TEST-TIME ADAPTATION OF END-TO-END TRAJECTORY PLANNING VIA SIMULATION-BASED SELF-VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing Test-Time Adaptation (TTA) approaches mitigate distribution shifts for classification, detection, or segmentation, and the few for end-to-end planning update only the score decoder over a fixed trajectory vocabulary. More critically, the trajectory predicted under corruption is executed without refinement or verification, although a single unsafe one can cause a safety incident. We propose Simulation-based TTA (SimTTA), a verification-driven framework that addresses two key challenges. To verify the predicted trajectory, Geometric Trajectory Refinement (GTR) expands it in the output space into candidates beyond the reach of the corrupted model. Simulation-based Self-Verification (SSV) then simulates each candidate on a scene context reconstructed from the perception outputs of the model itself, and selects the highest-scoring one as the final trajectory without ground-truth labels or training. To adapt the planning model, the Temporally-Verified Entropy (TVE) loss minimizes the prediction entropy only on the BEV cells whose predicted classes remain the same across consecutive samples. Thus, the following samples are planned and verified with a more reliable representation. Experiments across eight corruption types demonstrate that our SimTTA significantly enhances trajectory quality under such shifts. The source code will be released.

1 INTRODUCTION

End-to-end autonomous driving has emerged as a promising paradigm in which models directly learn driving policies from raw sensor inputs, building upon advances in upstream perception tasks such as object detection (Chen et al., 2025a; Huang et al., 2021; Wang et al., 2022b), object tracking (Zeng et al., 2022; Zhang et al., 2021; 2022), and online mapping (Liao et al., 2022; 2025b; Liu et al., 2023). Among downstream tasks, trajectory planning has gained significant attention as the core component that translates scene understanding into safe driving behavior, predicting the sequence of future waypoints that the vehicle will follow (Jiang et al., 2023; Liao et al., 2025a), and recent end-to-end planners have achieved strong performance on real-world datasets such as nuScenes (Caesar et al., 2020), KITTI (Geiger et al., 2012), OpenScene (Yang et al., 2024b), and nuPlan (Caesar et al., 2021).

Despite these recent advancements, planners trained under clean sensor conditions can produce unsafe trajectories, such as colliding with surrounding vehicles or departing from the lane. Such failures arise from distribution shifts caused by camera noise, adverse weather, or field-of-view changes. In autonomous driving, planner degradation can directly lead to safety incidents, while these unpredictable corruptions can occur at any time during deployment. The ability to immediately adapt to such shifts without offline retraining is therefore critical for safe autonomous driving.

Test-time adaptation (TTA) has been explored to mitigate these distribution shifts without retraining on test-domain data, through optimization signals such as entropy minimization (Wang et al., 2021; Niu et al., 2023), self-supervised learning (Wang et al., 2022a), and weight-space manipulation (Yang et al., 2025a; Liao et al., 2026). Yet, these methods mainly target classification, detection, and segmentation rather than trajectory planning. The closest work adapts an end-to-end planner by minimizing the entropy over clustered trajectory candidates (Sima et al., 2025), but it does not address sensor corruption, leaving TTA for trajectory planning under corruption largely unexplored.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

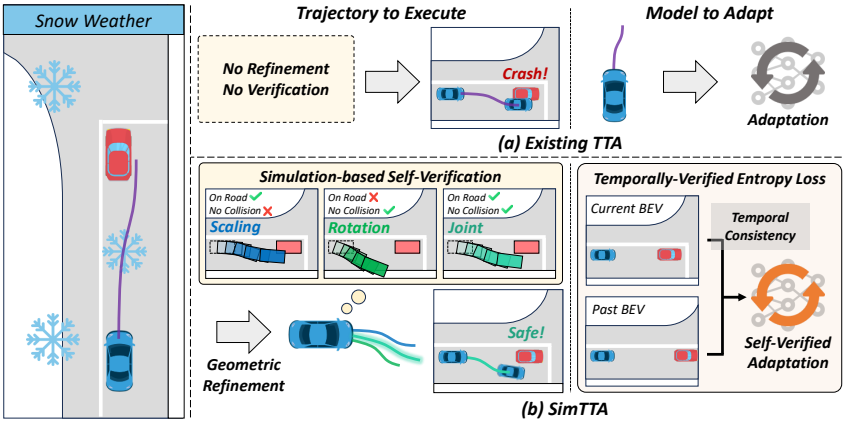


Figure 1: Conceptual comparison of (a) existing TTA methods, which adapt the model to the corrupted input distribution but produce a single trajectory without verifying its future consequences, and (b) our SimTTA, which generates multiple trajectory candidates and selects the highest-scoring one based on a simulation-based self-verification signal obtained from the perception outputs of the model.

A more critical limitation is that existing TTA methods directly use the trajectory predicted under corruption without refining or verifying it. As shown in Figure 1(a), they use the prediction on the current sample to adapt the model for the following samples, but the trajectory produced on that corrupted sample is executed without any refinement. This trajectory may cause planning-relevant violations (*e.g.*, collisions, drivable area departures, unsafe accelerations), yet none of these is checked before it is executed. In autonomous driving, a trajectory is an actual plan that the vehicle executes in real-world deployment, so a single unverified trajectory can lead to a safety incident.

To overcome these limitations, we devise Simulation-based TTA (SimTTA) to address two key challenges: (*i*) how to **refine and verify the current trajectory** predicted under corruption so that it can be executed safely, and (*ii*) how to **adapt the model itself** so that the following samples are planned and verified more reliably under corruption.

To address challenge (*i*), we introduce Geometric Trajectory Refinement (GTR), which refines the predicted trajectory through direct perturbations in the output space. Under corruption, the planner can produce distorted or unsafe trajectories with limited alternatives for recovery. GTR therefore expands the prediction along two geometric directions, *trajectory scaling and rotation*, to generate refined candidates that the model cannot predict on its own. We then introduce Simulation-based Self-Verification (SSV) to verify which candidate is safe and suitable for execution. SSV reconstructs the future scene context from the perception outputs of the planner and simulates each candidate within this context by scoring its safety and driving quality without any external knowledge. The candidate with the highest verification score is selected as the final trajectory.

