

Investigating Planning Attacks against End-to-End Autonomous Driving

Ruoyu Song
Purdue University
West Lafayette, IN, USA
song464@purdue.edu

Chenyi Wang
University of Arizona
Tucson, AZ, USA
chenyiw@arizona.edu

Raymond Muller
Lawrence Livermore National
Laboratory
Livermore, CA, USA
muller15@llnl.gov

Jean-Philippe Monteuiis
Qualcomm Technologies, Inc.
Boxborough, MA, USA
jmonteuu@qti.qualcomm.com

Ryan Gerdes
Virginia Tech
Blacksburg, VA, USA
rgerdes@vt.edu

Jonathan Petit
Qualcomm Technologies, Inc.
Boxborough, MA, USA
petit@qti.qualcomm.com

Ming F. Li
University of Arizona
Tucson, AZ, USA
lim@arizona.edu

Antonio Bianchi
Purdue University
West Lafayette, IN, USA
antonio@purdue.edu

Z. Berkay Celik
Purdue University
West Lafayette, IN, USA
zcelik@purdue.edu

Abstract

End-to-end autonomous driving systems (E2E ADSs) achieve state-of-the-art driving performance and increasingly replace conventional modular driving pipelines. These systems optimize multiple transformer modules (e.g., MapFormer and MotionFormer) jointly through a shared latent space to process sensor data and output planning trajectories. This shared architecture fundamentally changes the attack surface compared to conventional ADSs. However, it remains largely unexplored how manipulation of raw sensor inputs propagates through the latent space to induce unsafe trajectories, and whether decoded observable outputs provide reliable anomaly signals for attack detection.

In this paper, we introduce JANUS, an adversarial framework that abstracts recent E2E ADSs into a unified model, investigates how attacks can manipulate shared latent representations to induce unsafe planning, and examines whether decoded observable outputs can support consistency-based detection. To achieve these goals, we first formulate a joint objective function that generates input perturbations or patches to force planning failures. We then introduce constraints within this function to maintain the decoded outputs close to their clean states. Lastly, we adapt principles from existing consistency-based defenses for E2E ADSs to assess whether they can detect attacks generated with and without these constraints. We use the JANUS framework to evaluate three recent E2E ADSs (GenAD, VAD, and UniAD) on the nuScenes dataset. JANUS forces severe planning failures: it increases the baseline collision rate by a factor of 15 (up to 49.8%) and the off-road rate by a factor of 90 (up to 13.8%). Crucially, it achieves these failures while limiting the relative degradation in detection and map scores to below 20%,

substantially weakening the observable signals used by consistency-based defenses. Through this effort, we provide a framework for systematically analyzing shared-latent vulnerabilities in E2E ADSs and informing the design of more robust defenses.

CCS Concepts

• **Security and privacy** → **Systems security**; • **Computer systems organization** → **Embedded and cyber-physical systems**.

Keywords

end-to-end autonomous driving, adversarial attacks, autonomous driving security

ACM Reference Format:

Ruoyu Song, Chenyi Wang, Raymond Muller, Jean-Philippe Monteuiis, Ryan Gerdes, Jonathan Petit, Ming F. Li, Antonio Bianchi, and Z. Berkay Celik. 2026. Investigating Planning Attacks against End-to-End Autonomous Driving. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26)*, November 15–19, 2026, The Hague, Netherlands. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3830454.3846795>

1 Introduction

Traditional Autonomous Driving Systems (ADS) decompose vehicle operation into sequential sub-modules, specifically perception, localization, and planning, as illustrated in Figure 1 (top) [1, 25, 35]. These modules communicate via observable interfaces, such as bounding boxes and occupancy grids. For instance, the perception unit transmits a list of detected objects to the prediction module. This modular structure, however, could result in information loss and error propagation between components [36]. For example, when the perception module misses or mislocalizes a nearby vehicle due to sensor noise or occlusion, the prediction module will extrapolate trajectories from incomplete observations, and the planner may then commit to an unsafe maneuver (e.g., a lane change).



This work is licensed under a Creative Commons Attribution 4.0 International License. *CCS '26, The Hague, Netherlands*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2871-6/2026/11

<https://doi.org/10.1145/3830454.3846795>

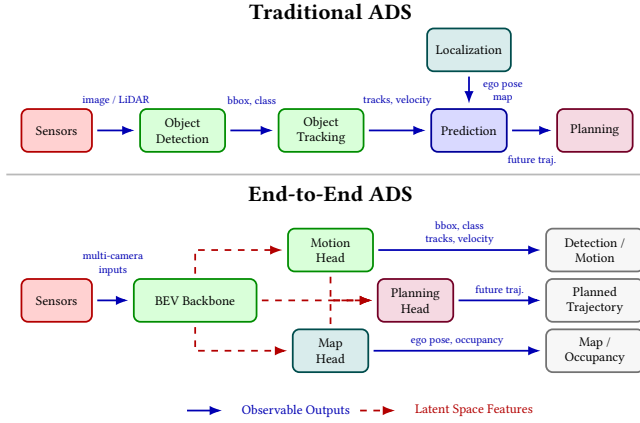


Figure 1: (Top) Traditional Modular ADS: Modules are isolated and communicate via explicit, human-readable interfaces (e.g., bounding boxes, trajectory waypoints). (Bottom) End-to-End ADS: Perception and planning are integrated into a unified framework. Information propagates via high-dimensional latent space features (red dashed arrows).

Recently, both academia (e.g., VAD [9] and UniAD [6]) and industry (e.g., Waymo [29] and Tesla [24]) have shown a growing interest in End-to-End (E2E) ADS. Figure 1 (bottom) abstracts the E2E ADS pipeline: unlike traditional modular stacks, E2E ADSs do not route planning through decoded observable representations (e.g., bounding boxes, lane boundaries). Instead, they encode raw sensor inputs into a shared latent state and task-level states. The planning head then uses these states to predict the final trajectory.

A key component of E2E ADSs is the Bird’s Eye View (BEV) encoder, which aggregates multi-view camera inputs into a shared latent representation [12, 32, 39]. Built on this BEV space, models often include task heads such as *MapTR* [13] (map prediction) and *AgentFormer* [34] (motion estimation), alongside a planning head that outputs the final trajectory. Sharing BEV features across these heads enables joint training rather than isolated optimization.

Recent E2E ADSs, such as UniAD [6] and VAD [9], share a BEV latent-space backbone but differ in how they parameterize and couple mapping, motion forecasting, and planning (e.g., via task-specific heads and training objectives) while still producing the final trajectory from latent features.

Extensive research exists on adversarial attacks against traditional modular ADS pipelines [2, 3, 15, 26, 36]. In these systems, attacks exploit the observable representation between modules, such as object detection/tracking [4, 18] or localization [23]. These attacks aim to change object labels, displace bounding box positions, or add/remove objects, ultimately causing the planner to crash. For example, an adversarial patch on a vehicle causes the perception module to output “no front vehicles”. This error propagates downstream: because the planner relies directly on the perception output, it maintains its speed despite the obstacle.

E2E ADSs fundamentally change this attack surface. This is because the planner does not necessarily consume decoded observable outputs as explicit inputs (although these systems may still decode

observable outputs from latent features, e.g., bounding boxes and map elements). Instead, the final trajectory is produced from shared latent features shaped by multi-view fusion, temporal reasoning, and jointly trained task heads. Thus, corrupting an observable representation is no longer sufficient to control the planned trajectory. Attack strategies designed for modular pipelines therefore do not directly transfer to E2E ADSs [7].

Existing research on adversarial attacks against E2E ADSs follows two primary directions. One line investigates module-wise perturbations through noise injection into intermediate representations to identify vulnerabilities in the interaction among E2E components [28, 37]. Another line evaluates E2E robustness under sensor input perturbations to determine their impact on planning results [10]. These studies demonstrate that E2E ADSs are vulnerable to both latent space and input-level perturbations.

These approaches, however, possess two fundamental flaws that prevent their assessment in practical deployments. Module-wise studies are useful for the evaluation of interactions in latent space among internal models; however, they require privileged access to latent representations. This assumption makes the threat model unrealistic for sensor-level adversaries, because a direct transformation from physical sensor inputs to the target latent space may not exist. Input-level robustness studies are more realistic; however, they optimize perturbations to cause trajectory degradation without considering how they change the observable outputs (e.g., the geometry of lane markings from *MapTR* and the object movement behavior from *AgentFormer*).

Common to both approaches, an attack that ignores unified observable model outputs enables existing consistency-based monitors [17, 19, 31, 33] to identify these modifications and detect planning attacks. These monitors use consistency between observable outputs to flag inconsistencies. If an attack corrupts both the planned trajectory and observable outputs, the monitor flags a consistency violation (e.g., object motion does not follow object label). In contrast, if observable outputs remain close to clean inference, monitors lose their primary signal, and the planning output can pass undetected. This leads to our central question:

How can a sensor-level adversary induce safety-critical planning failures while keeping the decoded observable outputs close to their clean states?

In this paper, we introduce JANUS, an adversarial framework for analyzing planning failures in E2E ADSs. JANUS generates sensor-level perturbations or patches that drive the planning head toward safety-critical outcomes, such as collisions and off-road trajectories. At the same time, it constrains decoded observable outputs, such as object detections and map predictions, to remain close to their clean states. This design allows us to test whether the shared latent representation can be manipulated to produce unsafe plans without comparable degradation in the observable outputs.

JANUS constrains the attack to a clean reference and optimizes a unified objective that (i) pushes the planner toward unsafe trajectories (e.g., collision/off-road) while (ii) penalizes drift in observable outputs relative to the clean inference. We then solve this objective under input-space constraints, including digital ℓ_p bounds and localized patches. For patch attacks, we use sensitivity-based placement to select regions that most strongly influence the planning output.

We evaluate JANUS on three recent E2E ADSs, GenAD, VAD, and UniAD, using the nuScenes dataset. JANUS substantially increases safety-critical failures: the collision rate rises from 3.34% to 49.81% on GenAD, from 8.50% to 30.68% on VAD, and from 5.1% to 17.1% on UniAD. The off-road rate also increases, reaching 13.8%, 10.9%, and 8.7% on the three models, respectively. Despite these failures, JANUS keeps decoded observable outputs close to clean inference. While the trajectory-deviation baseline degrades detection and map quality by up to 42.9% and 42.0%, respectively, JANUS limits the worst observed degradation to 17.5% and 16.6%. Our results expose a safety-critical failure mode: current shared-latent E2E ADSs can output unsafe trajectories even when their observable outputs remain largely intact.

