	–	13.13
C-F-VII		0.65	2.56	5.25	2.46	3.84		[-,-,-,✓,O]	0.76	0.87	4.79
C-E-I		0.59	1.65	3.97	12.66	25.21		[-,-,✓,E,E]	0.57	–	11.60
C-C-V		0.56	1.49	5.23	1.95	3.14		[-,-,-,✓,U]	0.84	0.97	8.24
C-C-III		0.48	1.60	4.24	288.17	312.27		–	–	–	–
C-C-X		0.40	0.92	3.79	8.61	12.77		[-,-,-,✓,E]	0.66	–	6.72
C-C-IX		0.36	0.85	3.71	5.93	8.36		[-,-,-,-,✓]	0.60	–	10.07
C-A-IV		0.35	0.73	2.92	2.05	12.11		[-,✓,O,O,O]	0.78	0.80	5.55
C-E-VI		0.25	0.84	4.10	5.93	10.81		[-,✓,O,O,O]	0.55	0.55	8.53
C-B-IV		0.25	0.42	2.49	290.17	426.39		–	–	–	–
C-B-II		0.22	0.50	2.46	11.03	18.15		[-,✓,O,O,O]	1.29	0.73	7.13
C-C-II		0.21	0.56	2.42	13.47	21.39		[-,-,-,-,✓]	0.47	–	4.43
C-B-VI		0.19	0.30	1.77	2.10	12.18		[-,-,-,-,✓]	0.60	–	6.10
C-D-IV		0.17	0.40	2.62	1.29	2.07		[-,-,-,-,✓]	0.50	–	3.62
C-B-I		0.15	0.39	2.17	11.62	25.91		[-,-,-,-,✓]	0.71	–	6.49
C-F-V		0.12	0.38	2.25	0.56	1.04		[-,-,-,-,✓]	0.86	–	4.80
C-B-V		0.11	0.37	2.13	1.25	3.73		[-,-,-,-,✓]	1.04	–	4.37
C-F-VI		0.10	0.25	2.00	1.85	2.54		[-,-,-,-,✓]	0.74	–	3.85
C-A-III		0.10	0.19	2.02	1.10	2.62		[-,-,-,-,✓]	1.10	–	3.83
C-D-II		0.10	0.39	2.94	0.50	4.45		[-,-,-,✓,O]	0.79	0.72	3.47
C-B-VII		0.10	0.10	1.33	13.07	23.60		[-,-,-,✓,C]	0.83	0.80	5.45
C-A-I		0.10	0.25	2.29	16.25	26.21		[-,-,-,-,✓]	0.93	–	5.12
C-E-II		0.08	0.29	2.47	0.66	9.52		[-,-,-,-,✓]	0.70	–	3.76
C-B-III		0.07	0.23	1.82	3.78	18.48		[-,-,-,-,✓]	0.88	–	3.80
C-C-IV		0.05	0.14	2.23	1.95	3.14		[-,-,-,-,✓]	0.76	–	4.64
C-C-VI		0.05	0.13	1.43	4.06	6.27		[-,-,-,-,✓]	0.67	–	3.80
C-A-II		0.05	0.10	1.43	48.35	69.15		–	–	–	–
C-E-III		0.02	0.03	1.06	1.58	5.01		[-,-,-,-,✓]	0.83	–	2.74
C-A-V		0.02	0.06	1.57	15.67	19.72		[-,-,-,-,✓]	0.82	–	4.09
C-D-I		0.01	0.02	0.70	8.17	17.72		[-,-,-,-,✓]	0.85	–	4.45
C-F-IV		0.01	0.03	1.07	3.34	4.20		[-,-,-,-,✓]	1.02	–	3.43

LiDAR	Trend	Avg. Ret.	Min. Ret.	Max. ATE	Avg.	Max.	Completion	Failure type	Avg CTE	Fail CTE	Cov. 5m
L-B-VI		38.01	23.40	0.14	1.38	1.98		[✓,C,C]	0.94	1.07	0.95
L-A-II		34.62	32.82	0.72	22164.49	28611.20		–	–	–	–
L-A-III		42.94	39.32	0.58	35948.57	42401.67		–	–	–	–
L-A-IX		77.06	52.40	0.55	1.67	3.57		[C,C,C]	1.81	1.81	–
L-B-III		75.11	64.60	0.52	1.05	2.12		[✓,C,C]	1.38	1.61	0.97
L-B-II		73.92	64.74	0.44	1.36	2.66		[-,-,✓]	0.80	–	0.98
L-A-X		78.65	65.88	0.40	1.39	2.09		[✓,C,C]	1.92	2.38	0.97
L-A-VIII		87.68	73.49	0.22	439.68	576.27		–	–	–	–
L-F-III		88.61	76.62	0.47	5.32	8.11		[-,✓,C]	0.60	–	0.99
L-C-I		88.31	78.40	0.21	7.14	12.12		[✓,R,C]	2.07	2.63	0.97
L-F-IV		83.94	78.95	0.51	3.24	13.34		[C,C,U]	0.77	0.77	–
L-A-IV		89.42	80.77	0.42	1.60	3.18		[✓,U,C]	1.17	1.39	0.98
L-F-II		88.39	82.73	0.60	127.68	193.36		[✓,C,C]	1.67	2.26	0.97
L-A-VI		91.14	86.63	0.08	1.71	2.79		[U,C,C]	2.83	2.83	–
L-B-V		91.92	86.77	0.23	1.19	2.36		[✓,U,C]	1.05	1.33	0.96
L-A-VII		95.43	87.23	0.18	568.89	778.81		–	–	–	–
L-D-III		93.58	87.98	0.13	108.65	154.00		[C,C,C]	0.71	0.71	–
L-C-II		94.40	90.19	0.11	2.63	3.70		[-,✓,C]	2.77	4.85	0.94
L-B-IV		94.43	90.99	0.20	114.07	179.77		[-,-,✓]	0.44	–	0.98
L-E(x3)		95.93	93.20	0.14	170.00	287.15		–	–	–	–
L-A-I		95.01	93.70	0.12	873.95	1246.71		–	–	–	–
L-B-I		96.72	94.00	0.07	1.82	2.97		[✓,C,C]	0.65	0.81	1.00
L-D-IV		96.97	94.64	0.06	114.55	172.10		[C,C,C]	0.90	0.90	–
L-C-III		96.49	94.66	0.07	1.34	1.78		[C,U,C]	1.68	1.68	–
L-D-II		97.37	94.94	0.08	14.47	27.32		[-,-,✓]	0.77	–	0.99
L-A-V		95.77	95.48	0.11	850.58	1114.71		–	–	–	–
L-D-I		97.98	96.28	0.06	146.61	252.66		–	–	–	–
L-F-I		97.34	96.56	0.19	5.72	12.73		[-,-,✓]	0.48	–	1.00
L-D-V		98.15	97.35	0.04	135.04	256.83		–	–	–	–