To address challenge (*ii*), we introduce Temporally-Verified Entropy (TVE) loss, which adapts the bird's-eye-view (BEV) representation of the planner using the temporal consistency of its predictions. GTR and SSV alone leave the corrupted model unchanged, so the same failure recurs on every sample. TVE loss therefore updates the model by minimizing the prediction entropy only on the BEV cells whose predicted classes remain the same across consecutive samples, so that the following samples start from a more reliable BEV representation. Since SSV also relies on this representation, the adaptation improves both the trajectory and its verification.

As a result, SimTTA addresses deployment-time distribution shifts by verifying future consequences of candidate trajectories through simulation (see Figure 1(b)). It also remains effective under mild or no shift, as SSV can still select a better candidate when the original prediction is already near optimal.

The major contributions can be summarized as follows:

- We propose a new Test-Time Adaptation (TTA) framework for safe autonomous driving that performs simulation-based self-verification of predicted trajectories.
- We introduce Geometric Trajectory Refinement (GTR), which expands the predicted trajectory into candidates in the output space, and Simulation-based Self-Verification (SSV), which scores each candidate through simulation on the perception outputs of the model.

- We develop the Temporally-Verified Entropy (TVE) loss, which adapts the BEV representation of the planner to sensor corruption using the temporal consistency of its predictions, improving both the trajectory and its verification on the following samples.
- Extensive experiments on OpenScene and nuScenes-C demonstrate consistent improvements under all eight corruption types of each benchmark, outperforming existing TTA methods.

2 Related Work

2.1 End-To-End Autonomous Driving

End-to-end autonomous driving jointly optimizes perception (Li et al., 2024b; Wang et al., 2023; Chen et al., 2025a), prediction (Shi et al., 2022; Zhou et al., 2023), and planning (Cheng et al., 2024b;a) in a single differentiable framework, evolving from direct trajectory regression (Prakash et al., 2021; Hu et al., 2022; Jia et al., 2023a;b) to multi-task architectures (Hu et al., 2023), vectorized (Jiang et al., 2023) and sparse (Sun et al., 2025) scene encodings, anchor vocabularies (Chen et al., 2024a; Li et al., 2024a), and diffusion-based policies (Liao et al., 2025a). These planners are developed under clean sensor conditions and degrade under real-world distribution shifts, motivating adaptation at test time.

2.2 Test-time Adaptation

Test-time adaptation (TTA) adapts a pretrained model to an unseen target distribution using unlabeled test data (Liang et al., 2025). TENT (Wang et al., 2021) minimizes prediction entropy to update the BN affine parameters, and subsequent methods stabilize this signal on noisy streams (Niu et al., 2023), replace it with consistency to a teacher or deteriorated view (Wang et al., 2022a; Niu et al., 2025), merge checkpoints (Chen et al., 2025b; Yang et al., 2025a), or extend TTA to regression (Adachi et al., 2025), while in driving perception monocular 3D detectors are adapted with score-based objectives (Lin et al., 2024) and auxiliary normalization parameters (Liao et al., 2026). Centaur (Sima et al., 2025) adapts an end-to-end planner over clustered trajectory candidates, but requires a discrete trajectory vocabulary and updates only the score decoder rather than the backbone, where sensor corruption first degrades the representation. None of these signals verifies trajectory safety, whereas SimTTA verifies planning quality through simulation.

2.3 Trajectory Evaluation and Selection at Inference

A growing line of work generates trajectory candidates and selects among them through an evaluation step. World models (Li et al., 2025; Yang et al., 2025b; Zheng et al., 2025; Zhang et al., 2025) roll out future scene states to score candidates, Hydra-MDP (Li et al., 2024a) ranks them with scoring heads distilled from expert teachers, Diffusion-ES (Yang et al., 2024a) searches a trajectory diffusion model against a rule-based reward, and other work refines the trajectory against a learned scorer (Xu et al., 2026) or verifier (He et al., 2026). These scoring modules are learned on clean data or read privileged scene state unavailable to a sensor-based planner, whereas the simulator of SimTTA runs on the perception outputs of the planner itself and requires no training and no privileged input.

3 Method

For each test sample t, SimTTA takes multi-view camera images, LiDAR point clouds, the HD map, and the ego state as input (Figure 2), and predicts a trajectory $\tau \in \mathbb{R}^{T\times 3}$ of T future waypoints with position (x, y) and heading ψ. Sensor corruption can degrade the trajectory, perception outputs, and BEV predictions. SimTTA addresses this with three components: Geometric Trajectory Refinement (GTR) expands τ into a candidate set beyond the reach of the corrupted model (Section 3.1), Simulation-based Self-Verification (SSV) simulates each candidate and selects the highest-scoring one for execution (Section 3.2), and the Temporally-Verified Entropy (TVE) loss adapts the BN affine parameters on temporally verified cells for the following samples (Section 3.3).

3.1 Geometric Trajectory Refinement (GTR)

The predicted trajectory τ serves as the starting point of the candidate search, but under corruption it can exhibit systematic geometric errors that the model may not reliably correct on its own. We therefore introduce Geometric Trajectory Refinement (GTR), which perturbs τ directly in the output space to generate diverse candidates beyond those produced by the model itself.

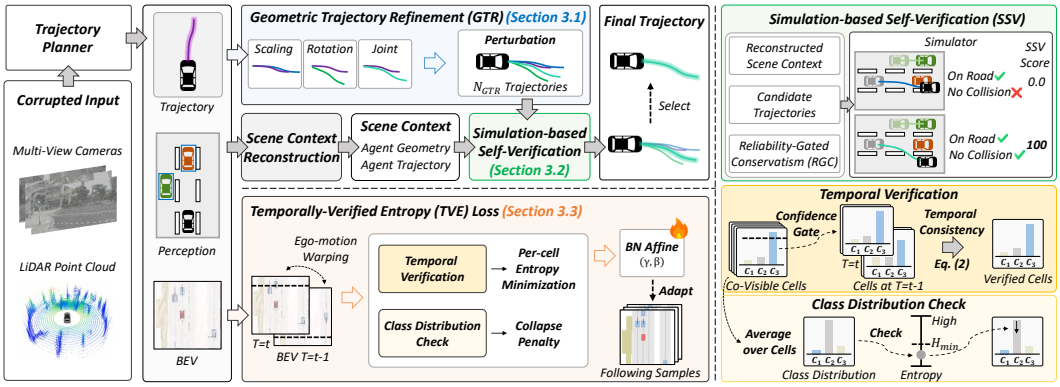


Figure 2: Overall architecture of our SimTTA. GTR perturbs the predicted trajectory along scale, rotation, and joint branches. SCR builds the reconstructed scene context $\tilde{c}_t$ from the perception outputs, on which SSV produces the score $\tilde{S}$ per candidate. RGC adjusts the scores by the perception reliability, and the highest-scoring candidate becomes the final trajectory. TVE loss minimizes the prediction entropy only on the BEV cells that pass the temporal verification and penalizes a collapse of the class distribution averaged over all cells, updating the BN affine parameters at test time.