In summary, we make the following contributions:

- We formalize an observable-output-preserving threat model for multi-task E2E ADSs. This model separates planning-specific failures from a broad corruption of the shared representation because it forces unsafe trajectories while the observable outputs remain close to the clean inference.
- We design and develop JANUS, an adversarial framework that realizes this threat model under input-space constraints. It combines clean-reference anchoring, route-conditioned safety objectives, and preservation constraints for bounded digital perturbations and localized patches.
- We evaluate JANUS on GenAD, VAD, and UniAD with the nuScenes dataset. JANUS substantially increases collision and off-road rates while the observable output degradation remains below 20%. This demonstrates that current shared-latent E2E ADSs can output unsafe trajectories even when decoded observable outputs remain largely intact.

2 Background and Related Work

End-to-end Autonomous Driving Systems. End-to-end ADSs jointly learn perception and planning from raw sensor inputs within a unified architecture. Rather than relying on explicit intermediate interfaces between modular components, these systems build a shared latent scene representation that supports multiple downstream tasks [6, 9, 14, 30, 38]. In many recent designs, both the planning output and observable outputs, such as object detections or map predictions, are derived from this shared representation. UniAD [6] pioneered this planning-oriented paradigm by connecting full-stack tasks through a shared query interface, ensuring that upstream perceptual features are directly aligned with downstream planning objectives. To address the computational overhead of such dense unified models, VAD [9] subsequently introduced vectorized scene representations, replacing expensive rasterized maps with sparse vector queries to significantly improve inference efficiency while maintaining planning safety.

More recently, the field has evolved from deterministic regression toward generative modeling to better handle the inherent uncertainty of dynamic driving environments. While predecessors such as UniAD and VAD typically regress a single trajectory, GenAD [38] reformulates planning as a generative task using Variational Autoencoders (VAEs [11]). By learning a structural latent space of potential futures, these frameworks can sample diverse and plausible trajectories instead of averaging conflicting possibilities. This

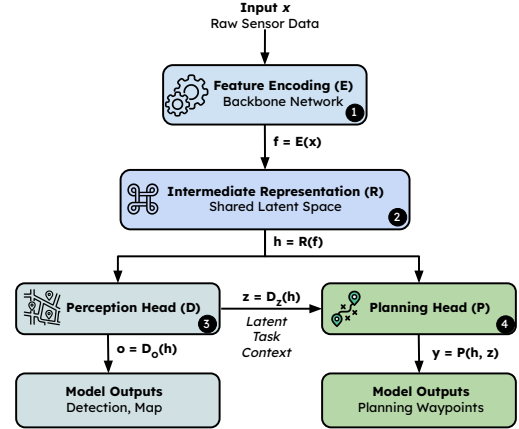


Figure 2: Abstracted E2E model: Sensor input x is encoded into a shared latent state h , which produces decoded observable outputs o and a latent task context z for planning. JANUS targets the planned trajectory y while preserving o .

marks a shift toward systems with unified representations and stronger probabilistic reasoning.

Adversarial Attacks Against E2E ADSs. E2E systems fuse multi-view, temporal sensor data into a shared latent scene representation that supports both planning and auxiliary perception outputs. Because the planner does not depend on a single explicit interface, attacks that corrupt standalone perception outputs do not directly translate to control failures in E2E ADSs.

Recent work has begun to study adversarial robustness in E2E ADSs. One line of work injects perturbations into internal module interfaces or intermediate representations to stress-test the interaction among E2E components [28, 37]. Another line benchmarks E2E ADS robustness under sensor-input perturbations and natural corruptions, and reports the resulting degradation in planning and perception outputs [10]. These studies show that E2E ADSs are vulnerable under perturbations, and motivate our focus on how planning failures relate to changes in decoded observable outputs.

Positioning of JANUS. In contrast to module-wise perturbation studies, JANUS keeps the attack at the sensor-input level. The adversary may use model parameters and gradients during offline optimization, but the realized attack modifies only camera inputs. It cannot overwrite BEV features, query embeddings, module outputs, temporal states, or other intermediate tensors.

In contrast to input-level robustness benchmarks, JANUS does not treat broad model degradation as the attack outcome. Instead, JANUS separates safety-critical planning failures from output corruption by targeting collisions and off-road behavior while constraining decoded object and map outputs to remain close to clean inference.

3 Abstract E2E System Model

Modern E2E driving architectures differ substantially in their design choices. For instance, UniAD [6] uses dense query-based interactions across task heads, VAD [9] replaces rasterized maps with

sparse vector representations, and GenAD [38] reformulates planning as a generative task through variational autoencoders. Despite these differences, all three architectures share a common structural property central to our study: planning and observable outputs are coupled through shared latent and task-level states, rather than through explicit decoded interfaces as in modular pipelines.

We abstract this common structure as a multi-task function $F(x)$ that produces a planned trajectory and decoded observable outputs from raw sensor input x , as shown in Fig. 2:

$$\begin{aligned} f &= E(x), & h &= R(f), \\ o &= D_o(h), & z &= D_z(h), \\ y &= P(h, z). \end{aligned} \quad (1)$$

Here, $E(\cdot)$ extracts features f from the raw sensor input, and $R(\cdot)$ maps these features into a shared latent state h . The observable head produces decoded outputs o , such as detections, maps, or motion predictions, through $D_o(\cdot)$. It also produces latent task context z through $D_z(\cdot)$, which abstracts model-specific interactions between perception, prediction, and planning. The planning head $P(\cdot)$ then outputs the ego trajectory y from the shared latent state h and the latent task context z .

The pipeline consists of four stages: ❶ a feature encoder E that extracts features f from raw sensor input x ; ❷ an intermediate representation module R that maps these features into a shared latent state h ; ❸ a perception head that produces decoded observable outputs o through $D_o(h)$, such as 3D object detections, map predictions, or motion predictions; and ❹ a planning head P that outputs the ego vehicle's future trajectory y . The perception head also produces latent task context $z = D_z(h)$, which abstracts model-specific interactions among perception, prediction, and planning before the final trajectory is produced. Importantly, z denotes internal latent context, not the decoded observable output o itself.

The critical property is that the observable outputs o and the latent task context z are both derived from the shared latent state h , and the planning head produces y using this task context together with the shared state. This shared dependency enables joint training and is widely regarded as a strength of E2E architectures. At the same time, it motivates the question we study in this paper: does this shared representation reliably tie safe planning to correct observable outputs? If it does, then a perturbation that substantially disrupts planning should also cause visible errors in the model's observable outputs. If not, then planning may fail even when the model's observable outputs remain largely intact.

Scope of Analysis. We focus on multi-task E2E architectures in which observable outputs and planning outputs are coupled through a shared latent state at inference time. Architectures outside this scope, including diffusion-based planners [14] and parallelized designs such as PARA-Drive [30], are discussed in Section 8.

4 Threat Model

We consider an adversary who aims to induce safety-critical planning failures, such as collisions or lane departures, in a deployed E2E driving system. At the same time, the adversary seeks to keep the observable outputs close to their clean-inference values. This preservation makes the threat more critical: if detections or maps

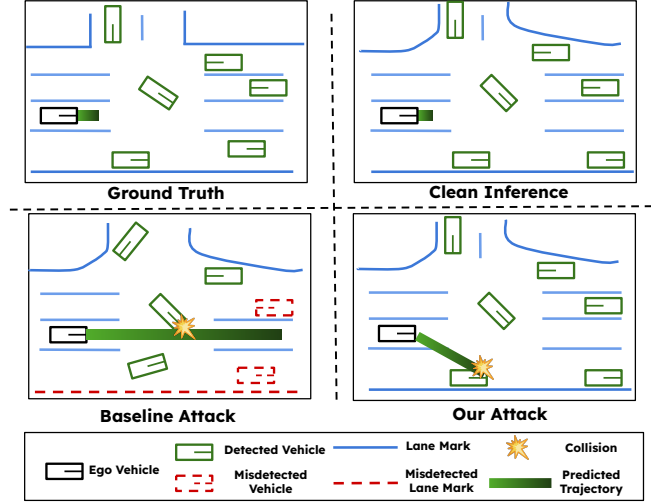


Figure 3: Motivating example. Clean inference produces a safe trajectory with correct observable outputs. A planning-only baseline attack causes a collision but also degrades observable outputs, while JANUS causes a comparable collision while keeping observable outputs close to the clean inference. This shows that planning can fail without comparable degradation in observable outputs.

are also corrupted, output-based monitors can flag the visible errors; on the other hand, if they remain close to clean inference, the model presents a plausible scene while producing an unsafe plan, and the attack is hard to detect.

In achieving this goal, the adversary operates at the sensor-input level: given a clean input x , it produces a perturbed input $x + \delta$. The deployed model then processes the perturbed input and outputs the planned trajectory y and observable outputs o . We consider two perturbation models: ℓ_p -bounded digital noise over the full input and localized adversarial patches over a small image region. Such perturbations can be realized through projected digital perturbations [16, 18, 20] or physical patches [5, 21, 22, 27].

We assume white-box knowledge for perturbation generation, but the adversary can only modify the sensor input. That is, the adversary may use model parameters and gradients to optimize δ , but cannot read, overwrite, or inject perturbations into intermediate latent states during deployment. This separates JANUS from privileged-feature attacks: the attack is generated with strong model knowledge, but it is realized only through input-space changes. We further evaluate cross-model transfer in Section 7.6 to assess a weaker black-box setting where perturbations are generated on one model and applied to another.

5 Motivating Example

A natural intuition about multi-task E2E architectures is that the shared latent representation should tie planning to the model's observable outputs. If the model still detects nearby vehicles correctly and predicts road structure accurately, then the planned trajectory may also appear trustworthy. Under this intuition, a perturbation

strong enough to cause a safety-critical planning failure should also introduce visible errors in the model’s observable outputs, since both depend on the same latent state.

Figure 3 shows that this intuition does not always hold. A projected gradient descent (PGD) attack (bottom-left) that maximizes trajectory error shows the expected behavior: it induces a collision, but also degrades the model’s observable outputs. Vehicles are missed, and lane boundaries are displaced. In this case, the planning failure appears with broad corruption of the shared representation.

JANUS reveals a different behavior (bottom-right). It induces a collision of comparable severity while the model’s observable outputs remain close to the clean inference. The detected vehicles and lane structure appear correct, even though the predicted trajectory becomes unsafe. This example shows that planning can fail without comparable degradation in the model’s observable outputs.