and Max. MSE indicates degradation concentrated at one or a few higher severity levels rather than across all severities, as illustrated by *C-F*$_{VII}$ (Avg. 0.65, Max. 2.56), whose error rises sharply only at the highest severity. Avg. MSE does not capture the full offline effect. Nearly all perturbations (41/42), except *C-D*$_{I}$, exceed a Max. dev. of 1.046 (about one sixth of a steering-wheel turn), and 17/42 exceed 3.14 (half a steering-wheel turn). Thus, even perturbations with small average error can still induce large instantaneous steering commands, as shown by *C-F*$_{IV}$ (Max. MSE 0.03, Max. dev. 1.07).

For the LiDAR-based ADS, the trend bars report inverse retention, so increasing bars indicate lower retention at higher severity. Most perturbations follow this monotonic pattern.

The affine-transformation family (*L-E*) is reported as an aggregated row in Table 1, since its three variants show similar retention, ATE, and latency behaviour. Only *L-A*$_{II}$ and *L-A*$_{III}$ deviate, although both remain among the three most harmful perturbations by Avg. Ret. *L-B*$_{VI}$, by contrast, follows the monotonic trend and shows the strongest degradation, with Avg. Ret. 38.01% and Min. Ret. 23.40%.

Across LiDAR perturbations, Avg. Ret. and Min. Ret. are generally aligned, indicating similar average and worst-case degradation. However, retention impact is often small: nearly half of the perturbations (14/31) reduce retention by at most 10%, while only 11/31 reduce minimum retention below 80%. ATE is less aligned with retention: *L-B*$_{VI}$, with the lowest Min. Ret., ranks only 18th in maximum ATE, while *L-F*$_{II}$, the second-highest perturbation in ATE, ranks only 13th in retention. More broadly, six perturbations exceed 0.5 m in maximum ATE, whereas ten remain below 0.25 m. Thus, perturbations can affect retention and localization differently.

RQ$_1$ (model-level testing): Camera perturbations follow different severity-response patterns and can still trigger sharp steering reactions despite limited overall degradation. LiDAR perturbations mainly reduce detection performance as perturbation strength increases, while their effect on localization is less consistent.

4.5.2 Hardware-in-the-loop feasibility (RQ$_2$). The next two columns of Table 1 report average and maximum latency for camera and LiDAR perturbations. For camera-based perturbations, most satisfy the 33 ms real-time constraint: only 5/42 exceed it, and only 2/42 also exceed 200 ms, indicating clear infeasibility. Although two infeasible cases rank relatively high in effectiveness (5th and 7th), most of the strongest perturbations remain within real-time limits. Node overhead is negligible for all feasible perturbations, reaching at most 0.71 ms (*C-C*$_{V}$) and averaging below 0.36 ms.

For LiDAR perturbations, feasibility is more restrictive. While many remain within the 200 ms budget, 6/31 exceed it in average latency and 11/31 in maximum latency. Two of the three most effective perturbations are infeasible, but most infeasible cases are not top ranked, indicating that high latency is not generally associated with high impact. The affine-transformation family (*L-E*), reported as an aggregate in Table 1, exceeds the real-time limit overall and is not among the strongest LiDAR perturbations.

This contrast is clearest in the weather-related *L-A* family. For snow, *L-A*$_{II}$ (3D-Corruptions-AD) and *L-A*$_{III}$ (MultiCorrupt) are infeasible, with average latency reaching ~35 s, whereas *L-A*$_{I}$ and our lightweight approximation *L-A*$_{IV}$ remain within real-time constraints, ranking 22nd and 12th in effectiveness. A similar trend appears for rain and fog: the heavier variants (*L-A*$_{V}$, *L-A*$_{VII}$, *L-A*$_{VIII}$) are infeasible, whereas the lightweight approximations (*L-A*$_{VI}$, *L-A*$_{IX}$) remain within budget and are also more effective than their heavier counterparts. Node overhead for feasible LiDAR perturbations remains small, reaching at most 2.48 ms (*L-F*$_{IV}$) and averaging below 2 ms.

RQ$_2$ (hardware-in-the-loop feasibility): Most camera perturbations satisfy real-time constraints. For LiDAR, feasibility depends strongly on implementation: several existing weather perturbations are too slow for online use, while the lightweight variants remain within budget and, in some cases, are also more effective.

4.5.3 System-level testing (RQ$_3$). In the last five columns of Table 1, we report completion rate (bars), failure type, and CTE for camera and LiDAR perturbations, together with steering jitter (camera) and detection coverage within 5,m (LiDAR) for non-failing runs.

For camera perturbations, the most severe cases produce different failure profiles. *C-C*$_{XII}$ and *C-D*$_{III}$ mostly fail with *erratic start* (E), only succeeding once at intensities 1 and 2, respectively. *C-F*$_{I}$ instead fails with *collision* (C) after partial execution (13–57% completion), *C-E*$_{V}$ transitions from *erratic start* to *understeering* as intensity decreases, and *C-C*$_{XI}$ consistently fails with *understeering* after roughly half of the route. Severe perturbations can thus either prevent progress immediately or induce failure later through accumulating trajectory deviation. Failures are not limited to the strongest perturbations. *C-A*$_{IV}$, *C-E*$_{VI}$, and *C-B*$_{II}$ all fail with *oversteering* after similar completion rates, *C-D*$_{II}$ fails with *oversteering* after 72% completion, and *C-B*$_{VII}$ fails with *collision* after 96% completion despite low model-level error. Conversely, several perturbations with substantial model-level impact, including *C-C*$_{VIII}$, *C-E*$_{IV}$, and *C-F*$_{III}$, complete successfully at the highest intensity. *C-C*$_{X}$ further illustrates this mismatch, failing only at intensity 5 although its strongest model-level effect occurs at intensity 4.