To determine the most relevant directions for trajectory refinement, we conduct an empirical study under corruption (Section 4.5). Using PDMS (Appendix A.7) as the trajectory quality metric, we find that approximately 97% of its variation is explained by two geometric factors: trajectory **scaling** and **rotation**. Based on this observation, GTR perturbs τ along each direction and jointly. *(i)* **Scale branch** samples $s \sim \mathcal{N}(1, \sigma_s^2)$ and scales all waypoint positions (x, y) by s, producing N_s candidates. *(ii)* **Rotation branch** samples $r \sim \mathcal{N}(0, \sigma_r^2)$, rotates all waypoint positions by r, and adjusts their headings ψ accordingly, producing N_r candidates. *(iii)* **Joint branch** samples (s, r) from the same distributions and applies both transformations, producing N_j candidates. Detailed hyperparameters and clipping ranges are in Appendix A.9. We set N_s, N_r, N_j to the same value, denoted as N_{PTB}.

The candidates from the three branches and τ itself constitute the candidate set $\mathcal{C}$, and we write $\hat{\tau}$ for a trajectory in $\mathcal{C}$. We retain τ in $\mathcal{C}$ so that the selection can fall back on the prediction when no perturbation proves safer. The set thus contains $N_{\text{GTR}} = 1+N_s+N_r+N_j$ trajectories. Every $\hat{\tau} \in \mathcal{C}$ is scored through simulation (Section 3.2), and the highest-scoring one is selected as the final output.

3.2 Simulation-based Self-Verification (SSV)

GTR generates N_{GTR} candidate trajectories, but the planner itself provides no direct signal indicating which candidate should be executed. We therefore introduce Simulation-based Self-Verification (SSV) to evaluate each candidate through simulation for trajectory selection. Specifically, we adopt PDMS S as the simulation score, as it evaluates driving safety and quality in a scene context c_t through collision avoidance, drivable-area compliance, progress, and comfort (Dauner et al., 2024). However, directly computing PDMS is infeasible: while the HD map and ego state in c_t are available, the agent geometry (position, heading, and box shape) and trajectories require ground-truth annotations.

To approximate the unavailable ground-truth scene context at deployment, we introduce Scene Context Reconstruction (SCR), which reconstructs the agent information from the perception outputs, yielding $\tilde{c}_t$. SCR obtains the geometry of each agent i from detections and estimates its velocity $\mathbf{v}^{(i)}$ by matching detections across consecutive samples (Appendix A.8). Since the goal of SSV is to select the best candidate, which requires only the relative ranking rather than accurate absolute scores, we extrapolate the future trajectory of each agent assuming constant velocity over $H = 51$ steps:

$$\mathbf{p}_{t+h}^{(i)} = \mathbf{p}_t^{(i)} + \mathbf{v}^{(i)} \cdot h\Delta t_{\text{sim}}, \quad h \in \{0, 1, \ldots, H-1\}, \tag{1}$$

where $\mathbf{p}_t^{(i)}$ is the detected position of agent i and Δt_{sim} is the simulation step size, with the heading and box shape kept constant. These reconstructed agents complete $\tilde{c}_t$, in which SSV simulates each candidate $\hat{\tau}$ and scores it in the PDMS form as the SSV score $\tilde{S}$:

$$\tilde{S}(\hat{\tau}, \tilde{c}_t) = s_{\text{NC}} \cdot s_{\text{DAC}} \cdot \frac{5\, s_{\text{TTC}} + 5\, s_{\text{EP}} + 2\, s_{\text{C}}}{12}, \tag{2}$$

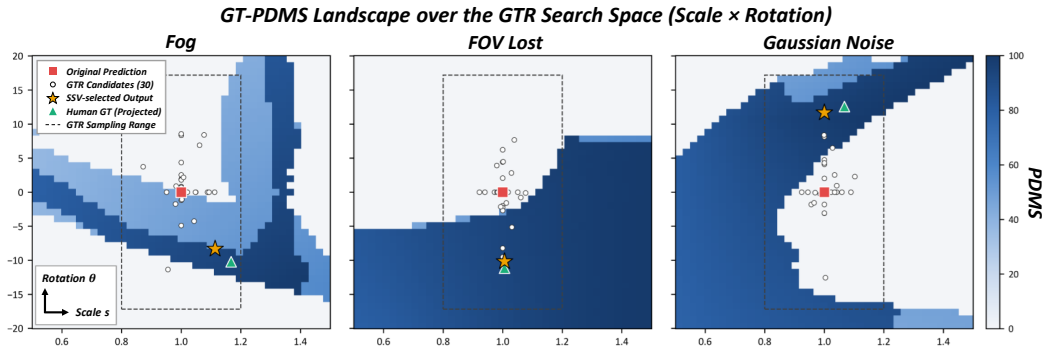


Figure 3: PDMS landscape over the GTR search space. Each candidate (white dots) obtained by scaling and rotating is mapped onto a diverse PDMS space, and the trajectory that is the best in terms of safety (yellow star) is selected among them, which lies close to the human GT (green triangle).

where s_{NC} and s_{DAC} are the no-collision and drivable area compliance terms that zero out the score upon a violation, while s_{TTC}, s_{EP}, and s_{C} measure time-to-collision, ego progress, and comfort, with all sub-scores computed on the reconstructed scene context $\tilde{c}_t$.

Table 1: Agreement between the SSV score $\tilde{S}$ and the PDMS S over the candidate set $\mathcal{C}$ of each sample.