If planning failures always came with visible errors in detections or maps, then those outputs could still serve as a useful proxy for planning reliability. Figure 3 shows that this assumption is too strong for current multi-task E2E architectures. The shared backbone does not guarantee that correct-looking observable outputs imply safe planning. Instead, planning and observable outputs can diverge under adversarial perturbation, despite their dependence on the same latent representation.

6 JANUS System

We present JANUS, an adversarial framework for studying whether multi-task E2E driving systems can produce safety-critical planning failures while their decoded observable outputs remain close to clean inference. Figure 4 illustrates the overall workflow of JANUS. Given an E2E model that outputs a planned trajectory y and decoded observable outputs o , such as object detections and map predictions, JANUS constructs an input-space perturbation or patch that pushes the planned trajectory toward an unsafe outcome while preserving o relative to the clean reference.

JANUS proceeds in three phases. First, JANUS runs a clean forward pass to obtain reference outputs ($y_{\text{clean}}, o_{\text{clean}}$). Second, JANUS performs offline perturbation optimization: it selects safety-relevant targets, constructs a joint objective that combines a planning-failure objective (Section 6.1) with observable-output preservation (Section 6.2), and solves the constrained input-space attack (Section 6.3). For patch attacks, this phase also includes sensitivity-based placement (Section 6.4). Third, JANUS applies the optimized perturbation or patch at inference time to induce unsafe planning while keeping observable outputs close to the clean reference.

6.1 Route-Conditioned Collision Loss

Standard planning attacks often maximize the ℓ_2 distance between the clean trajectory and the adversarial trajectory. However, large geometric deviation alone does not reliably produce severe safety failures. In many cases, this objective yields comparatively benign behaviors, such as drifting into empty space or unnecessary deceleration, rather than collisions or off-road events.

A natural alternative is to minimize the distance to the nearest scene obstacle. However, this objective ignores vehicle kinematics. The spatially nearest obstacle may lie behind the ego vehicle or outside its forward reachable corridor. Pushing the planner toward

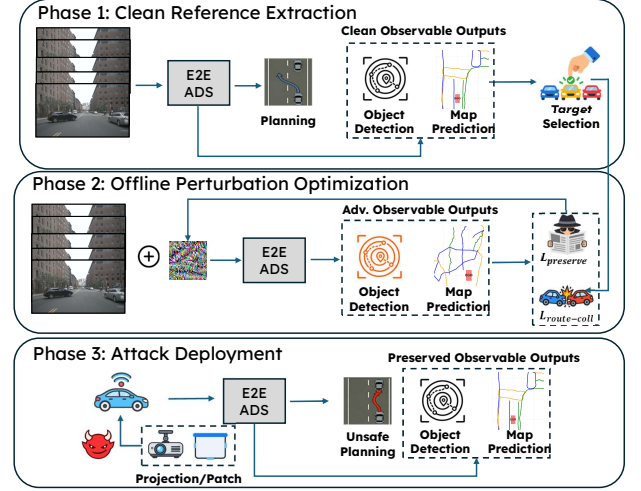


Figure 4: Illustration of JANUS’s attack phases.

such targets can require physically implausible maneuvers, which leads the optimization toward degenerate solutions.

To address this limitation, we introduce the Route-Conditioned Collision Loss ($\mathcal{L}_{\text{route-coll}}$), which restricts the attack objective to obstacles that lie along the clean driving route. This focuses the perturbation budget on targets that are both safety-relevant and consistent with the ego vehicle’s forward driving direction.

Target Selection. We first define a speed-adaptive angular threshold $\theta(v)$ that determines the valid attack corridor:

$$\theta(v) = \min \left(\theta_{\max}, \frac{b}{\max(v, v_{\min})} \right). \quad (2)$$

We then filter the set of scene obstacles $\mathcal{O}$ and retain only those that lie longitudinally ahead of the ego vehicle and within the angular corridor induced by the clean route:

$$\mathcal{V} = \left\{ i \in \mathcal{O} \mid (\mathbf{p}_i^\top \mathbf{d} > 0) \wedge \left(\frac{|\mathbf{p}_i \times \mathbf{d}|}{\mathbf{p}_i^\top \mathbf{d}} < \tan \theta(v) \right) \right\} \quad (3)$$

where $\mathbf{d}$ is the clean heading vector. The selected target i^* is the element of $\mathcal{V}$ with minimum lateral offset $|\mathbf{p}_i \times \mathbf{d}|$. Figure 5 illustrates this selection rule: JANUS only considers obstacles that are forward and within the speed-adaptive angular corridor of the clean route.

This target-selection rule has two benefits. First, it removes obstacles that are inconsistent with the ego vehicle’s forward driving corridor, such as objects behind the ego vehicle or objects far outside the driving direction. Second, it anchors the attack to the clean driving intent rather than to an already corrupted trajectory. As a result, JANUS avoids degenerate cases in which the optimization first drives the planner toward an unrealistic direction and then optimizes collision relative to that corrupted path.

Objective Formulation. To account for scene dynamics, we extrapolate the selected target’s future position at timestep t as

$$\hat{\mathbf{p}}_{i^*}^{(t)} = \mathbf{p}_{i^*} + \mathbf{v}_{i^*} \cdot t \Delta t, \quad (4)$$

where $\mathbf{p}_{i^*}$ and $\mathbf{v}_{i^*}$ denote the target’s relative position and velocity in the ego-centric frame.

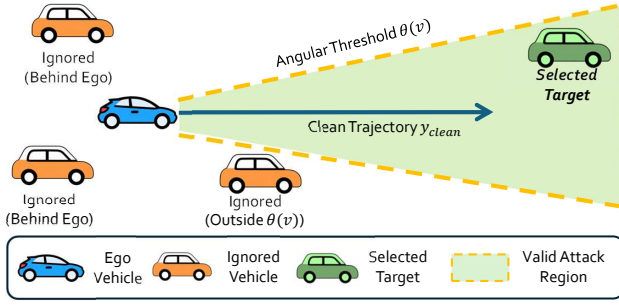


Figure 5: Route-conditioned target selection. JANUS selects i^* only from the forward, route-consistent valid region $\mathcal{V}$.

We define the geometric attraction term as

$$\mathcal{L}_{\text{geo}} = \frac{1}{T} \sum_{t=1}^T w_t \left\| \mathbf{y}_{\text{adv}}^{(t)} - \hat{\mathbf{p}}_{i^*}^{(t)} \right\|_2, \quad (5)$$

and the soft collision reward as

$$\mathcal{L}_{\text{coll}} = -\sigma\left(\frac{r_c - d_{\min}}{\tau}\right), \quad d_{\min} = \min_t \left\| \mathbf{y}_{\text{adv}}^{(t)} - \hat{\mathbf{p}}_{i^*}^{(t)} \right\|_2. \quad (6)$$

Here, w_t weights timesteps, r_c is the collision radius, τ sets reward smoothness, and $\sigma(\cdot)$ denotes the sigmoid function.

Finally, we use a directional regularization term to keep the optimized trajectory aligned with the clean driving direction:

$$\mathcal{L}_{\text{dir}} = 1 - \frac{\Delta \mathbf{y}_{\text{adv}} \cdot \mathbf{d}}{\|\Delta \mathbf{y}_{\text{adv}}\|_2 \|\mathbf{d}\|_2 + \eta}. \quad (7)$$

where $\Delta \mathbf{y}_{\text{adv}} = \mathbf{y}_{\text{adv}}^{(T)} - \mathbf{y}_{\text{adv}}^{(1)}$, $\mathbf{d}$ is the clean route direction, and η is a small constant for numerical stability.

The full route-conditioned collision loss is

$$\mathcal{L}_{\text{route-coll}} = \lambda_{\text{geo}} \mathcal{L}_{\text{geo}} + \lambda_{\text{coll}} \mathcal{L}_{\text{coll}} + \lambda_{\text{dir}} \mathcal{L}_{\text{dir}}. \quad (8)$$

If the valid target set $\mathcal{V}$ is empty, JANUS skips the frame and does not generate a route-conditioned collision objective for it.

6.2 Perceptual Consistency Constraints

The goal of JANUS is not only to induce an unsafe plan, but also to preserve the model's observable outputs. This preservation requirement serves two purposes. First, it keeps the attacked inference close to the clean reference on outputs that are directly visible to the system designer, such as detections and maps. Second, it helps isolate whether planning can fail without comparable corruption of the model's observable predictions. To enforce this property, we introduce two preservation losses: one for dynamic object detection and one for static map prediction.

Object Detection Consistency ($\mathcal{L}_{\text{det}}$). We preserve the model's understanding of dynamic agents by constraining the adversarial detections to remain close to the clean reference detections from the same frame. Let $\hat{\mathbf{b}}_{\text{clean}} = \{(\mathbf{c}_k, \mathbf{r}_k)\}_{k=1}^K$ denote the clean detection predictions, where $\mathbf{c}_k$ are class logits and $\mathbf{r}_k$ are bounding-box parameters. Let $\hat{\mathbf{b}}_{\text{adv}}$ denote the corresponding adversarial predictions at the same detection-query indices. We define $\mathbf{p}_k = \text{softmax}(\mathbf{c}_k)$

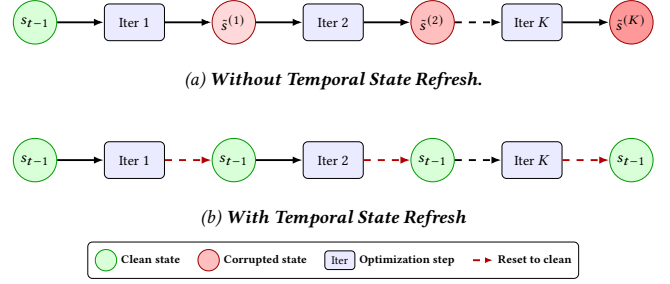


Figure 6: Temporal State Refresh in JANUS. (a) Without temporal state refresh, the temporal state is carried across optimization iterations and gradually drifts away from the clean reference. (b) With temporal state refresh, JANUS resets the temporal state to the clean reference before each iteration, which keeps the optimization anchored to the current frame and avoids accumulated state drift.

as the corresponding class-probability distribution. We define the detection preservation loss as

$$\mathcal{L}_{\text{det}} = \frac{1}{K} \sum_{k=1}^K \left(\left\| \mathbf{r}_{k,\text{adv}} - \mathbf{r}_{k,\text{clean}} \right\|_1 + \lambda_{\text{cls}} D_{\text{KL}}(\mathbf{p}_{k,\text{clean}} \parallel \mathbf{p}_{k,\text{adv}}) \right) \quad (9)$$

The first term preserves spatial localization, while the second preserves semantic confidence. Together, they keep the adversarial detections close to the model's clean detection of the scene.