Even when runs do not fail, CTE and jitter reveal degradation not visible from completion alone. *C-B*$_{II}$ shows the highest Avg. CTE (1.29) despite low model-level error, and small Avg. MSE can still coincide with noticeable trajectory deviation, as seen for *C-F*$_{IV}$ and *C-D*$_{I}$. The two metrics also distinguish degradation patterns: high CTE with low jitter indicates smooth but biased trajectories (e.g., *C-B*$_{II}$), whereas moderate CTE with high jitter indicates unstable steering despite successful completion (e.g., *C-F*$_{III}$, jitter 13.13). Comparing successful and failing runs further shows that some failures develop progressively, with higher pre-failure deviation (e.g., *C-E*$_{V}$: 0.75 vs 0.91), whereas others are more abrupt (*C-B*$_{VII}$: 0.83 vs 0.80). Overall, for camera perturbations, model-level effects provide only limited guidance for closed-loop behaviour.

For LiDAR perturbations, failures are more concentrated, with *collision* (C) as the dominant outcome. Among the highest-impact perturbations, *L-B*$_{VI}$, *L-B*$_{III}$, and *L-A*$_{IV}$ fail at intensities 2 and 1 after partial completion, while *L-A*$_{IX}$ fails at all tested intensities. Unlike the camera model, LiDAR perturbations can therfore still induce failure at the lowest tested intensity. However, strong detector-level degradation is neither necessary nor sufficient for system-level failure. Some perturbations with comparatively strong model-level effects, such as *L-B*$_{II}$, *L-B*$_{IV}$, and *L-D*$_{II}$, succeed at all intensities. Conversely, perturbations with modest model-level effects can still

fail early: *L-C*$_{II}$ fails at intensity 1 after only 10% completion, *L-C*$_{I}$ fails at intensities 2 and 1 after 13% and 24% completion, and *L-C*$_{III}$ fails despite near-nominal model-level predictions. Similar model-level trends can also lead to different outcomes: *L-B*$_{II}$ and *L-B*$_{III}$ show comparable trend shapes, yet only the latter fails, while *L-D*$_{II}$ succeeds at all intensities, whereas *L-D*$_{III}$ and *L-D*$_{IV}$ fail throughout.

Non-collision failures are less common but still recurring. *L-A*$_{VI}$ fails with *understeering* and then *collision*, *L-A*$_{IV}$ and *L-B*$_{V}$ show mixed *understeering*/*collision*, and *L-C*$_{I}$ is the only *road departure* case. Unlike in camera, *erratic start* does not occur: LiDAR runs begin successfully and deteriorate during execution. Coverage within 5 m remains high overall, with the lowest value at 0.94, indicating that vehicles are detected in most frames, but this metric does not capture temporal instability. During real runs, vehicles briefly disappeared and reappeared across consecutive frames, suggesting unstable inputs to the planner. This interpretation is supported by higher pre-failure CTE in several cases, most strongly for *L-C*$_{II}$ (2.77 to 4.85), and also for *L-A*$_{X}$, *L-C*$_{I}$, *L-B*$_{III}$, *L-B*$_{V}$, *L-B*$_{VI}$, and *L-B*$_{I}$. Under this view, even brief missed detections can be sufficient to cause *collision*, while repeated disappearance and reappearance of obstacles can also contribute to *understeering* and *road departure* by preventing the planner from converging to a stable trajectory.

RQ$_3$ (system-level testing): Camera and LiDAR perturbations both induce harmful closed-loop effects, but in different ways. For camera, failures vary in type and onset, and even successful runs can show substantial trajectory deviation or steering instability. For LiDAR, failures are more concentrated, predominantly as collisions, and can arise even when detector-level degradation appears small. Overall, model-level effects provide only limited guidance for on-vehicle behaviour.

4.5.4 Perturbation-based fine-tuning (RQ$_4$). Table 2 reports completion, failure type, and Avg. CTE and jitter for both the fine-tuned (FT) and baseline (Non-FT) models across different environmental conditions. For successful runs, these metrics are computed over the full trajectory; for failing runs, they are computed only over the completed portion before failure.

Fine-tuning consistently improves robustness. The FT model completes all runs across all tested conditions, whereas the Non-FT model frequently fails, including *collision* (C), *oversteering* (O), and *road departure* (R).

Under nominal conditions, both models achieve full completion. However, the FT model shows lower Avg. CTE (1.02 vs. 1.18), lower Avg. jitter (0.44 vs. 0.64), lower Max. jitter (24.91 vs. 26.75), and lower jitter variability (1.25 vs. 1.47), indicating smoother and more accurate trajectory tracking even without distribution shift.

Differences become more pronounced under adverse conditions. In Wetground, the Non-FT model achieves one full completion, but the other two runs terminate at 42% and 62% with *road departure* and *collision*, alongside a large increase in Max. CTE (3.09); the FT model completes all runs. In Mud and Raindrops, the Non-FT model never completes the route, with mixed *collision*/*oversteering* failures and completion rates between 43% and 60% in Mud and between 35% and 61% in Raindrops. In Sun, all Non-FT runs end in *collision*, with one run reaching 67% completion and the other two failing very early (14% and 13%). In all three conditions, the FT model again completes every run.

Table 2: RQ$_4$: Fine-tuning results under nominal and real weather conditions.