	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{FL}$	$\mathcal{G}$	$\mathcal{MO}$	$\mathcal{MB}$	$\mathcal{T}$	**Mean**
$\rho\uparrow$	0.71	0.75	0.75	0.76	0.67	0.75	0.76	0.73	**0.74**
$r\uparrow$	0.72	0.74	0.75	0.74	0.68	0.74	0.74	0.71	**0.73**
Top-1↑	73.5	76.8	70.6	83.8	70.8	77.4	76.6	74.6	**75.5**
Top-5↑	89.2	91.1	87.4	94.1	88.3	92.0	91.2	89.8	**90.5**
Regret↓	8.75	6.24	6.69	6.86	11.07	6.19	6.28	8.40	**7.56**

To verify that the SSV score is a reliable substitute for the PDMS at selection time, Table 1 measures how closely this simulated safety matches the actual PDMS. Across the eight corruption types the estimated SSV score $\tilde{S}$ and the PDMS S correlate at an average Spearman ρ of 0.74 and Pearson r of 0.73. We also rank the selected candidate within $\mathcal{C}$ under the PDMS, where Top-k is the fraction of samples on which it ranks within the best k and Regret is its average PDMS gap to the best candidate. SSV reaches 75.5% Top-1 and 90.4% Top-5, against 3.2% and 16.1% for a candidate picked at random, at a regret of 7.56 PDMS points. It indicates that SSV picks a near-optimal candidate on most samples, and even when it misses the best candidate, the executed trajectory stays close to it in safety and driving quality.

This analysis confirms that SSV simulates the actual safety of a given trajectory without any ground-truth scene context. Figure 3 visualizes this agreement, where every point of the plane is a trajectory obtained by scaling and rotating the prediction, colored by its PDMS. The prediction of the model (red square) tends to lie away from the high-scoring region that the GTR candidates (white dots) reach, and the candidate that SSV selects (yellow star) lands inside it, close to the human driver (green triangle) that neither GTR nor SSV observes. See Appendix A.11 for all corruption types.

Reliability-Gated Conservatism (RGC). The reconstructed scene context $\tilde{c}_t$ inherits the perception outputs of the model, so under severe corruption the SSV score $\tilde{S}$ itself becomes unreliable, and a safe policy is to avoid aggressive candidates rather than to trust the score. To this end, we measure the reliability of the perception by the temporal agreement $a_t \in [0, 1]$, the fraction of the confidently predicted BEV cells whose class stays the same across two consecutive samples after compensating for the ego motion (Section 3.3). As a_t stays close to one on most samples, we set an agreement floor a_0 below which the perception is regarded as unreliable, and rescale $[a_0, 1]$ onto $[0, 1]$ with a gate $\lambda_t = \text{clip}\left((a_t - a_0)/(1 - a_0), 0, 1\right)$. RGC then penalizes the candidates faster than the prediction:

$$\tilde{S}_{\text{RGC}}(\hat{\tau}) = \tilde{S}(\hat{\tau}, \tilde{c}_t) - w\,(1 - \lambda_t)\max(0,\, s_{\hat{\tau}} - 1), \tag{3}$$

where $s_{\hat{\tau}}$ is the scale factor s that GTR applied to $\hat{\tau}$ and w controls the penalty strength. The penalty vanishes on reliable samples and suppresses aggressive candidates otherwise.

The candidate with the highest reliable score in the candidate set $\mathcal{C}$ serves as the final output $\hat{\tau}^*$:

$$\hat{\tau}^* = \arg\max_{\hat{\tau} \in \mathcal{C}} \tilde{S}_{\text{RGC}}(\hat{\tau}). \tag{4}$$

The vehicle finally executes $\hat{\tau}^*$. The selection, however, is limited by the corrupted model, since both the candidates and $\tilde{c}_t$ come from its outputs. Adapting the model itself makes both more reliable on the following samples, as described in Section 3.3.

3.3 TEMPORALLY-VERIFIED ENTROPY (TVE) LOSS

Under corruption the BEV representation of the planner degrades, and since every sample passes through the same model, the degradation persists along the stream. SimTTA therefore adapts the BN affine parameters at test time. Minimizing the entropy of the BEV prediction on confident cells is a natural objective, yet corruption can make the model confidently wrong and naive entropy minimization reinforces such errors. The proposed Temporally-Verified Entropy (TVE) loss therefore updates the model using only the cells verified by temporal consistency.

Let p_t be the BEV class probability map predicted on sample t, which assigns a class distribution to each BEV cell u, and $c_t(u)$ its largest class probability, the cell confidence. Let $\mathcal{V}_t$ be the co-visible cells observed on both samples t and $t-1$. A confidence gate keeps the confident ones among them:

$$\mathcal{K}_t = \{\, u \in \mathcal{V}_t \mid c_t(u) \geq \delta \,\}, \tag{5}$$

where δ is a confidence threshold. Since the static scene structure does not change between consecutive samples, the previous prediction warped into the current ego frame should agree with the current prediction. With $\hat{y}_t(u)$ the class predicted at cell u and $\hat{y}_{t-1\to t}(u)$ the class of the warped prediction at the same cell, the verified cells Ω_t are those whose classes agree:

$$\Omega_t = \{\, u \in \mathcal{K}_t \mid \hat{y}_t(u) = \hat{y}_{t-1\to t}(u) \,\}. \tag{6}$$

On these verified cells, the loss minimizes the per-cell prediction entropy:

$$\mathcal{L}_{\text{ent}} = \frac{1}{|\Omega_t|} \sum_{u\in\Omega_t} H\big(p_t(u)\big), \tag{7}$$

where $H(p) = -\sum_k p_k \log p_k$ is the entropy of a class distribution. Entropy minimization alone, however, can collapse to a single dominant class everywhere, so we add a penalty that activates only when the entropy of the class distribution $\bar{p}_t$ averaged over all cells falls below an entropy floor $H_{\min}$:

$$\mathcal{L}_{\text{TVE}} = \mathcal{L}_{\text{ent}} + \eta \max\big(0,\, H_{\min} - H(\bar{p}_t)\big), \tag{8}$$

where η sets the strength of the penalty. The warped prediction enters only as a binary mask and carries no gradient, so earlier prediction errors do not propagate into the update. Backpropagation reaches only the BN affine parameters (BN statistics in Appendix A.9), and TVE loss reuses the perception forward pass at no additional cost. The ratio of the verified cells to the confident cells $\mathcal{K}_t$, $a_t = |\Omega_t|/|\mathcal{K}_t|$, is the agreement that the reliability gate uses (Section 3.2).

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate SimTTA on OpenScene (Yang et al., 2024b) and nuScenes-C (Xie et al., 2025). OpenScene, derived from nuPlan (Caesar et al., 2021), provides synchronized multi-view cameras, LiDAR, ego states, and HD maps under diverse driving conditions, while nuScenes-C is a corrupted version of the nuScenes (Caesar et al., 2020) validation set with six-view camera images. For consistent trajectory planning evaluation, we follow the NAVSIM (Dauner et al., 2024) closed-loop protocol using the navtest split, containing 12,146 OpenScene samples and 6,019 corrupted nuScenes-C samples. Unless otherwise specified, we use OpenScene as the primary evaluation benchmark, while nuScenes-C results are reported in Section 4.2 and Appendix A.11.