Map Structure Consistency ($\mathcal{L}_{\text{map}}$). We preserve the geometric structure of the static scene by keeping the adversarial map output close to the clean reference map. Let $\mathcal{M}_{\text{clean}}$ and $\mathcal{M}_{\text{adv}}$ denote the clean and adversarial map points. We define the map preservation loss using the symmetric Chamfer Distance:

$$\mathcal{L}_{\text{map}} = \sum_{x \in \mathcal{M}_{\text{clean}}} \min_{y \in \mathcal{M}_{\text{adv}}} \|x - y\|_2^2 + \sum_{y \in \mathcal{M}_{\text{adv}}} \min_{x \in \mathcal{M}_{\text{clean}}} \|x - y\|_2^2 \quad (10)$$

This loss preserves lane geometry and road boundaries under attack, so the attacked map output remains close to the clean reference rather than exhibiting large topological shifts.

In practice, both preservation losses are defined with respect to the clean inference on the same frame. This makes the preservation objective model-consistent: instead of enforcing agreement with external labels, JANUS asks whether the attacked inference remains close to the model's own clean observable outputs. We focus on detection and map prediction because they are the most visible auxiliary heads shared across the evaluated models and the most natural signal for consistency-based monitoring.

6.3 Optimization Strategy

We optimize JANUS as a constrained input-space attack that balances unsafe planning with observable-output preservation. The total objective combines the route-conditioned collision loss with the detection and map preservation losses:

$$\mathcal{L}_{\text{total}} = \underbrace{\lambda_{\text{route}} \mathcal{L}_{\text{route-coll}}}_{\text{Safety Violation}} + \underbrace{\lambda_{\text{det}} \mathcal{L}_{\text{det}} + \lambda_{\text{map}} \mathcal{L}_{\text{map}}}_{\text{Perceptual Preservation}} \quad (11)$$

Algorithm 1 JANUS Attack Optimization

```

1: Input: Current frame  $\mathbf{x}_t$ , clean temporal state  $\mathbf{s}_{t-1}^{\text{ref}}$ , model  $F$ ,
   attack mode  $m \in \{\text{DIGITAL}, \text{PATCH}\}$ 
2: Hyperparameters: steps  $K$ , step size  $\alpha$ , budget  $\epsilon$ , patch size
    $(H_p, W_p)$ , weights  $\lambda_{\text{route}}, \lambda_{\text{det}}, \lambda_{\text{map}}$ 
3: Output: Adversarial input  $\mathbf{x}_{\text{adv}}$ , clean propagated state  $\mathbf{s}_t^{\text{ref}}$ 
4: // Clean Reference Extraction
5:  $(\mathbf{y}_{\text{ref}}, \mathbf{o}_{\text{ref}}, \mathbf{s}_{\text{clean}}) \leftarrow F(\mathbf{x}_t, \mathbf{s}_{t-1}^{\text{ref}})$ 
6:  $\mathbf{s}_t^{\text{ref}} \leftarrow \mathbf{s}_{\text{clean}}$ 
7: // Patch Mask Initialization
8: if  $m = \text{PATCH}$  then
9:    $\mathbf{M} \leftarrow \text{SELECTPATCHREGION}(\mathbf{x}_t, \mathbf{s}_{\text{clean}}, H_p, W_p)$ 
10: else
11:    $\mathbf{M} \leftarrow \mathbf{1}$ 
12: end if
13: // Iterative Optimization
14: Initialize perturbation  $\delta^{(0)} \leftarrow \mathbf{0}$  or random start
15: for  $k = 0$  to  $K - 1$  do
16:   Reset temporal state:  $\mathbf{s} \leftarrow \mathbf{s}_{t-1}^{\text{ref}}$ 
17:   Form attacked input:
18:      $\mathbf{x}^{(k)} \leftarrow (\mathbf{1} - \mathbf{M}) \odot \mathbf{x}_t + \mathbf{M} \odot (\mathbf{x}_t + \delta^{(k)})$ 
19:    $(\mathbf{y}_{\text{adv}}, \mathbf{o}_{\text{adv}}, \_) \leftarrow F(\mathbf{x}^{(k)}, \mathbf{s})$ 
20:    $\mathcal{L}_{\text{total}} \leftarrow \lambda_{\text{route}} \mathcal{L}_{\text{route-coll}}(\mathbf{y}_{\text{adv}})$ 
21:      $+ \lambda_{\text{det}} \mathcal{L}_{\text{det}}(\mathbf{o}_{\text{adv}}, \mathbf{o}_{\text{ref}}) + \lambda_{\text{map}} \mathcal{L}_{\text{map}}(\mathbf{o}_{\text{adv}}, \mathbf{o}_{\text{ref}})$ 
22:    $\delta^{(k+1)} \leftarrow \delta^{(k)} - \alpha \cdot \text{sign}(\nabla_{\delta} \mathcal{L}_{\text{total}})$ 
23:   if  $m = \text{DIGITAL}$  then
24:      $\delta^{(k+1)} \leftarrow \Pi_{\epsilon}(\delta^{(k+1)})$ 
25:   else
26:     Clip  $\delta^{(k+1)}$  to the valid pixel range
27:   end if
28: end for
29: // Final Adversarial Input
30:  $\mathbf{x}_{\text{adv}} \leftarrow (\mathbf{1} - \mathbf{M}) \odot \mathbf{x}_t + \mathbf{M} \odot (\mathbf{x}_t + \delta^{(K)})$ 
31: return  $\mathbf{x}_{\text{adv}}, \mathbf{s}_t^{\text{ref}}$ 

```

where the λ terms control the trade-off between attack strength and preservation.

For each frame, JANUS first computes a clean reference trajectory, observable outputs, and temporal state from the unperturbed input. These clean outputs serve as the reference for the preservation constraints, while the clean temporal state is buffered for the next timestamp (Algorithm 1, Lines 5–6).

Temporal State Refresh. A central challenge in attacking video-based E2E planners such as GenAD and UniAD is their dependence on temporal history, for example through BEV memory queues. If this state is carried across optimization iterations, the attack gradually conditions on a drifted history rather than the clean one. To avoid this, JANUS uses a **Temporal State Refresh** strategy. Figure 6 illustrates the difference: without refresh, each optimization step inherits the state produced by the previous step; with refresh, each step starts from the same clean temporal state.

Let $\mathbf{s}_{t-1}$ denote the clean temporal state from the previous timestamp. During optimization of the current frame t , we keep $\mathbf{s}_{t-1}$ fixed and reset the model’s internal temporal memory to this same

clean state at every iteration k :

$$\mathbf{x}_t^{(k+1)} = \Pi_{\epsilon} \left(\mathbf{x}_t^{(k)} - \alpha \cdot \text{sign} \left(\nabla_{\mathbf{x}} \mathcal{L}_{\text{total}}(F(\mathbf{x}_t^{(k)}, \mathbf{s}_{t-1})) \right) \right) \quad (12)$$

This refresh step should be distinguished from the clean-state buffer used across timestamps. During optimization of frame t , every update is computed under the same fixed history condition $\mathbf{s}_{t-1}$. After the attack on the current frame is finalized, the clean temporal state is propagated forward for the next timestamp. This separation keeps the optimization focused on the current frame without compounding perturbation-induced drift through the temporal buffer. Algorithm 1 implements this by resetting the temporal state before each optimization step (Line 16) and then updating the perturbation under this fixed history condition (Lines 16–22). Figure 6 illustrates this distinction: without refresh, the temporal state drifts across optimization iterations, whereas JANUS resets the temporal state to the same clean reference before each iteration.

Constraint Realization. For digital attacks, we use the standard ℓ_{∞} projection Π_{ϵ} to keep the perturbation within a bounded input-space budget. For patch attacks, we optimize a localized perturbation under a binary mask $\mathbf{M}$ that specifies the patch region, so only the selected pixels are modified.

6.4 Patch Placement Optimization

For patch-based attacks, JANUS places the perturbation using a sensitivity-based search rather than random placement. The goal is to allocate the limited patch budget to the image region that most strongly influences the planning output.

We first compute an input-space sensitivity map from the gradient magnitude of the route-conditioned planning objective with respect to the current frame. This map identifies regions where the planner is most responsive under the clean reference state.

Given the sensitivity map, JANUS performs a sliding-window search over candidate regions of size $H_p \times W_p$ and selects the window with the largest accumulated sensitivity. The binary mask $\mathbf{M}$ is then initialized at the selected location and used throughout the masked optimization loop.

This strategy improves patch effectiveness under a fixed patch size by concentrating the perturbation on the image region that is most important to the planner.

7 Evaluation

We evaluate JANUS against three popular end-to-end ADSs: UniAD [6], VAD [9] and GenAD [38]. We present JANUS’s results by focusing on the following research questions:

- RQ1** How effectively can input-space adversaries induce safety-critical planning failures in E2E ADSs?
- RQ2** To what extent can input-space adversaries preserve observable object and map outputs while inducing safety-critical planning failures?
- RQ3** How much observable-output inconsistency remains available to consistency-based monitors?
- RQ4** Why can safety-critical planning failures occur without comparable degradation in observable perception outputs in shared-latent E2E architectures?
- RQ5** How well do perturbations generated on one E2E ADS transfer to other architectures in causing collisions?

7.1 Evaluation Setup

Datasets and Models. We test three end-to-end autonomous driving models: UniAD [6], VAD [9], and GenAD [38]. As discussed in Section 3, these models represent three different design choices, but they share the property studied in this paper: the planned trajectory and decoded observable outputs are coupled through shared latent and task-level states. This property is central to our study because it lets us test whether unsafe planning can occur even when the observable outputs remain close to clean inference.

Evaluation Metrics. We evaluate attacks along three aspects: planning impact, object detection accuracy and map prediction accuracy. Planning impact captures the safety consequence, while detection and map accuracy capture whether observable scene outputs remain reliable. Evaluating these together separates planning-specific failures from attacks corrupting the shared representation.

For planning impact, we report Collision Rate (CR), Off-Road Rate (ORR), and Average Displacement Error (ADE). CR and ORR measure two safety-critical failures: CR captures unsafe interaction with objects, while ORR captures violations of the drivable area. We report both because an attack may cause collisions without leaving the road, or leave the road without colliding with another object. ADE measures the deviation of the predicted trajectory from the ground-truth trajectory, but large trajectory error alone does not necessarily imply a safety violation.