Weather	SUT	Completion	Fail. Types	CTE		Jitter		
				Avg	Max	Avg	Max	Std
Nominal	FT		[✓,✓,✓]	1.02	1.11	0.44	24.91	1.25
	Non-FT		[✓,✓,✓]	1.18	1.49	0.64	26.75	1.47
Wetground	FT		[✓,✓,✓]	1.21	1.29	0.29	12.26	0.83
	Non-FT		[R,C,✓]	1.84	3.09	0.45	13.94	1.17
Mud	FT		[✓,✓,✓]	1.34	1.49	0.38	19.40	1.03
	Non-FT		[O,O,C]	1.78	2.26	0.27	11.35	0.76
Sun	FT		[✓,✓,✓]	1.12	1.53	0.36	21.06	1.07
	Non-FT		[C,C,C]	2.08	2.58	0.51	18.55	1.29
Raindrops	FT		[✓,✓,✓]	1.47	1.61	0.36	19.58	1.03
	Non-FT		[C,O,O]	1.74	3.25	0.53	18.51	1.30

For perturbed conditions, comparison is restricted to the shared non-failing portion of each run. Even under this conservative comparison, the FT model maintains lower Avg. CTE in every condition and lower Avg. jitter in all cases except Mud. Although it often shows higher maximum jitter, its lower standard deviation in most perturbed conditions suggests that these peaks are isolated transient corrections rather than sustained unstable behavior.

RQ$_4$ (perturbation-based fine-tuning): Perturbation-based fine-tuning improves both robustness and stability. It removes failures across all tested conditions and reduces trajectory deviation and steering variability, indicating improved closed-loop behaviour.

4.6 Threats to Validity

Internal Validity. In a closed-loop system, attributing failures to a single component is not always straightforward. We address this by evaluating the same perturbations across model-level, hardware-in-the-loop, and on-vehicle setups, enabling us to observe how effects evolve from prediction changes to full system behavior. We also relate model-level metrics to driving outcomes such as run success, trajectory deviation, and control stability.

External Validity. A potential threat is that, for the end-to-end model, training and testing take place on the same route, and all experiments use a single vehicle. We mitigate this by separating training and testing in time and season (autumn vs. winter), and by evaluating on a public road rather than a closed track. Because parked cars and other scene elements change across days, including vehicles partially occupying the lane, the correct trajectory can differ substantially between training and testing, reducing the risk of overfitting to a fixed scenario. Although all experiments are conducted in a single scenario, the route captures a complex real-world urban scenario with non-trivial driving conditions. To limit confounding variability, RQ$_3$ evaluates camera perturbations on the same day in rapid succession, with LiDAR perturbations tested on a separate day, while RQ$_4$ alternates fine-tuned and non-fine-tuned variants. Finally, we study two popular ADS architectures, namely an end-to-end and a modular stack, and complement synthetic perturbations with real-world conditions [16, 27].

Reproducibility. Real-world experiments are hard to reproduce due to hardware and environmental variability. We mitigate this with recorded datasets, ROS bag replay for model-level and HiL experiments, deterministic training and evaluation, and shared perturbation implementations across execution levels.

5 QUALITATIVE ANALYSIS

Transfer analysis. To assess whether model-level effects (RQ_1) transfer to ViL behavior, we compare offline measurements with system-level outcomes for matched perturbation–intensity pairs that satisfy latency constraints and have valid measurements in both settings. We quantify transfer using (i) Spearman's ρ between offline and ViL measures, with 95% bootstrap confidence intervals and Benjamini–Hochberg-corrected significance, and (ii) top-k agreement, defined as overlap between the top five perturbation–intensity pairs in the offline and ViL rankings.

For the camera model, transfer is only moderate. Offline MSE is moderately associated with failure rate ($\rho = 0.56$, 95% CI [0.46, 0.64], $q < 10^{-4}$) and steering jitter ($\rho = 0.50$, 95% CI [0.25, 0.71], $q < 10^{-4}$). Offline maximum steering deviation shows similar associations with failure rate ($\rho = 0.55$, 95% CI [0.44, 0.64], $q < 10^{-4}$) and steering jitter ($\rho = 0.48$, 95% CI [0.23, 0.67], $q = 0.0001$). However, perturbation-level agreement is weak: top-five overlap is 0.00 for offline MSE versus ViL failure rate, and 0.20 for offline maximum deviation versus ViL steering jitter.

For LiDAR, transfer is weaker and less consistent. Offline retention loss is moderately associated with failure rate ($\rho = 0.40$, 95% CI [0.16, 0.61], $q = 0.0041$), but not with 5 m coverage ($\rho = -0.10$, 95% CI [-0.42, 0.21], $q = 0.5771$). Offline ATE shows only weak association with failure rate ($\rho = 0.19$, 95% CI [-0.06, 0.43], $q = 0.2156$) and average cross-track error ($\rho = 0.17$, 95% CI [-0.15, 0.44], $q = 0.3865$). Top-five overlap is likewise limited: 0.40 for retention loss versus failure rate, and 0.20 for ATE versus cross-track error.

Overall, offline model-level measurements provide limited guidance for ViL behavior: associations are at most moderate and do not reliably identify the most harmful perturbations in closed-loop execution. Transfer is somewhat stronger for the camera model than for LiDAR, but insufficient in both cases.

Failure hotspots. We also examine where failures occur during ViL execution. Figure 4 overlays camera and LiDAR closed-loop trajectories on the route, colors them by perturbation category, and marks failure type. For each failing run, we use the trajectory endpoint as the failure point and cluster these points in the vehicle map frame with DBSCAN, separately by failure type so that nearby but behaviorally distinct events, such as oversteer and understeer, are not merged. This yields failure-type-consistent hotspots; isolated failures remain as noise and are shown in Figure 4 as semi-transparent circles. For each cluster, we summarize the dominant failure type, dominant and contributing perturbation types.

Figure 4 shows five recurring camera hotspots, three concentrated in the central route segment. The first, at the route start, is dominated by erratic-start failures, mainly from category C, with additional contributions from D and E. Mid-route, failures split into adjacent but distinct modes: a large oversteer hotspot involving A, B, C, E, and F; an understeer hotspot driven mainly by C and secondarily by E; and a nearby collision hotspot caused by C and F, reflecting milder understeer cases where the vehicle begins to recover but still drifts into the parked cars after the turn. A final late hotspot near the route endpoint is again collision-dominated, mainly from B and C. Overall, category C is the most broadly disruptive camera category, spanning start failures, control loss, and collisions, whereas the remaining categories are more localized: B and F mainly contribute to collision-prone regions, E to the central control-loss area, and D almost entirely to the starting hotspot.