Corruptions. Following (Dong et al., 2023), we apply eight corruption types to OpenScene to evaluate robustness under deployment-time distribution shifts. These corruptions cover four categories: weather (Snow, Rain, Fog), sensor (FOV Lost, Gaussian Noise), motion (Moving Object, Motion Blur), and alignment (Temporal Misalignment). For nuScenes-C, we use its eight natural corruptions, including image degradation (Motion Blur, Color Quantization), illumination shift (Dark, Brightness), adverse weather (Snow, Fog), and sensor failure (Camera Crash, Frame Lost). Unless otherwise specified, all results are evaluated at the highest severity level following (Wang et al., 2021; Chen et al., 2025b; Yang et al., 2025a; Liao et al., 2026), severity 5 for OpenScene and hard for nuScenes-C. Results across severity levels and compound corruptions are provided in Appendices A.4 and A.5.

Pre-trained Base Planners. As a model-agnostic method, SimTTA can be applied to different end-to-end planners. We consider three representative planners: DiffusionDrive (Liao et al., 2025a),

Table 2: Comparison of PDMS on the NAVSIM navtest split built on OpenScene. We report PDMS as mean±std for each of the eight corruption types and the mean across all corruptions, all averaged over three random seeds (Snow ($\mathcal{S}$), Rain ($\mathcal{R}$), Fog ($\mathcal{F}$), FOV lost ($\mathcal{FL}$), Gaussian noise ($\mathcal{G}$), Moving object ($\mathcal{MO}$), Motion blur ($\mathcal{MB}$), and Temporal misalignment ($\mathcal{T}$)).

Methods	Weather			Sensor		Motion		Alignment	Mean
	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{FL}$	$\mathcal{G}$	$\mathcal{MO}$	$\mathcal{MB}$	$\mathcal{T}$	
Source (*No Adapt.*)	70.7	84.7	81.2	71.8	73.5	87.9	87.2	74.5	78.9
BN Adapt.	82.7±0.06	84.4±0.06	83.5±0.06	68.9±0.06	78.0±0.06	84.9±0.00	83.1±0.00	72.8±0.06	79.8
Tent(ICLR'21)	82.0±0.40	83.9±0.29	82.4±0.17	65.1±0.87	76.9±0.35	84.5±0.06	82.1±0.31	72.6±0.26	78.7
MonoTTA(ECCV'24)	80.7±0.09	83.2±0.05	82.0±0.09	67.2±0.05	76.4±0.05	83.8±0.05	82.0±0.05	71.2±0.08	78.3
MOS(ICLR'25)	68.0±1.85	82.7±0.55	79.5±0.15	70.4±0.15	71.8±3.21	85.6±1.04	84.7±0.15	73.3±0.26	77.0
SSA(ICLR'25)	82.1±0.08	84.7±0.05	84.2±0.05	69.9±0.00	78.2±0.05	85.2±0.00	83.6±0.05	72.7±0.12	80.1
SPA(ICML'25)	79.9±1.60	82.1±0.29	80.3±0.16	60.2±4.22	74.1±2.00	83.1±0.17	79.4±0.31	71.5±0.76	76.3
CodeMerge(NeurIPS'25)	70.8±0.06	84.7±0.00	81.2±0.06	71.8±0.12	73.3±0.23	87.9±0.06	87.2±0.00	74.5±0.10	78.9
LearnableBN(AAAI'26)	70.9±0.10	85.2±0.26	81.5±0.21	71.9±0.10	73.5±0.12	87.9±0.06	87.1±0.06	74.5±0.06	79.0
SimTTA (Ours)	**85.1**±0.01	**88.2**±0.01	**87.4**±0.07	**82.7**±0.07	**81.3**±0.10	**88.4**±0.04	**87.8**±0.05	**82.1**±0.04	**85.4**

Table 3: Comparison of PDMS on the nuScenes-C benchmark. We report PDMS as mean±std for each corruption type and the mean across all corruptions, all averaged over three random seeds (Motion Blur ($\mathcal{MB}$), Color Quantization ($\mathcal{CQ}$), Dark ($\mathcal{D}$), Bright ($\mathcal{B}$), Snow ($\mathcal{S}$), Fog ($\mathcal{F}$), Camera Crash ($\mathcal{CC}$) and Frame Lost ($\mathcal{FL}$)).

Methods	Image Degradation		Illumination Shift		Adverse Weather		Sensor Failure		Mean
	$\mathcal{MB}$	$\mathcal{CQ}$	$\mathcal{D}$	$\mathcal{B}$	$\mathcal{S}$	$\mathcal{F}$	$\mathcal{CC}$	$\mathcal{FL}$	
Source (*No Adapt.*)	87.0	86.8	87.0	87.1	87.6	86.6	85.8	86.0	86.7
BN Adapt.	87.5±0.00	87.6±0.00	87.4±0.00	87.6±0.00	87.3±0.00	87.4±0.00	86.2±0.00	86.3±0.00	87.2
Tent(ICLR'21)	88.2±0.00	88.7±0.00	87.8±0.00	88.4±0.00	87.9±0.00	88.3±0.00	86.4±0.70	87.4±0.30	87.9
MonoTTA(ECCV'24)	85.4±0.06	84.7±0.06	85.0±0.07	85.2±0.06	84.9±0.06	85.1±0.03	85.4±0.06	84.0±0.25	85.1
MOS(ICLR'25)	72.7±2.20	74.3±3.80	74.3±0.90	73.1±1.40	72.4±4.90	71.7±1.90	78.9±2.70	71.4±1.60	73.6
SSA(ICLR'25)	85.5±0.02	84.7±0.03	84.8±0.00	85.0±0.03	84.8±0.03	84.8±0.02	84.0±0.08	84.4±0.18	84.8
SPA(ICML'25)	85.4±0.06	84.5±0.06	84.5±0.04	85.0±0.26	84.7±0.24	85.0±0.12	80.5±2.50	83.4±0.79	84.1
CodeMerge(NeurIPS'25)	70.7±2.90	71.5±7.00	69.7±1.30	72.5±5.90	72.6±4.50	67.8±2.30	59.7±2.80	66.8±6.00	68.9
LearnableBN(AAAI'26)	87.1±0.00	87.0±0.00	87.2±0.00	87.3±0.00	87.8±0.00	86.7±0.00	85.9±0.00	86.2±0.00	86.9
SimTTA (Ours)	**91.7**±0.17	**90.7**±0.20	**91.5**±0.13	**91.9**±0.12	**91.4**±0.15	**92.1**±0.19	**87.7**±0.40	**88.7**±0.20	**90.7**

TransFuser (Prakash et al., 2021), and UniAD (Hu et al., 2023). Among them, we adopt the recent DiffusionDrive as our default baseline planner due to its strong planning performance. Results on TransFuser and UniAD are provided in Appendix A.3.