For object detection accuracy, we report the nuScenes Detection Score (NDS), which summarizes detection quality across localization, scale, orientation, velocity, attribute, and classification terms. For map prediction accuracy, we report mAP for GenAD and VAD because their map heads output vectorized map elements. For UniAD, we report mIoU because its map head outputs rasterized BEV semantic maps. We compare these task metrics against their clean-inference values to quantify how much observable-output quality degrades under attack.

Together, these metrics show whether the attack can cause unsafe planning while limiting degradation in observable outputs relative to clean inference.

Baselines. We compare JANUS with three attack variants that all modify only the sensor input and use the same iterative projected optimization setting. The first is Trajectory Deviation, a planning-only baseline that maximizes the distance between clean and adversarial trajectories. This baseline tests whether simply moving the predicted trajectory away from its clean state is sufficient to cause safety-critical failures. The second is Trajectory Deviation + Preservation, which adds the same observable-output preservation losses to this baseline. This variant isolates whether preservation alone can keep object and map outputs close to clean inference when the planning objective remains trajectory deviation. The third is JANUS w/o Preservation, which uses our route-conditioned planning objective without observable-output preservation. This variant isolates the effect of the preservation constraint within JANUS.

Attack Settings. Unless stated otherwise, we generate attacks in a white-box setting under the sensor-input constraints defined in Section 4: the adversary can modify only camera inputs, either through an ℓ_∞ -bounded perturbation or a localized patch, and cannot perturb internal features. The main digital attack uses $\epsilon = 8/255$, attacks all six cameras, and runs 20 PGD steps. We also evaluate

localized patch attacks and cross-model transfer to test whether the attack persists under spatially constrained and less model-specific settings. For transfer, we optimize perturbations on a source model and apply them to a target model without further optimization.

7.2 Attack Effectiveness (RQ1)

We first evaluate whether JANUS can induce safety-critical planning failures in three E2E ADSs. The *Planning Impact* columns in Table 1 report the main results. We compare JANUS with three input-space attack variants: a trajectory-deviation baseline, the same baseline with preservation constraints, and JANUS without preservation. This comparison helps attribute the results to two design choices in JANUS: the route-conditioned planning objective and the output-preservation constraint.

Across all three models, JANUS substantially increases unsafe planning outcomes over clean inference. JANUS raises the CR from 3.34% to 49.81% (14.9 $\times$) and the off-road rate from 0.15% to 13.80% (92 $\times$) on GenAD, from 8.50% to 30.68% (3.6 $\times$) and 0.93% to 10.89% (11.7 $\times$) on VAD, and from 5.1% to 17.1% (3.4 $\times$) and 5.2% to 8.7% (1.7 $\times$) on UniAD. These results show that JANUS can induce safety-critical planning failures across different E2E architectures.

We note that JANUS does not achieve the largest attack effect for every planning metric in Table 1. This is expected because the compared attack variants are generated under different objectives. The trajectory-deviation baseline can sometimes produce larger raw CR, ORR, or ADE, but it is not constrained to keep observable outputs close to clean inference. Thus, RQ1 focuses on whether JANUS can still cause safety-critical planning failures under its preservation-aware objective. We return to the output-preservation side of this tradeoff in RQ2.

The comparison with JANUS w/o Preservation further shows that this constraint affects architectures differently. Adding preservation only modestly reduces CR on GenAD (51.55% to 49.81%) and VAD (35.33% to 30.68%), but the reduction is larger on UniAD (28.6% to 17.1%), which suggests a stronger tension between planning failure and output preservation. Even under this stricter constraint, JANUS still causes substantial safety-critical failures.

Patch-Based Attacks. We next evaluate JANUS under localized patch constraints. Table 2 summarizes the results for three-camera patch attacks, with the patch area matched to about 4% of each model’s input tensor. Even when the perturbation is restricted to a small image region, JANUS can still induce planning failures while keeping observable-output degradation small.

The results again show an architecture-dependent tradeoff. On GenAD and VAD, JANUS achieves stronger planning impact than the strongest preservation-based patch baseline. On GenAD, the CR increases from 10.92% to 17.21%, and the ORR increases from 11.07% to 16.15%, while Δ NDS remains the same and Δ Map changes only slightly. On VAD, JANUS raises the CR from 8.87% to 12.41% with nearly unchanged ORR, although observable-output degradation increases slightly. On UniAD, JANUS yields slightly lower CR and ORR than the baseline, but it better preserves observable outputs, reducing both Δ NDS and Δ Map.

Table 1: Attack effectiveness and output preservation on nuScenes. We report Collision Rate (CR), Off-Road Rate (ORR), and Average Displacement Error (ADE) to measure planning impact, and NDS and Map Score to measure observable-output preservation. Map Score denotes mAP for GenAD and VAD, and mIoU for UniAD.

Model	Method	Planning Impact			Output Preservation	
		CR (%)↑	ORR (%)↑	ADE (m)↑	NDS↑	Map Score↑
GenAD	Clean	3.34	0.15	1.34	39.1	76.7
	Traj. Deviation	36.16	19.11	15.89	22.3 (-42.9%)	45.5 (-40.7%)
	Traj.+ Preservation	27.82	17.51	12.96	29.9 (-23.5%)	63.0 (-17.9%)
	JANUS (w/o Preservation)	51.55	14.33	6.36	28.6 (-26.7%)	56.5 (-26.4%)
	JANUS	49.81	13.80	6.29	32.2 (-17.5%)	64.0 (-16.6%)
VAD	Clean	8.50	0.93	1.52	66.7	66.4
	Traj. Deviation	31.21	17.93	11.79	47.6 (-28.7%)	38.5 (-42.0%)
	Traj.+ Preservation	19.92	11.02	4.59	61.1 (-8.3%)	60.8 (-8.5%)
	JANUS (w/o Preservation)	35.33	12.48	6.42	51.9 (-22.1%)	51.2 (-22.9%)
	JANUS	30.68	10.89	5.67	56.8 (-14.8%)	57.1 (-14.0%)
UniAD	Clean	5.1	5.2	1.22	55.8	44.2
	Traj. Deviation	32.6	20.7	16.71	37.8 (-32.2%)	33.0 (-25.4%)
	Traj.+ Preservation	27.8	17.0	14.20	40.1 (-28.3%)	37.0 (-16.4%)
	JANUS (w/o Preservation)	28.6	11.8	6.10	43.5 (-22.2%)	36.6 (-17.3%)
	JANUS	17.1	8.7	2.52	50.7 (-9.1%)	40.5 (-8.5%)

Table 2: Patch-based attack summary for three-camera attacks with patch area matched to about 4% of each model’s input tensor. Lower Δ NDS and Δ Map indicate smaller observable-output degradation.

Model	Method	CR (%)↑	ORR (%)↑	Δ NDS (%)↓	Δ Map (%)↓
GenAD	Traj. + Preservation	10.92	11.07	3.4	8.1
	JANUS	17.21	16.15	3.4	8.7
VAD	Traj. + Preservation	8.87	3.82	4.3	3.8
	JANUS	12.41	3.81	5.0	5.1
UniAD	Traj. + Preservation	12.36	9.44	3.6	4.8
	JANUS	11.41	8.95	2.8	4.0

Overall, the patch results show that JANUS is not limited to dense digital perturbations. Under localized patch constraints, JANUS remains competitive in planning impact while maintaining small observable-output degradation across all three architectures.

Sensitivity to Attack Budget. We next conduct a separate set of evaluation runs to examine how attack strength changes under three budget controls: the digital perturbation budget ϵ , the number of attacked cameras, and the patch coverage. Figure 7 shows that larger budgets usually produce stronger planning failures, but the effect is not perfectly monotonic. Under the digital ϵ sweep, both VAD and GenAD show their strongest impact at moderate-to-large budgets, while UniAD remains comparatively stable across the tested range. For example, VAD’s CR increases from 29.61% at $\epsilon=2/255$ to 35.86% at $\epsilon=4/255$, and GenAD stays above 44% CR across all four budgets. In contrast, UniAD remains near 20% CR and about 8% off-road rate throughout the sweep.

A similar trend appears when we increase the number of attacked cameras. Broader camera access usually gives the adversary more

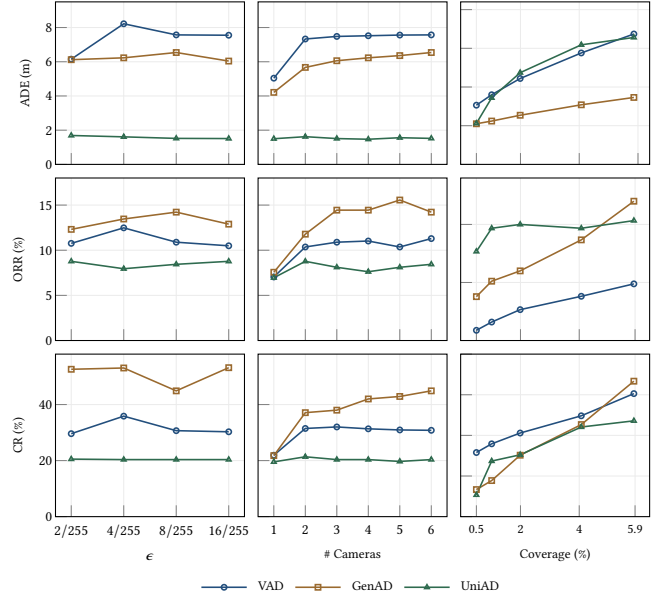


Figure 7: Attack budget sweep across models. Columns: perturbation budget ϵ , number of attacked cameras ($\epsilon=8/255$), and patch coverage (%). Rows: Average Displacement Error (ADE), off-road rate (ORR), and collision rate (CR) at 3 s. The sensitivity sweeps were conducted independently from the main evaluation in Table 1.

control over the shared scene representation and leads to higher attack success, especially for GenAD. Its CR rises from 21.78% with

one attacked camera to 44.89% with six, while its off-road rate increases from 7.56% to 14.22%. VAD also improves from 21.91% to about 31% CR, although the gain saturates after two cameras. UniAD again changes only modestly, which suggests that JANUS can affect planning even with limited view access but benefits differently across model architectures.

The patch sweep shows the clearest budget-response trend. As coverage grows from about 0.5% to 5.9%, CR increases on all three models: from 7.89% to 15.14% on VAD, from 3.33% to 16.67% on GenAD, and from 2.67% to 11.79% on UniAD. Off-road rate and ADE also rise with coverage, although UniAD's off-road rate saturates earlier than the other two models. These results show that JANUS degrades planning safety through multiple failure modes, and that larger patch area translates into a more direct and consistent increase in attack strength than the digital ϵ sweep.