LiDAR failures are organized around one dominant early collision hotspot and a smaller set of secondary clusters. The main hotspot appears early in the route, is strongly collision-dominated, and is led by category A, although B, C, D, and F also contribute, indicating that multiple LiDAR perturbation categories collapse into the same severe early failure mode. A second collision cluster appears later in the route and is mainly associated with category B, with smaller collision clusters at intermediate completion as well. Unlike the camera case, LiDAR exhibits two understeer hotspots rather than a broad mix of control failures: one near the route end, dominated by category A with additional contribution from C, and one at mid completion involving A, B, and F. Overall, LiDAR failures are dominated by collision-prone regions, with category A as the most disruptive category across both early collisions and late understeer, while category B contributes to later collision hotspots.

6 DISCUSSION

Differences across sensors. Camera perturbations often cause abrupt steering changes, so even small average errors can produce unstable behavior. LiDAR perturbations instead tend to degrade performance more gradually, mainly by reducing detection quality. This difference reflects both sensing modality and system design: the end-to-end model maps inputs directly to control and is therefore sensitive to small changes, whereas the modular stack separates perception and control, leading to more predictable degradation, as also observed in prior work [2, 60]. This is also reflected in vehicle-in-the-loop failures: camera perturbations produce more diverse control-loss behaviors, whereas LiDAR perturbations are more concentrated in collision-prone regions. For developers, this suggests that robustness evaluation should account for system design: control stability is critical for vision-based systems, while detection quality is central for LiDAR-based systems.

Real-Time constraints matter. Not all perturbations proposed in the literature, especially for LiDAR, are practically applicable. Some are effective offline but exceed real-time constraints and cannot be used in closed-loop execution. However, many perturbations with strong behavioral impact remain feasible. This shows that real-world robustness testing must consider runtime constraints [5, 45].

Effectiveness of perturbation-based training. Training with perturbed data improves robustness to real-world disturbances such as glare, raindrops, and occlusions in our evaluation. These gains are visible not only in offline metrics but also in closed-loop driving. Importantly, they are observed even when the perturbations are simple and do not closely match real-world effects. Consistent with prior work in data augmentation, this suggests that exposure to diverse perturbations can improve generalization without requiring physically accurate modeling [12, 29]. Hence, even simple perturbations can yield meaningful robustness gains.

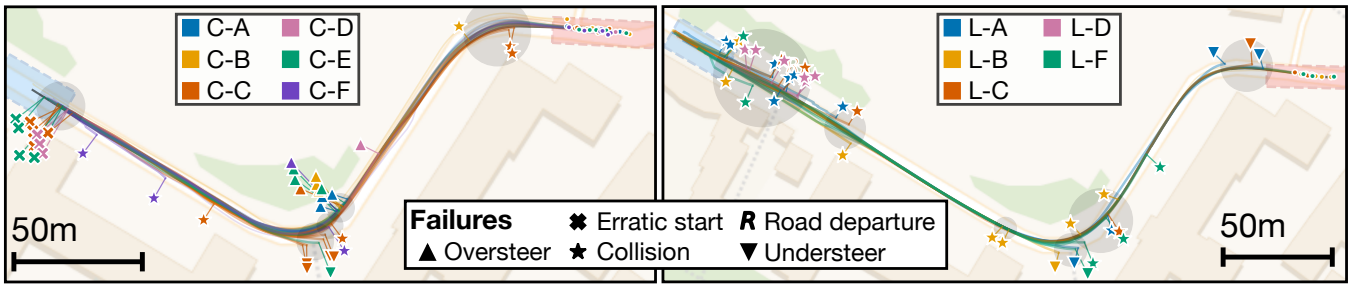

Figure 4: Testing trajectories for camera (left) and LiDAR (right).

Simulation vs. real-world testing. Offline datasets and simulation are widely used for testing, but both have limitations. Dataset-based evaluation is open-loop and therefore does not capture feedback effects [16, 28]. Simulation enables closed-loop testing, but relies on simplified sensing and system models, leading to a sim-to-real gap [61, 63, 68]. Our transfer analysis shows that offline measurements are at best moderately associated with vehicle-in-the-loop outcomes and do not reliably identify the most harmful perturbations. Our hotspot analysis further shows that failures cluster in specific route segments and modes, which become visible only in real-world closed-loop execution, where perception errors accumulate and interact with control over time. Overall, dataset- and simulation-based testing are necessary but not sufficient; real-world closed-loop evaluation is required to capture system-level effects that do not appear in simplified settings.

7 RELATED WORK

Model-level robustness benchmarking. Prior work has further analyzed the effect of corruption types on generalization, including noise, blur, and environmental degradations [14, 21, 46]. In autonomous driving, similar evaluations are conducted for both camera and LiDAR perception. Vision-based robustness is commonly studied using perturbation-based evaluation [2, 60], while LiDAR robustness is addressed through benchmarks such as 3D-Corruptions-AD [15, 16] and MultiCorrupt [6], which evaluate noise, density, weather, and geometric distortions in point clouds. These approaches operate offline and evaluate inputs without capturing temporal effects in closed-loop driving.

System-level and simulation-based testing. Several works extend perturbation-based testing to system-level evaluation in simulation. DeepXplore [51] and DeepTest [61, 61] apply input transformations to identify erroneous behaviours, while DeepRoad [68] generates transformed driving scenes using GAN-based methods. PerturbationDrive [36] introduces a comprehensive library of camera perturbations (e.g., noise, blur, weather, geometric distortions, color transformations) and evaluates them in simulation-based closed-loop driving, also exploring perturbation-based retraining.

Other approaches focus on online perturbations or runtime analysis. Adversarial methods [42, 66] generate perturbations during execution to induce failures, while DeepManeuver [63] applies state-aware perturbations in closed-loop simulation. Runtime monitoring approaches such as Luan et al. [45] and MarMot [5] assess prediction consistency by comparing outputs under controlled input variations. These works capture temporal effects but are typically limited to simulation and often focus on a single sensing modality.