Baseline TTA Methods. We compare our SimTTA against eight existing TTA methods spanning recent venues: BN Adapt. (Schneider et al., 2020), Tent (ICLR'21) (Wang et al., 2021), MonoTTA (ECCV'24) (Lin et al., 2024), MOS (ICLR'25) (Chen et al., 2025b), SSA (ICLR'25) (Adachi et al., 2025), SPA (ICML'25) (Niu et al., 2025), CodeMerge (NeurIPS'25) (Yang et al., 2025a), and LearnableBN (AAAI'26) (Liao et al., 2026). The Source baseline ('Source') corresponds to the pre-trained base planner without any adaptation. For all baselines, we follow their official implementations and apply them under the same streaming protocol as our method.

4.2 Comparison with Previous Methods

Results on OpenScene. Table 2 shows the results on the NAVSIM (Dauner et al., 2024) navtest split built on OpenScene (Yang et al., 2024b), averaged over three random seeds. SimTTA achieves a mean PDMS of 85.4 and improves over the 'Source' (no adaptation) on all eight corruption types and outperforms the strongest baseline SSA (Adachi et al., 2025) by 5.3 points. In contrast, four of the eight baselines fall below the Source on average. These methods optimize objectives defined on the corrupted input, such as prediction entropy or feature alignment, which do not directly reflect whether the resulting trajectory is safe. SimTTA overcomes this limitation by selecting the candidate with the highest simulated score and adapting the model only on temporally verified predictions.

Results on nuScenes-C. Table 3 reports the PDMS on the nuScenes-C benchmark (Xie et al., 2025), averaged over three random seeds, where the scene composition and sensor configurations differ from NAVSIM. SimTTA reaches a mean PDMS of 90.7 and is the best on all eight corruption types,

Table 4: Ablation studies of the proposed components. GTR + SSV denotes our simulation-based candidate selection, and $\mathcal{L}_{\text{TVE}}$ denotes the Temporally-Verified Entropy loss, which alone already outperforms every baseline in Table 2.

GTR + SSV	$\mathcal{L}_{\text{TVE}}$	Weather			Sensor		Motion		Alignment	Mean
		$\mathcal{S}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{FL}$	$\mathcal{G}$	$\mathcal{MO}$	$\mathcal{MB}$	$\mathcal{T}$	
-	-	70.7	84.7	81.2	71.8	73.5	87.9	87.2	74.5	78.9
✓	-	78.1	87.0	83.8	83.9	80.8	88.6	88.2	82.0	84.1
-	✓	82.4	86.3	85.3	70.2	77.4	86.6	85.2	73.8	80.9
✓	✓	**85.0**	**88.2**	**87.5**	**82.8**	**81.4**	**88.4**	**87.8**	**82.1**	**85.4**

Table 5: Effect of hyperparameters in SimTTA. We report the impact of (a) reliability floor (a_0), (b) conservatism weight (w), and (c) GTR perturbations per branch (N_{PTB}). **Bold** font indicates default configuration of our SimTTA.

(a) Effect of a_0.

a_0	Avg.PDMS
0.85	85.2
0.90	85.3
0.95	**85.4**
0.99	85.4

(b) Effect of w.

w	Avg.PDMS
1	85.2
3	85.3
5	**85.4**
7	85.4

(c) Effect of N_{PTB}.

N_{PTB}	Avg.PDMS	FPS
1	83.1	18.4
5	85.0	12.8
10	**85.4**	**9.4**
15	85.4	7.3

Table 6: Efficiency comparison results.

Method	GPU Mem.	FPS	Avg.PDMS
Source (*No Adapt.*)	0.8 GB	45.1	78.9
BN Adapt.	0.8 GB	32.7	79.8
Tent(ICLR'21)	1.1 GB	9.82	78.7
MonoTTA(ECCV'24)	1.6 GB	8.22	78.3
MOS(ICLR'25)	2.2 GB	0.91	77.0
SSA(ICLR'25)	2.1 GB	8.86	80.1
SPA(ICML'25)	1.9 GB	4.33	76.3
CodeMerge(NeurIPS'25)	2.2 GB	1.21	78.9
LearnableBN(AAAI'26)	1.3 GB	8.80	79.0
SimTTA (Ours)	1.4 GB	9.43	85.4

2.8 points above the strongest baseline Tent (Wang et al., 2021), while five baselines fall below the 'Source'. These results show that SimTTA remains effective across datasets with different scene compositions and sensor configurations.

4.3 Ablation Study

We conduct ablation studies to analyze the effect of the (*i*) proposed module, and (*ii*) hyperparameters. We use DiffusionDrive as the baseline and evaluate on the NAVSIM navtest split.

Effect of the Proposed Modules. Table 4 shows the effect of GTR, SSV, and the TVE loss, where adding each component consistently improves performance and the full SimTTA achieves the highest result. Notably, the TVE loss alone reaches 80.9, exceeding the 80.1 of the best baseline SSA (Adachi et al., 2025), indicating that our adaptation itself is already more effective than existing TTA methods. Moreover, combining TVE loss with GTR and SSV further improves performance, showing that model adaptation and trajectory-level verification provide complementary benefits. Overall, SimTTA improves robustness under corrupted inputs through trajectory refinement, simulation-based verification, and temporally verified adaptation.

Effect of Hyperparameters. We analyze the reliability floor a_0, conservatism weight w, and the number of perturbations N_{PTB}, where N_s, N_r, and N_j are all set to N_{PTB}. This gives $N_{\text{GTR}}{=}1{+}3N_{\text{PTB}}$ candidate trajectories (Section 3.1). As shown in Tables 5c, 5b, and 5a, mean PDMS is relatively insensitive to a_0 and w. Reducing N_{PTB} from 10 to 5 increases FPS from 9.4 to 12.8 with only a marginal PDMS drop from 85.4 to 85.0, showing that SimTTA can trade a small performance loss for faster, near-real-time inference. We therefore use $N_{\text{PTB}}{=}10$, $w{=}5$, and $a_0{=}0.95$ by default.