7.3 Output Preservation (RQ2)

We next ask whether JANUS preserves observable outputs when it induces planning failures. The *Output Preservation* columns in Table 1 report object detection and map prediction quality on GenAD, VAD, and UniAD. We use relative degradation from clean inference to measure how much each attack changes these outputs.

For object detection, JANUS causes substantially smaller NDS degradation than the Trajectory Deviation baseline on all three models. The NDS drop decreases from 42.9% to 17.5% on GenAD, from 28.7% to 14.8% on VAD, and from 32.2% to 9.1% on UniAD. These results show that JANUS keeps object detection outputs much closer to clean inference than a trajectory-only attack.

We observe the same pattern for map prediction. On GenAD, the Map Score drop decreases from 40.7% under Trajectory Deviation to 16.6% under JANUS. On VAD, it decreases from 42.0% to 14.0%. On UniAD, it decreases from 25.4% to 8.5%. This result shows that JANUS preserves map outputs much more effectively than the trajectory-only baseline while still causing unsafe trajectories.

The comparison with JANUS w/o Preservation further clarifies the role of the preservation term. After we add preservation constraints, the NDS drop decreases from 26.7% to 17.5% on GenAD, from 22.1% to 14.8% on VAD, and from 22.2% to 9.1% on UniAD. The Map Score drop follows the same trend: it decreases from 26.4% to 16.6% on GenAD, from 22.9% to 14.0% on VAD, and from 17.3% to 8.5% on UniAD. These results show that the preservation objective consistently reduces observable-output degradation on both tasks.

Finally, Table 1 highlights that preservation should be interpreted together with planning impact. Trajectory Deviation + Preservation can also reduce observable-output degradation, and on VAD it preserves NDS and Map Score more than JANUS. However, it also produces a lower CR and lower ADE than JANUS, with a comparable off-road rate. Across the three models, JANUS maintains substantial safety-critical planning failures while keeping both NDS and Map Score degradation below 20%. Overall, the results show that JANUS can induce unsafe planning while keeping observable outputs close to clean inference.

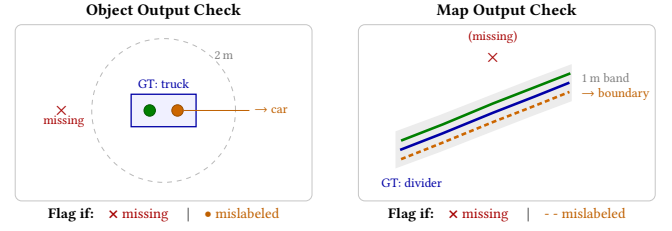


Figure 8: Clean-reference output check used in RQ3. A frame is flagged if a clean-correct object (left) or map polyline (right) is **missing or **mislabeled** under attack, with a 2 m matching radius for objects and a 1 m band for polylines.**

Table 3: Flag rates for consistency-based monitors across attack formulations. Lower rates indicate stronger evasion.

Target Model	Attack Method (PGD)	Object-Output Flag Rate ↓	Map-Output Flag Rate ↓
GenAD	Traj. Deviation	85.4%	80.0%
	Traj. Deviation + Preservation	66.6%	51.3%
	JANUS w/o Preservation	62.5%	61.5%
	JANUS	55.1%	42.7%
VAD	Traj. Deviation	75.2%	77.0%
	Traj. Deviation + Preservation	22.8%	19.8%
	JANUS w/o Preservation	75.3%	57.4%
	JANUS	50.8%	40.6%
UniAD	Traj. Deviation	90.8%	83.7%
	Traj. Deviation + Preservation	75.2%	61.8%
	JANUS w/o Preservation	73.3%	58.6%
	JANUS	32.1%	30.4%

Table 4: Post-monitor collision success rate under the object- and map-output monitors. A successful attack both induces a collision and is not flagged by the corresponding monitor.

Target Model	Attack (PGD)	Object Monitor ↑	Map Monitor ↑
GenAD	Traj. Deviation	4.6%	8.6%
	JANUS	14.3%	20.7%
VAD	Traj. Deviation	5.3%	5.5%
	JANUS	6.1%	9.8%
UniAD	Traj. Deviation	1.6%	2.5%
	JANUS	5.5%	10.5%

7.4 Implications for Consistency-Based Detection (RQ3)

We next study whether JANUS leaves enough observable-output changes for existing consistency-based defenses to detect the attack. Existing defenses use observable outputs, such as object labels, object locations, and map elements, to detect attacks through visible inconsistencies [17, 31, 33]. Here, we test whether JANUS reduces this detection signal by keeping these outputs close to clean inference. Unlike RQ2, which reports aggregate task metrics, this analysis asks whether clean-correct object and map outputs visibly change on each *frame* under attack.

Specifically, we measure observable-output changes with a clean-reference output check at each frame, illustrated in Figure 8. For object outputs, we first identify ground-truth objects that the clean model predicts correctly. An object is clean-correct if the clean prediction matches it within 2 m in BEV and has the correct class label. We then compare this same object under attack. If no attacked prediction matches the object within 2 m, we mark the object as missing. If an attacked prediction matches the object but has a different class label, we mark the object as mislabeled.

The object-based check flags a frame if at least one clean-correct object in that frame is missing or mislabeled under attack. The Object-Output Flag Rate is the percentage of evaluable frames flagged by this rule. For map outputs, we identify clean-correct polylines and match attacked predictions using a 1 m band. A frame is flagged if at least one clean-correct polyline becomes missing or is assigned a different map class under attack. The Map-Output Flag Rate is the percentage of evaluable frames flagged by this rule.

Table 3 shows that JANUS leaves fewer visible changes in decoded object and map outputs than attacks without preservation. The Trajectory Deviation baseline generally produces high flag rates because it causes changes in perception outputs. Adding preservation constraints reduces these rates. The same trend appears for JANUS: compared with JANUS without preservation, JANUS reduces object- and map-output flag rates across the evaluated models.

To measure end-to-end attack effectiveness under monitoring, Table 4 reports the fraction of frames where the attack induces a collision and is not flagged. JANUS achieves higher post-monitor collision success than Trajectory Deviation across all three models, reaching 14.3%, 6.1%, and 5.5% under the object monitor and 20.7%, 9.8%, and 10.5% under the map monitor for GenAD, VAD, and UniAD, respectively.

These results show that JANUS weakens the signal used by existing consistency-based defenses. Such defenses can still flag attacks that visibly corrupt detections or maps. However, when an attack induces unsafe planning while keeping decoded object and map outputs close to clean inference, output-based consistency checks become less reliable.

7.5 Root Cause Analysis (RQ4)

The results show that JANUS can induce unsafe planning while keeping observable outputs close to clean inference. We examine why this behavior is possible in shared-latent E2E architectures. We focus on two questions: whether planning and observable heads respond similarly to input perturbations, and whether the preservation term suppresses observable-output drift during optimization.

Planning-Perception Coupling. A natural assumption about multi-task E2E driving is that planning, detection, and map prediction should respond in similar ways to an input perturbation because they depend on the same latent representation h . If this coupling were tight, a perturbation that disrupts planning would also tend to induce comparable changes in the observable outputs.

We test this assumption at the local linear level by measuring the cosine similarity between the input gradients of the planning, detection, and map heads on clean validation frames. Across UniAD, VAD, and GenAD, all mean similarities have magnitude below 0.07; we report the full table and protocol in Appendix D.

Table 5: Raw head-output drift d_P , d_D , d_M for Traj. Deviation and its preservation-constrained variant.

Model	Attack	$d_P \uparrow$	$d_D \downarrow$	$d_M \downarrow$
GenAD	Traj. Deviation	1.81	0.244	0.385
	Traj. + Preservation	1.52	0.201 (−18%)	0.305 (−21%)
VAD	Traj. Deviation	5.05	0.185	0.327
	Traj. + Preservation	2.04	0.067 (−64%)	0.106 (−68%)
UniAD	Traj. Deviation	0.030	0.046	0.067
	Traj. + Preservation	0.023	0.037 (−20%)	0.055 (−18%)

This result suggests that the three heads do not share a strongly aligned input direction. This first-order analysis captures the gradient geometry at clean input. It does not characterize how the heads respond once PGD moves the input far from the clean reference, which we examine next through raw output drift.

Effect of the Preservation Term. The gradient analysis above gives a local explanation: near the clean input, planning, detection, and map heads do not move in strongly aligned directions. However, this local result does not by itself show that an optimized perturbation will keep observable outputs close to clean inference. To isolate the role of the preservation term, we compare two matched attacks that share the same planning objective and differ only in whether preservation is enabled: Traj. and Traj. + Preservation. This matched pair keeps the planning objective fixed and changes only the preservation constraint, which isolates the effect of preservation. We measure raw output drift at each head:

$$d_X = \frac{\|X(x + \delta) - X(x_{\text{clean}})\|_2}{\|X(x_{\text{clean}})\|_2},$$

where $X \in \{P, D, M\}$ denotes the trajectory, detection, and map heads. Table 5 reports the mean drift across validation frames.

The pattern is consistent across all three models. After we add the preservation term, the detection drift d_D drops by 18% to 64%, and the map drift d_M drops by 18% to 68%. Because the planning objective is unchanged across this pair, the most direct explanation is that the preservation term suppresses drift in the observable heads during optimization.

The drift values above measure suppression in the raw output space, where the preservation loss directly acts, while Table 1 reports the decoded task metrics. The two views are complementary: raw drift confirms that the preservation term suppresses change inside the optimization, and the decoded metrics confirm that this suppression carries through to the observable outputs. Together, these results suggest that weak cross-head alignment leaves room for selective disruption, and the preservation term exploits that room by suppressing drift in the observable heads.

7.6 Cross-Model Transferability (RQ5)

We examine whether a perturbation optimized on one E2E ADS can affect another architecture. For each source-target pair, we optimize a white-box perturbation on the source model and retain the frames where the attack successfully induces a collision. We then convert the perturbation to the target model’s input space and evaluate it on the target model without further optimization. We define the

Table 6: Cross-model transferability measured by Transfer Success Rate (TSR, %). TSR denotes the fraction of source-successful attack frames that also induce a collision on the target model after transfer.

Source model	Target model		
	UniAD	VAD	GenAD
UniAD	-	0.0	12.3
VAD	46.9	-	5.9
GenAD	6.2	10.7	-

Transfer Success Rate (TSR) as the fraction of these source-successful frames that also induce a collision on the target model. Because the models use different input resolutions and preprocessing pipelines, this conversion removes source padding, undoes source normalization, resizes the perturbation to the target resolution, applies target normalization, and preserves the six-camera correspondence.