Real-world robustness evaluation. Real-world failures of autonomous driving systems highlight the importance of robustness under operational conditions [8, 23, 31, 39, 54, 56]. Environmental factors such as adverse weather and sensor noise can significantly degrade perception performance in practice [16, 27]. Existing perturbation-based studies are primarily conducted on offline datasets or simulation [5, 36, 42, 45, 51, 61, 63, 66, 68], while real-world analyses rely on observed failures rather than controlled perturbations [8, 23, 31, 39, 54, 56].

There is limited work evaluating perturbations in real-world closed-loop driving, particularly across multiple sensing modalities. In this work, we evaluate perturbations across both camera and LiDAR modalities using a unified pipeline spanning dataset-level analysis, hardware-in-the-loop testing, and real-world closed-loop driving, enabling consistent evaluation across different levels of system integration.

8 CONCLUSIONS AND FUTURE WORK

This paper systematically evaluates perturbation-based robustness testing for autonomous driving systems using 72 camera and LiDAR perturbations at the model level, in hardware-in-the-loop, and in vehicle-in-the-loop experiments on a full-scale autonomous vehicle. Across over one million ADS predictions and more than 48 hours of driving on public roads in real-world urban conditions, we study two different ADS stacks: an end-to-end vision-based driving model and a modular LiDAR-based perception, planning, and control stack.

We find that perturbations affect ADS differently across testing levels. Some perturbations with low model-level effects still lead to unstable control, lane departures, or collisions at the system level, while others with strong model-level degradation have only a limited impact in vehicle-in-the-loop driving. This shows that offline robustness does not directly translate to system behavior in practice. These findings underline the need for robustness evaluation across multiple testing levels rather than relying on offline results alone. We further show that real-world perturbation testing is feasible under runtime constraints, and that perturbation-based training improves robustness to naturalistic perturbations while also improving generalization under nominal conditions. Overall, this highlights the practical value of perturbation-based methods for evaluating and improving ADS robustness.

In future work, we will extend the study to more routes, environments, and ADS stacks and explore additional testing and robustness improvement strategies for real-world deployment

9 DATA AVAILABILITY STATEMENT

We provide a replication package with our framework, both ADS implementations, all evaluation scripts, processed datasets, and Docker containers for deployment on a ROS-enabled vehicle [53]. A public DOI is not released at submission time to avoid premature dissemination; the artifact is available anonymously for review and will be archived with a DOI upon acceptance. Raw datasets cannot be fully shared due to restrictions from public street recordings. To ensure transparency and reproducibility, we include code to process the raw data and reproduce all reported results.

REFERENCES

[1] 2023. PerturbationDrive. https://github.com/HannesLeonhard/PerturbationDrive.

[2] Nourdine Aliane. 2025. A Survey of Open-Source Autonomous Driving Systems and Their Impact on Research. *Information* 16, 4 (2025). https://www.mdpi.com/2078-2489/16/4/317

[3] Anonymous authors. 2019. Anonymized work.

[4] Autonomous Driving Lab, University of Tartu. 2023. Vehicle. https://adl.cs.ut.ee/lab/vehicle. Accessed: 2026-03-06.

[5] Jon Ayerdi, Asier Iriarte, Pablo Valle, Ibai Roman, Miren Illarramendi, and Aitor Arrieta. 2024. MarMot: Metamorphic Runtime Monitoring of Autonomous Driving Systems. *ACM Trans. Softw. Eng. Methodol.* 34, 1, Article 18 (Dec. 2024), 35 pages. https://doi.org/10.1145/3678171

[6] Till Beemelmanns, Quan Zhang, Christian Geller, and Lutz Eckstein. 2024. Multi-Corrupt: A Multi-Modal Robustness Dataset and Benchmark of LiDAR-Camera Fusion for 3D Object Detection. In *2024 IEEE Intelligent Vehicles Symposium (IV)*. 3255–3261. https://doi.org/10.1109/IV55156.2024.10588664

[7] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. arXiv:1604.07316 [cs.CV]

[8] Neal E. Boudette. 2017. Tesla's Self-Driving System Cleared in Deadly Crash. https://www.nytimes.com/2017/01/19/business/tesla-model-s-autopilot-fatal-crash.html.

[9] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. 2020. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11621–11631.

[10] ROS Community. 2021. velodyne - ROS Wiki. https://wiki.ros.org/velodyne. Accessed: 2025-05-22.

[11] Victor Crespo-Rodriguez, Neelofar, and Aldeida Aleti. 2024. PAFOT: A Position-Based Approach for Finding Optimal Tests of Autonomous Vehicles. In *Proceedings of the 5th ACM/IEEE International Conference on Automation of Software Test (AST 2024)* (Lisbon, Portugal) *(AST '24)*. ACM, 159–170. https://doi.org/10.1145/3644032.3644457

[12] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. 2020. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc.

[13] Penglin Dai, Kai Liu, Qingfeng Zhuge, Edwin H.-M. Sha, Victor Chung Sing Lee, and Sang Hyuk Son. 2016. Quality-of-Experience-Oriented Autonomous Intersection Control in Vehicular Networks. *IEEE Transactions on Intelligent Transportation Systems* 17, 7 (2016), 1956–1967. https://doi.org/10.1109/TITS.2016.2514271

[14] Samuel Dodge and Lina Karam. 2016. Understanding how image quality affects deep neural networks. In *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6. https://doi.org/10.1109/QoMEX.2016.7498955

[15] Yinpeng Dong, Caixin Kang, Jinlai Zhang, Zijian Zhu, Yikai Wang, Xiao Yang, Hang Su, Xingxing Wei, and Jun Zhu. 2023. Benchmarking Robustness of 3D Object Detection to Common Corruptions in Autonomous Driving. arXiv:2303.11040 [cs.CV]

[16] Yinpeng Dong, Caixin Kang, Jinlai Zhang, Zijian Zhu, Yikai Wang, Xiao Yang, Hang Su, Xingxing Wei, and Jun Zhu. 2023. Benchmarking Robustness of 3D Object Detection to Common Corruptions in Autonomous Driving. arXiv:2303.11040 [cs.CV] https://arxiv.org/abs/2303.11040 nuscenes-C + perturbations.