4.4 Visualization Results

Figure 4 compares the predicted trajectories of the Source, LearnableBN (Liao et al., 2026), and our SimTTA on two representative corruption types (Rain and FOV Lost). Each panel shows the BEV of the current input with the drivable area, lane structure, ego vehicle, surrounding agents, and the trajectory. While the Source and LearnableBN often produce trajectories that drive onto the sidewalk or collide with surrounding vehicles, our SimTTA consistently stays inside the drivable area and follows the lane direction. Visualizations on all eight corruption types are in Appendix A.11.

4.5 Discussion

Memory Efficiency and FPS. Table 6 compares GPU memory and frames-per-second (FPS), measured over adaptation and prediction. Ours runs at 9.4 FPS, approximately $7.8\times$ faster than CodeMerge and $10.4\times$ faster than MOS, and faster than every other baseline except Tent, which is marginally quicker at a 6.7 lower PDMS. Our GPU memory remains at 1.4 GB, below the 1.6 to 2.2

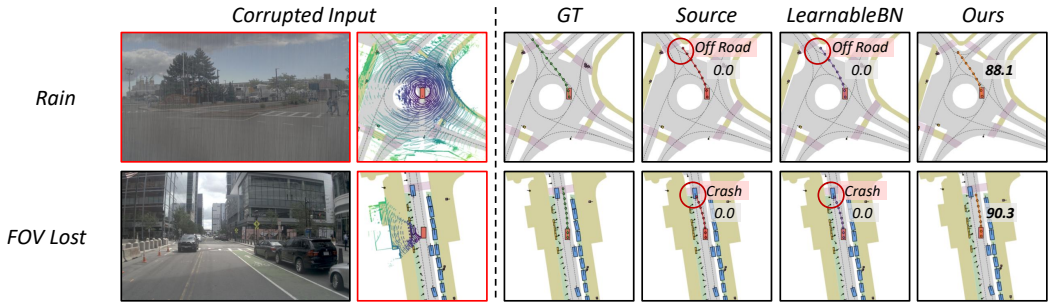


Figure 4: Qualitative comparison on the NAVSIM navtest split built on OpenScene. Each row corresponds to one corruption type (Rain, FOV Lost), and each column shows the BEV of the predicted trajectory by the Source, existing TTA method (LearnableBN), and our SimTTA.

GB of every baseline other than the BN-based ones (BN Adapt., Tent, and LearnableBN), since the candidate search operates in the trajectory space without duplicating model weights or features.

PDMS-Sensitive Subspace Analysis. A trajectory has $3T$ degrees of freedom, and searching all of them at deployment is too slow, so GTR is useful only if the variation that PDMS responds to concentrates in a few directions. We test this on DiffusionDrive by generating 1000 trajectory perturbations around the predicted trajectory of each scene drawn from the NAVSIM (Dauner et al., 2024) navtest split, and applying PCA to the perturbations that improve the PDMS the most. As shown in Figure 5, the first two components are the scaling (79%) and the rotation (18%) of the trajectory, together explaining approximately 97% of the PDMS variation across the eight corruption types, so the improving perturbations occupy a two-dimensional subspace rather than the full $3T$. Scaling moves the trajectory along the road, where the lane ahead stays clear over a long stretch, so a wide range of scalings improves it, while rotation moves it sideways, where a small angle already crosses the lane boundary or a neighbouring agent, so only a narrow range of angles improves it. The remaining 3% changes the shape of the trajectory, which keeps the ego in the same region, and sampling it does not improve the mean PDMS. The two components align with the scaling and rotation directions at cosine similarity 0.98, so GTR samples this plane through its scale, rotation, and joint branches. Sampling only this plane keeps the candidate set at 31 trajectories, so GTR runs at 9.4 FPS while providing reliable trajectories that the prediction of the model alone cannot reach.

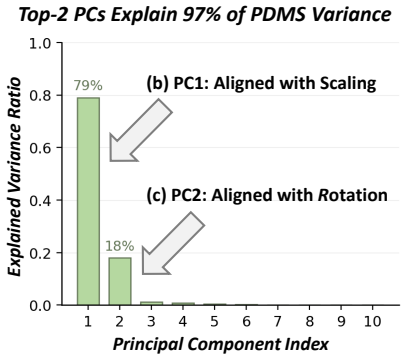


Figure 5: PDMS-sensitive subspace analysis. PC1 and PC2 explain ∼97% of PDMS variation across eight corruptions.

Limitation. In SimTTA, while the verification improves the trajectory under most corruptions, GTR keeps its number of candidates small to stay within the latency budget of test time and TVE updates only the BN affine parameters to keep the stream stable, so when a corruption is severe enough to misalign the prediction with the road, neither the candidates nor the adapted BEV representation recovers it. Thus, exploring a way to enlarge the candidate set within a usable latency and to adapt more of the model without the updates accumulating into a degradation over a long stream can be a promising avenue for our future work. See Appendix A.12 for a detailed discussion.

5 Conclusion

We present a verification-driven test-time adaptation framework for trajectory planning in autonomous driving. Our method consists of three key components: the TVE loss adapts only the BN affine parameters using the BEV cells that pass the temporal verification, GTR geometrically expands the predicted trajectory into candidates that reach output regions the corrupted model may not produce directly, and SSV scores each candidate by simulating it on a reconstructed scene context from the perception outputs of the model, while a lightweight reliability gate keeps the selection conservative on unreliable samples. Experimental results demonstrate the effectiveness of our SimTTA, achieving consistent state-of-the-art performance across eight corruption types on two datasets.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation, a task for which disclosure is required: parts of the manuscript were initially written in the authors' first language and translated into English with the aid of a large language model, after which the authors revised and verified the resulting text.

We have not used generative AI tools to help develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to design or provide feedback on our research methodology or experiments, to implement methods, to clean or reformat datasets, or to interpret results; all of these were carried out by the authors. Generating synthetic data sets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and supporting qualitative and thematic data analysis are not applicable to this work.

Additionally, we used generative AI tools to draft parts of the manuscript and to edit it for readability, including grammar, word choice, and sentence-level rephrasing.