Table 6 reports the TSR under this setting. The transfer results are asymmetric and source-target dependent. The clearest transfer case is VAD→UniAD, where 46.9% of source-successful attack frames remain successful after transfer. UniAD→GenAD reaches 12.3% TSR, while most other source-target pairs remain much weaker. In contrast, UniAD→VAD does not transfer under this protocol. These results suggest that the planning vulnerability exposed by JANUS is not isolated to one architecture; however, the specific perturbations found on one model do not reliably generalize to others.

This behavior is consistent with differences in model architecture and preprocessing. Transfers between VAD and UniAD are less affected by preprocessing mismatch than transfers involving GenAD, while resizing and normalization can still weaken perturbation structure in the reverse direction. We therefore do not position JANUS as a strong black-box transfer attack. Instead, this experiment shows that cross-model transfer among E2E ADSs is possible in some source-target pairs, but remains inconsistent and highly dependent on the source and target models.

7.7 Physical Experiments

We conduct a real-world experiment to evaluate the feasibility of the print-and-capture patch pipeline beyond digital insertion, rather than a complete real-world deployment study. We evaluate the GenAD model and capture an outdoor driving sequence with an Insta360 X3 camera [8]. We convert the 360-degree video into nuScenes six-camera inputs at a resolution of 1600×900 and 2 Hz. We provide a detailed description of the setup in Appendix E.

This physical experiment follows the core JANUS attack methodology. We first run GenAD on the clean real-world input to obtain the clean trajectory and temporal context. This clean pass provides the reference for the patch optimization. We use this reference to optimize a 100×100-pixel patch with the same route-conditioned planning objective as JANUS. This budget modifies about 2% of the total six-camera image area after GenAD’s preprocessing. It keeps the perturbation spatially localized but still large enough to survive camera capture, resize operations, and reprojection, and to resemble a physically plausible printed object such as a board or poster.



Figure 9: Real-world physical-patch result on GenAD. Captured physical-patch views are inserted into the clean temporal context, shifting the straight-command plan toward the selected target.

To ensure patch robustness against the print-and-capture pipeline, we apply Expectation over Transformation (EoT) during optimization. These transformations include size adjustments, brightness and contrast modifications, and pixel noise. We print the optimized patches on 89 cm × 89 cm boards and record a second sequence where the camera observes the printed patch. We position the printed boards at approximately the same camera-view locations from the three-camera patch setting in our digital evaluation.

Figure 9 presents the result for a representative frame. Under the straight command, the clean plan ends at (−0.382, 35.016) m. After the insertion of the captured physical-patch views, the endpoint shifts to (1.673, 36.254) m. This translation shifts the endpoint by (+2.055, +1.238) m and causes a collision with the selected target. The BEV overlay displays the shifted trajectory alongside the clean detection and map context. This visual representation provides qualitative evidence that the attack successfully targets the planning behavior without broad corruption of the observable outputs.

8 Discussion and Limitations

Scope of Architectural Coverage. We focus on multi-task E2E architectures in which planning and observable outputs are coupled through shared latent or task-level states at inference time. This includes systems where the planner may use latent context from perception, mapping, motion, or occupancy heads, while the decoded observable outputs remain available for monitoring. Two architecture families fall outside this scope. First, diffusion-based planners (e.g., DiffusionDrive [14]) generate trajectories through an iterative denoising process and require a different attack and preservation formulation. Second, parallelized architectures such as PARA-Drive [30] reduce runtime coupling between planning and other heads. In PARA-Drive, the planning head is trained with shared BEV features but operates independently of the perception and prediction heads after co-training. As a result, preserving observable outputs does not test the same planning-perception coupling question as in the shared-latent architectures in this paper.

Defense Implications. A direct defense is to monitor observable outputs such as object detections and map predictions. If an attack causes missed vehicles, wrong classes, or distorted lane boundaries, then such checks can flag the input before the plan is executed. However, our results show that this defense is incomplete. JANUS can induce unsafe trajectories while keeping observable outputs close to the clean inference. This means that perception and map consistency can serve as useful warning signals, but they cannot be treated as a proxy for planning safety.

A more promising direction is cross-validation between planning and perception. Instead of checking perception outputs alone, the system should verify whether the planned trajectory is consistent with the scene predicted by the perception heads. For example, a trajectory that intersects a detected vehicle, crosses a predicted lane boundary, or violates the drivable area should be rejected or passed to a fallback planner. In traditional modular ADSs, the planner consumes explicit perception outputs, which inherently couples planning decisions to the perceived scene. In modern E2E ADSs, planning and observable outputs may instead be decoded separately from shared latent features, so the final plan is not necessarily constrained by those observable outputs. E2E systems therefore require explicit planning-perception cross-validation before execution.

Real-world deployment. JANUS includes a physical-patch experiment with print-and-capture optimization, but this optimization is used as an experimental adaptation rather than as a core component of the attack formulation. In particular, the main methodology focuses on clean-reference anchoring, route-conditioned planning loss, perceptual consistency constraints, temporal state refresh, and patch placement. For the real-world study, we additionally apply transformations that approximate printing, camera capture, lighting variation, and resolution changes. This design helps assess whether the attack signal can survive a physical imaging pipeline, but it does not constitute a complete deployment model.

A full real-world deployment study would require broader optimization and evaluation across patch pose, distance, illumination, camera exposure, motion blur, vehicle motion, scene dynamics, and closed-loop feedback. Our experiment maintains temporal context control and evaluates the captured physical-patch views offline, thereby isolating the patch effect but not measuring end-to-end attack success on a moving vehicle. We therefore treat the physical result as evidence that JANUS is not limited to digital insertion, while leaving systematic real-world deployment to future work.

9 Conclusion

We introduce JANUS, a framework for analyzing planning attacks against multi-task E2E autonomous driving systems. JANUS uses clean-reference anchoring, a route-conditioned collision objective, and observable-output preservation constraints to test whether unsafe trajectories can be induced while observable outputs remain close to clean inference. Across GenAD, VAD, and UniAD on nuScenes, JANUS raises collision and off-road rates under our primary six-camera digital attack setting. At the same time, the largest relative drops in detection and map scores are 17.5% and 16.6%, respectively. These results show that shared latent representations do not by themselves guarantee consistency between observable

perception and planning: an E2E model can keep clean-looking observable outputs while still producing an unsafe trajectory.

Acknowledgment

We thank the anonymous reviewers for their valuable suggestions. This work has been supported in part by the National Science Foundation (NSF) under grants CNS-2144645, IIS-2229876, and 1941524; by the National Center for Transportation Cybersecurity and Resiliency (TraCR), a U.S. Department of Transportation National University Transportation Center, under awards 69A3552344812 and 69A3552348317; and through a gift from Qualcomm Technologies, Inc. The views presented in this paper are those of the authors only. We would like to thank Ananth Shreekumar for discussions and Han Dai for assistance with the physical experiments.

References

- [1] Baidu Apollo Team. 2024. Apollo: An open autonomous driving platform. <https://github.com/ApolloAuto/apollo>.
- [2] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. 2021. Invisible for both Camera and LiDAR: Security of Multi-Sensor Fusion based Perception in Autonomous Driving Under Physical-World Attacks. In *IEEE Symposium on Security and Privacy (S&P)*.
- [3] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Ramapazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. 2019. Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
- [4] Kevin Eykholt, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Physical Adversarial Examples for Object Detectors. In *USENIX Workshop on Offensive Technologies (WOOT)*.
- [5] Kevin Eykholt, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [6] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. 2023. Planning-Oriented Autonomous Driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [7] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. 2023. Planning-Oriented Autonomous Driving: Supplementary Material. Supplementary material for the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).
- [8] Insta360. 2026. Insta360 X3: 360 Action Camera Built for Creativity. <https://www.insta360.com/product/insta360-x3>. Accessed: 2026-04-30.
- [9] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. 2023. VAD: Vectorized Scene Representation for Efficient Autonomous Driving. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.
- [10] Wei Jiang, Lu Wang, Tianyuan Zhang, Yuwei Chen, Jian Dong, Wei Bao, Zichao Zhang, and Qiang Fu. 2024. RobustE2E: Exploring the Robustness of End-to-End Autonomous Driving. *Electronics* (2024).
- [11] Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114* (2013).
- [12] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. 2024. BEVFormer: Learning Bird's-Eye-View Representation From LiDAR-Camera via Spatiotemporal Transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).
- [13] Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. 2023. MapTR: Structured Modeling and Learning for Online Vectorized HD Map Construction. In *International Conference on Learning Representations (ICLR)*.
- [14] Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and Xinggang Wang. 2025. DiffusionDrive: Truncated Diffusion Model for End-to-End Autonomous Driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [15] Yanmao Man, Ming Li, and Ryan Gerdes. 2020. {GhostImage}: Remote Perception Attacks against Camera-based Image Classification Systems. In *International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*.
- [16] Yanmao Man, Ming Li, and Ryan Gerdes. 2024. Remote Perception Attacks against Camera-based Object Recognition Systems and Countermeasures. *ACM*