[17] dSPACE dSPACE. 2026. dSPACE Microautobox. https://www.dspace.com/de/gmb/home.cfm

[18] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (Portland, Oregon) *(KDD'96)*. AAAI Press, 226–231.

[19] Autoware Foundation. 2022–2025. Autoware Core/Universe. https://github.com/autowarefoundation/autoware.

[20] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2013. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)* (2013).

[21] Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schütt, Matthias Bethge, and Felix A. Wichmann. 2018. Generalisation in humans and deep neural networks. In *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc.

[22] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. 2020. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics* 37, 3 (2020), 362–386.

[23] David Grossman. 2018. Uber Self-Driving Car Kills Pedestrian in Arizona. https://www.popularmechanics.com/technology/infrastructure/a19482100/uber-self-driving-car-kills-pedestrian-in-arizona/.

[24] Martin Hahner, Christos Sakaridis, Mario Bijelic, Felix Heide, Fisher Yu, Dengxin Dai, and Luc Van Gool. 2022. LiDAR Snowfall Simulation for Robust 3D Object Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[25] Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel Briand. 2020. Comparing Offline and Online Testing of Deep Neural Networks: An Autonomous Car Case Study. In *Proceedings of 13th IEEE International Conference on Software Testing, Verification and Validation (ICST '20)*. IEEE.

[26] Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel Briand. 2021. Can offline testing of deep neural networks replace their online testing? a case study of automated driving systems. *Empirical Software Engineering* 26, 5 (2021), 90.

[27] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. arXiv:1903.12261 [cs.LG] https://arxiv.org/abs/1903.12261

[28] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations (ICLR)*.

[29] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. 2020. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. *Proceedings of the International Conference on Learning Representations (ICLR)* (2020).

[30] Gabriel M. Hoffmann, Claire J. Tomlin, Michael Montemerlo, and Sebastian Thrun. 2007. Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing. In *2007 American Control Conference*. 2296–2301. https://doi.org/10.1109/ACC.2007.4282788

[31] Lily Jamali. 2025. *Tesla found partly to blame for fatal Autopilot crash.* BBC News. https://www.bbc.com/news/articles/c93dqpkwx4xo Accessed: 2026-02-03.

[32] Chiranjeevi Karri, José Machado da Silva, and Miguel Velhote Correia. 2023. Key Indicators to Assess the Performance of LiDAR-Based Perception Algorithms: A Literature Review. *IEEE Access* 11 (2023), 109142–109168. https://doi.org/10.1109/ACCESS.2023.3321912

[33] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

[34] Philip Koopman and Michael Wagner. 2017. Autonomous Vehicle Safety: An Interdisciplinary Challenge. *IEEE Intelligent Transportation Systems Magazine* 9, 1 (2017), 90–96. https://doi.org/10.1109/MITS.2016.2583491

[35] Jan Laermann, Wojciech Samek, and Nils Strodthoff. 2019. Achieving Generalizable Robustness of Deep Neural Networks by Stability Training. In *Pattern Recognition*. Springer, 360–373.

[36] Stefano Carlo Lambertenghi, Hannes Leonhard, and Andrea Stocco. 2025. Benchmarking Image Perturbations for Testing Automated Driving Assistance Systems. In *Proceedings of the 18th IEEE International Conference on Software Testing, Verification and Validation (ICST '25)*. IEEE, 12 pages.

[37] Stefano Carlo Lambertenghi and Andrea Stocco. 2024. Assessing Quality Metrics for Neural Reality Gap Input Mitigation in Autonomous Driving Testing. In *Proceedings of 17th IEEE International Conference on Software Testing, Verification and Validation (ICST '24)*. IEEE, 12 pages.

[38] Stefano Carlo Lambertenghi, Mirena Flores Valdez, and Andrea Stocco. 2025. A Multi-Modality Evaluation of the Reality Gap in Autonomous Driving Systems. In *40th IEEE/ACM International Conference on Automated Software Engineering, ASE 2025, Seoul, Korea, Republic of, November 16-20, 2025*. IEEE, 2808–2820. https://doi.org/10.1109/ASE63991.2025.00230

[39] Damon Lavrinc. 2018. This Is How Bad Self-Driving Cars Suck In The Rain. https://jalopnik.com/this-is-how-bad-self-driving-cars-suck-in-the-rain-1666268433.

[40] Hannes Leonhard, Stefano Carlo Lambertenghi, and Andrea Stocco. 2026. PerturbationDrive: A Framework for Perturbation-Based Testing of ADAS.

arXiv:2603.23661 [cs.SE] https://arxiv.org/abs/2603.23661

[41] Yunge Li and Lanyu Xu. 2024. Panoptic Perception for Autonomous Driving: A Survey. arXiv:2408.15388 [cs.RO] https://arxiv.org/abs/2408.15388

[42] Dongjie Liu, Jin Zhao, Axin Xi, Xinnian Huang Chao Wang, Kuncheng Lai, and Chang Liu. 2020. Data Augmentation Technology Driven By Image Style Transfer in Self-Driving Car Based on End-to-End Learning. *Computer Modeling in Engineering & Sciences* 122, 2 (2020), 593–617.

[43] Chengjie Lu, Shaukat Ali, and Tao Yue. 2024. EpiTESTER: Testing Autonomous Vehicles with Epigenetic Algorithm and Attention Mechanism. *IEEE Transactions on Software Engineering* (2024), 1–19. https://doi.org/10.1109/TSE.2024.3449429

[44] Chengjie Lu, Tao Yue, Man Zhang, and Shaukat Ali. 2023. DeepQTest: Testing Autonomous Driving Systems with Reinforcement Learning and Real-world Weather Data. arXiv:2310.05170 [cs.SE] https://arxiv.org/abs/2310.05170

[45] Siyu Luan, Zonghua Gu, and Shaohua Wan. 2023. Efficient Performance Prediction of End-to-End Autonomous Driving Under Continuous Distribution Shifts Based on Anomaly Detection. *Journal of Signal Processing Systems* 95, 12 (12 2023), 1455–1468. https://doi.org/10.1007/s11265-023-01893-5

[46] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S. Ecker, Matthias Bethge, and Wieland Brendel. 2020. Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming. arXiv:1907.07484 [cs.CV] https://arxiv.org/abs/1907.07484

[47] Agnieszka Mikołajczyk and Michał Grochowski. 2018. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*. 117–122. https://doi.org/10.1109/IIPHDW.2018.8388338

[48] Norman Mu and Justin Gilmer. 2019. MNIST-C: A Robustness Benchmark for Computer Vision. *CoRR* abs/1906.02337 (2019). arXiv:1906.02337 https://arxiv.org/abs/1906.02337

[49] Neelofar Neelofar and Aldeida Aleti. 2024. Identifying and Explaining Safety-critical Scenarios for Autonomous Vehicles via Key Features. *ACM Trans. Softw. Eng. Methodol.* 33, 4, Article 94 (April 2024), 32 pages. https://doi.org/10.1145/3640335

[50] Neelofar Neelofar and Aldeida Aleti. 2024. Towards Reliable AI: Adequacy Metrics for Ensuring the Quality of System-level Testing of Autonomous Vehicles. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE)* (Lisbon, Portugal). ACM, Article 68, 12 pages. https://doi.org/10.1145/3597503.3623314

[51] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. ACM, 1–18. https://doi.org/10.1145/3132747.3132785

[52] Fabian Poggenhans, Jan-Hendrik Pauls, Johannes Janosovits, Stefan Orf, Maximilian Naumann, Florian Kuhnt, and Matthias Mayr. 2018. Lanelet2: A High-Definition Map Framework for the Future of Automated Driving. In *Proc. IEEE Intell. Trans. Syst. Conf.* Hawaii, USA. http://www.mrt.kit.edu/z/publ/download/2018/Poggenhans2018Lanelet2.pdf

[53] Replication package 2026. Replication package. https://anonymous.4open.science/r/ASE2026-F55D/README.md.

[54] Reuters. 2026. *US opens probe after Waymo self-driving vehicle strikes child near school.* Reuters. https://www.reuters.com/world/us/us-opens-probe-after-waymo-self-driving-vehicle-strikes-child-near-school-2026-01-29/ Accessed: 2026-02-03.

[55] Evgenia Rusak, Lukas Schott, Roland S. Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. 2020. A Simple Way to Make Neural Networks Robust Against Diverse Image Corruptions. In *Computer Vision – ECCV 2020*. Springer, 53–69.

[56] Jonathan Stempel. 2025. *Tesla sued over Model S crash that killed three in New Jersey.* Reuters. https://www.reuters.com/legal/litigation/tesla-sued-over-new-jersey-crash-model-s-that-killed-three-2025-06-23/ Accessed: 2026-02-03.

[57] A. Stocco, B. Pulfer, and P. Tonella. 2023. Mind the Gap! A Study on the Transferability of Virtual Versus Physical-World Testing of Autonomous Driving Systems. *IEEE Transactions on Software Engineering* 49, 04 (apr 2023), 1928–1940. https://doi.org/10.1109/TSE.2022.3202311

[58] Andrea Stocco, Brian Pulfer, and Paolo Tonella. 2023. Model vs system level testing of autonomous driving systems: a replication and extension study. *Empirical Software Engineering* 28, 3 (May 2023), 73. https://doi.org/10.1007/s10664-023-10306-x

[59] DF Swinehart. 1962. The beer-lambert law. *Journal of chemical education* 39, 7 (1962), 333.

[60] Shuncheng Tang, Zhenya Zhang, Yi Zhang, Jixiang Zhou, Yan Guo, Shuang Liu, Shengjian Guo, Yan-Fu Li, Lei Ma, Yinxing Xue, and Yang Liu. 2023. A Survey on Automated Driving System Testing: Landscapes and Trends. *ACM Trans. Softw. Eng. Methodol.* 32, 5, Article 124 (July 2023), 62 pages. https://doi.org/10.1145/3579642

[61] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. ACM, 303–314. https://doi.org/10.1145/3180155.3180220

[62] Autonomous Driving Lab University of Tartu. 2026. Autoware Mini. https://github.com/UT-ADL/autoware_mini/tree/release/nodes/detection/lidar/cluster

[63] Meriel von Stein, David Shriver, and Sebastian Elbaum. 2023. DeepManeuver: Adversarial Test Generation for Trajectory Manipulation of Autonomous Vehicles. *IEEE Transactions on Software Engineering* 49, 10 (2023), 4496–4509. https://doi.org/10.1109/TSE.2023.3301443

[64] Shuai Wang and Zhendong Su. 2019. Metamorphic Testing for Object Detection Systems. *CoRR* abs/1912.12162 (2019). arXiv:1912.12162 http://arxiv.org/abs/1912.12162

[65] Shuai Wang and Zhendong Su. 2021. Metamorphic object insertion for testing object detection systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (Virtual Event, Australia) *(ASE '20)*. Association for Computing Machinery, New York, NY, USA, 1053–1065. https://doi.org/10.1145/3324884.3416584

[66] Hyung-Jin Yoon, Hamidreza Jafarnejadsani, and Petros Voulgaris. 2023. Learning When to Use Adaptive Adversarial Image Perturbations Against Autonomous Vehicles. *IEEE Robotics and Automation Letters* 8, 7 (2023), 4179–4186. https://doi.org/10.1109/LRA.2023.3280813

[67] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access* 8 (2020), 58443–58469.

[68] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE ASE (ASE '18)*. ACM, 132–142. https://doi.org/10.1145/3238147.3238187

[69] Xingyu Zhao, Javier Salido, Simos Gerasimou, and Radu Calinescu. 2025. On the Need for a Statistical Foundation in Scenario-Based Testing of Autonomous Vehicles. *arXiv preprint arXiv:2505.02274* (2025). https://arxiv.org/abs/2505.02274 Supported by UK EPSRC New Investigator Award.

[70] Husheng Zhou, Wei Li, Yuankun Zhu, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. 2018. DeepBillboard: Systematic Physical-World Testing of Autonomous Driving Systems. https://doi.org/10.48550/ARXIV.1812.10812