- Transactions on Cyber-Physical Systems* (2024).
- [17] Yanmao Man, Raymond Muller, Ming Li, Z. Berkay Celik, and Ryan Gerdes. 2023. That Person Moves Like A Car: Misclassification Attack Detection for Autonomous Systems Using Spatiotemporal Consistency. In *USENIX Security Symposium*.
 - [18] Raymond Muller, Yanmao Man, Z. Berkay Celik, Ming Li, and Ryan Gerdes. 2022. Physical Hijacking Attacks against Object Trackers. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
 - [19] Raymond Muller, Yanmao Man, Ming Li, Ryan Gerdes, Jonathan Petit, and Z. Berkay Celik. 2024. VOGUES: Validation of Object Guise using Estimated Components. In *USENIX Security Symposium*.
 - [20] Raymond Muller, Ruoyu Song, Chenyi Wang, Yuxia Zhan, Jean-Philippe Monteuis, Yanmao Man, Ming Li, Ryan Gerdes, Jonathan Petit, and Z. Berkay Celik. 2025. Investigating Physical Latency Attacks Against Camera-Based Perception. In *IEEE Symposium on Security and Privacy (S&P)*.
 - [21] Federico Nesti, Giulio Rossolini, Saasha Nair, Alessandro Biondi, and Giorgio Buttazzo. 2022. Evaluating the Robustness of Semantic Segmentation for Autonomous Driving Against Real-World Adversarial Patch Attacks. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.
 - [22] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. 2021. Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under {Physical-World} Attack. In *USENIX Security Symposium*.
 - [23] Junjie Shen, Jun Yeon Won, Zeyuan Chen, and Qi Alfred Chen. 2020. Drift with Devil: Security of {Multi-Sensor} Fusion based Localization in {High-Level} Autonomous Driving under {GPS} Spoofing. In *USENIX Security Symposium*.
 - [24] Tesla, Inc. 2026. Full Self-Driving Capability. <https://www.tesla.com/fsd>. Accessed: 2026-02-09.
 - [25] The Autoware Foundation. 2024. Autoware: The world's leading open-source software project for autonomous driving. <https://github.com/autowarefoundation/autoware>.
 - [26] Chenyi Wang, Yanmao Man, Raymond Muller, Ming Li, Z. Berkay Celik, Ryan Gerdes, and Jonathan Petit. 2024. Physical ID-Transfer Attacks against Multi-Object Tracking via Adversarial Trajectory. In *Annual Computer Security Applications Conference (ACSAC)*.
 - [27] Jian Wang, Fan Li, Song Lv, Lijun He, and Chao Shen. 2025. Physically Realizable Adversarial Creating Attack Against Vision-Based BEV Space 3D Object Detection. *IEEE Transactions on Image Processing* (2025).
 - [28] Lu Wang, Tianyuan Zhang, Yikai Han, Muiyang Fang, Ting Jin, and Jiaqi Kang. 2024. Attack End-to-End Autonomous Driving through Module-Wise Noise. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.
 - [29] Waymo. 2024. Introducing EMMA: End-to-End Multimodal Model for Autonomous Driving. <https://waymo.com/blog/2024/10/introducing-emma>. Accessed: 2026-02-09.
 - [30] Xinsuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. 2024. PARA-Drive: Parallelized Architecture for Real-time Autonomous Driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
 - [31] Yuan Xu, Gelei Deng, Xingshuo Han, Guanlin Li, Han Qiu, and Tianwei Zhang. 2024. PhysScout: Detecting Sensor Spoofing Attacks via Spatio-temporal Consistency. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
 - [32] Chenyu Yang, Yuntao Chen, Hao Tian, Chenxin Tao, Xizhou Zhu, Zhaoxiang Zhang, Gao Huang, Hongyang Li, Yu Qiao, Lewei Lu, Jie Zhou, and Jifeng Dai. 2023. BEVFormer v2: Adapting Modern Image Backbones to Bird's-Eye-View Recognition via Perspective Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
 - [33] Zhiyuan Yu, Ao Li, Ruoyao Wen, Yijia Chen, and Ning Zhang. 2024. PhysSense: Defending Physically Realizable Attacks for Autonomous Systems via Consistency Reasoning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
 - [34] Ye Yuan, Xinsuo Weng, Yanglan Ou, and Kris M. Kitani. 2021. AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.
 - [35] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* (2020).
 - [36] Qingzhao Zhang, Shaocheng Luo, Z. Morley Mao, Miroslav Pajic, and Michael K. Reiter. 2025. SoK: How Sensor Attacks Disrupt Autonomous Vehicles: An End-to-end Analysis, Challenges, and Missed Threats. *arXiv preprint arXiv:2509.11120* (2025).
 - [37] Tianyuan Zhang, Lu Wang, Jiaqi Kang, Xinwei Zhang, Siyuan Liang, Yuwei Chen, Aishan Liu, and Xianglong Liu. 2024. Module-wise Adaptive Adversarial Training for End-to-end Autonomous Driving. *arXiv preprint arXiv:2409.07321* (2024).
 - [38] Wenzhao Zheng, Ruiqi Song, Xianda Guo, Chenming Zhang, and Long Chen. 2024. GenAD: Generative End-to-End Autonomous Driving. In *European Conference on Computer Vision (ECCV)*. Springer.
 - [39] Jiayu Zou, Kun Tian, Zheng Zhu, Yun Ye, and Xingang Wang. 2024. DiffBEV: Conditional Diffusion Model for Bird's Eye View Perception. In *AAAI Conference on Artificial Intelligence*.

Table 7: Cosine similarity between loss gradients of different heads at the clean input, reported as mean $\pm$ SEM.

Model	$\rho(\text{plan}, \text{det})$	$\rho(\text{plan}, \text{map})$	$\rho(\text{det}, \text{map})$
UniAD	$+0.031 \pm 0.009$	-0.030 ± 0.008	-0.009 ± 0.006
VAD	-0.052 ± 0.024	$+0.021 \pm 0.020$	-0.005 ± 0.013
GenAD	$+0.006 \pm 0.021$	$+0.030 \pm 0.011$	$+0.062 \pm 0.016$

A Generative AI Usage

We used OpenAI ChatGPT, Google Gemini and Anthropic Claude to improve grammar, clarity, concision, and organization. We also used OpenAI Codex and Anthropic Claude Code for implementation assistance, including code completion, debugging, and refactoring. All AI-assisted edits and code changes were reviewed and verified by the authors for accuracy, originality, and correctness.

B Open Science

We will release an artifact containing the implementation of our JANUS pipeline for UniAD, VAD, and GenAD. The artifact will include the attack implementations, evaluation scripts, and configurations for reproducing the main results. The artifact will be available at <https://doi.org/10.6084/m9.figshare.33769768>.

C Ethical Considerations

This work studies security weaknesses in end-to-end autonomous driving systems through adversarial input-space attacks that induce unsafe planning while preserving observable outputs. The main stakeholders include system developers, road users, the research community, and the authors. While the results can support more rigorous robustness evaluation, monitoring, and defensive design, the attack methodology may also help adversaries understand how to manipulate E2E driving systems. We limit this risk by restricting experiments to public research models and offline evaluation, without tests on deployed vehicles, commercial systems, public infrastructure, human subjects, or private data. Our physical-patch experiment uses controlled offline view replacement and does not establish real-time or closed-loop attack feasibility. The released artifact focuses on reproducibility. We conclude that the defensive value of documenting this vulnerability outweighs the misuse risk under these constraints.

D Cosine Similarity

Table 7 reports the cosine similarity between input gradients from the planning, detection, and map heads. For each clean validation frame, we compute

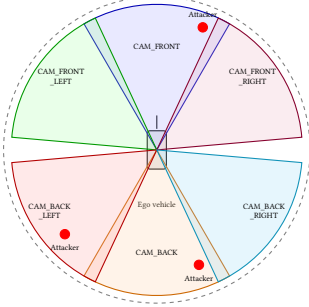
$$\rho(x) = \cos(\nabla_x \mathcal{L}_A, \nabla_x \mathcal{L}_B), \quad A, B \in \{\text{plan}, \text{det}, \text{map}\}.$$

Here, $\mathcal{L}_{\text{plan}}$ is the squared norm of the trajectory output. $\mathcal{L}_{\text{det}}$ and $\mathcal{L}_{\text{map}}$ are the squared norms of the top-20 decoded outputs from the detection and map heads. We report the mean and standard error of the mean over clean validation frames.

The results show that all three pairwise similarities are near zero across three ADSs. For example, the planning-detection similarity is $+0.031$ for UniAD, -0.052 for VAD, and $+0.006$ for GenAD. The planning-map similarity is also small: -0.030 for UniAD, $+0.021$ for



(a) Physical patch setup.



(b) Virtual camera layout.

Figure 10: Real-world physical-patch setup and 360°-to-nuScenes conversion. The printed patch is captured in an outdoor scene, and the 360° recording is converted into six virtual camera views for GenAD. Red markers indicate rough attacker positions.

VAD, and +0.030 for GenAD. The detection-map pair follows the same pattern, with values between -0.009 and $+0.062$.

These values suggest that the heads do not share a common dominant input-gradient direction at clean inputs. In other words, although planning, detection, and map prediction all depend on the shared latent representation, their first-order responses to input perturbations are not strongly aligned. This supports the main-text claim that shared latent features do not automatically force planning and observable outputs to change together.

This gradient-alignment analysis gives a local view of how the heads respond to input changes near clean inputs. The near-zero similarities suggest that planning, detection, and map prediction are not forced to follow the same input direction at first order. This helps explain why selective planning disruption is possible.

E Real World Experiment

Figure 10 illustrates the physical-patch setup and six-camera layout.

Offline Temporal State Control. The real-world experiment uses offline inference to isolate the effect of the captured physical patch on the current input. GenAD maintains temporal state across frames, so directly running a separate physical-patch video would mix several factors: the patch effect, camera-motion differences, frame alignment errors, and history-state drift. To avoid this confound, we first run the clean sequence and cache the clean temporal state. During physical-patch evaluation, we restore this clean temporal state and replace only the selected camera views with captured physical-patch views. This follows the same clean-reference design as JANUS: the attacked inference is compared against the model’s own clean trajectory and clean history, while the controlled change is the physical patch observed by the camera.

Print-and-Capture Robustness. To improve robustness to the physical pipeline, we apply EoT transformations during patch optimization. These transformations include bilinear down/up-sampling with scale sampled from $[0.33, 0.75]$, pixel noise with standard deviation sampled from $[2, 8]$, brightness shift within ± 15 , and contrast sampled from $[0.85, 1.15]$. These transformations approximate the resolution change, camera noise, lighting change, and reprojection effects introduced by printing, camera capture, and conversion into virtual camera views.

Offline View Replacement. After printing the optimized patch, we record a second outdoor sequence in which the physical patch is visible. We then evaluate the patch through offline view replacement. For the representative result in Figure 9, we replace CAM_FRONT, CAM_BACK, and CAM_BACK_LEFT in the clean temporal context with physical-patch views captured by the real camera. The other views remain clean. This protocol keeps the temporal context controlled while testing whether a printed patch observed by the real camera can still change the GenAD plan.

Observable Outputs. The real-world sequence lacks nuScenes-style ground-truth labels, so we do not report NDS or map mAP for this experiment. Instead, Figure 9 provides a qualitative check by overlaying the changed trajectory with the clean detection and map context. For the representative frame, the number of decoded detections above confidence 0.3 changes from 7 in the clean input to 5 after physical-patch replacement, while the number of decoded map elements above confidence 0.4 remains 3 in both cases. We therefore use this result only as qualitative support for the main observation that the visible effect is concentrated on planning.

Additional One-Frame Variants. We tested a few one-frame replacement variants during development. Their endpoint shifts ranged from about 0.20 m to 2.40 m. Because these variants are not sampled systematically across scenes, distances, lighting conditions, or viewpoints, we do not report them as a success rate. The main text therefore reports only the representative frame shown in Figure 9.