We have reviewed all AI-assisted work. Each drafted, translated, or edited passage was checked by the authors against the intended technical meaning, and we verified that no claim, numerical result, or citation was introduced, removed, or altered by the model; all references were confirmed against their original sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The problem setting and the constraints under which the adaptation operates are formalized in Appendix A.7. The definition of each component and the set of parameters updated at test time are specified in Section 3. The remaining procedure of Scene Context Reconstruction is detailed in Appendix A.8. The datasets, the corruption types and severity levels, the base planners, the baselines, and the evaluation protocol are described in Section 4.1. The complete experimental setup, including all hyperparameter values, the batch-normalization statistics used at inference, and the hardware, is reported in Appendix A.9. The release of our source code is described in Appendix A.13.

REFERENCES

Kazuki Adachi, Shin'ya Yamaguchi, Atsutoshi Kumagai, and Tomoki Hamagami. Test-time adaptation for regression by subspace alignment. In *International Conference on Learning Representations*, volume 2025, pp. 77850–77874, 2025.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11621–11631, 2020.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

Shaoyu Chen, Bo Jiang, Hao Gao, Bencheng Liao, Qing Xu, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. *arXiv preprint arXiv:2402.13243*, 2024a.

Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Chang Huang, and Wenyu Liu. Polardetr: Polar parametrization for vision-based surround-view 3d detection. *Image and Vision Computing*, 156:105438, 2025a.

Zhuoxiao Chen, Zixin Wang, Yadan Luo, Sen Wang, and Zi Huang. Dpo: dual-perturbation optimization for test-time adaptation in 3d object detection. In *Proceedings of the ACM International Conference on Multimedia*, pp. 4138–4147, 2024b.

Zhuoxiao Chen, Junjie Meng, Mahsa Baktashmotlagh, Yonggang Zhang, Zi Huang, and Yadan Luo. Mos: Model synergy for test-time adaptation on lidar-based 3d object detection. *International Conference on Learning Representations*, 2025b.

Jie Cheng, Yingbing Chen, and Qifeng Chen. Pluto: Pushing the limit of imitation learning-based planning for autonomous driving. *arXiv preprint arXiv:2404.14327*, 2024a.

Jie Cheng, Yingbing Chen, Xiaodong Mei, Bowen Yang, Bo Li, and Ming Liu. Rethinking imitation-based planners for autonomous driving. In *IEEE International Conference on Robotics and Automation*, pp. 14123–14130. IEEE, 2024b.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, et al. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *Advances in Neural Information Processing Systems*, 37:28706–28719, 2024.

Yinpeng Dong, Caixin Kang, Jinlai Zhang, Zijian Zhu, Yikai Wang, Xiao Yang, Hang Su, Xingxing Wei, and Jun Zhu. Benchmarking robustness of 3d object detection to common corruptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1022–1032, 2023.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2012.

Chong He, Yuechen Luo, Fang Li, Shaoqing Xu, and Fuxi Wen. Drivever: Lightweight trajectory evaluator as test-time verifier for autonomous driving. *arXiv preprint arXiv:2607.00399*, 2026.

Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *Proceedings of the European Conference on Computer Vision*, pp. 533–549. Springer, 2022.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17853–17862, 2023.

Junjie Huang, Guan Huang, Zheng Zhu, Yun Ye, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021.

Xiaosong Jia, Yulu Gao, Li Chen, Junchi Yan, Patrick Langechuan Liu, and Hongyang Li. Driveadapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7953–7963, 2023a.

Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21983–21994, 2023b.

Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8350, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yingyan Li, Yuqi Wang, Yang Liu, Jiawei He, Lue Fan, and Zhaoxiang Zhang. End-to-end driving with online trajectory evaluation via bev world model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27137–27146, 2025.

Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, et al. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024a.

Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):2020–2036, 2024b.

Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025.

Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. Maptr: Structured modeling and learning for online vectorized hd map construction. *arXiv preprint arXiv:2208.14437*, 2022.

Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, et al. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12037–12047, 2025a.

Bencheng Liao, Shaoyu Chen, Yunchi Zhang, Bo Jiang, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Maptrv2: An end-to-end framework for online vectorized hd map construction. *International Journal of Computer Vision*, 133(3):1352–1374, 2025b.

Dacheng Liao, Mengshi Qi, Liang Liu, and Huadong Ma. Improving batch normalization with test-time adaptation for robust object detection in self-driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 6925–6933, 2026.

Hongbin Lin, Yifan Zhang, Shuaicheng Niu, Shuguang Cui, and Zhen Li. Monotta: Fully test-time adaptation for monocular 3d object detection. In *Proceedings of the European Conference on Computer Vision*, pp. 96–114. Springer, 2024.

Yicheng Liu, Tianyuan Yuan, Yue Wang, Yilun Wang, and Hang Zhao. Vectormapnet: End-to-end vectorized hd map learning. In *International Conference on Machine Learning*, pp. 22352–22369. PMLR, 2023.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400*, 2023.

Shuaicheng Niu, Guohao Chen, Peilin Zhao, Tianyi Wang, Pengcheng Wu, and Zhiqi Shen. Self-bootstrapping for versatile test-time adaptation. In *International Conference on Machine Learning*, 2025.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7077–7087, 2021.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33:11539–11551, 2020.

Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 35:6531–6543, 2022.

Chonghao Sima, Kashyap Chitta, Zhiding Yu, Shiyi Lan, Ping Luo, Andreas Geiger, Hongyang Li, and Jose M Alvarez. Centaur: Robust end-to-end autonomous driving with test-time training. *arXiv preprint arXiv:2503.11650*, 2025.

Wenchao Sun, Xuewu Lin, Yining Shi, Chuang Zhang, Haoran Wu, and Sifa Zheng. Sparsedrive: End-to-end autonomous driving via sparse scene representation. In *IEEE International Conference on Robotics and Automation*, pp. 8795–8801. IEEE, 2025.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022a.

Shihao Wang, Yingfei Liu, Tiancai Wang, Ying Li, and Xiangyu Zhang. Exploring object-centric temporal modeling for efficient multi-view 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3621–3631, 2023.

Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, pp. 180–191. PMLR, 2022b.

Shaoyuan Xie, Lingdong Kong, Wenwei Zhang, Jiawei Ren, Liang Pan, Kai Chen, and Ziwei Liu. Benchmarking and improving bird's eye view perception robustness in autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3878–3894, 2025.

Yihong Xu, Eloi Zablocki, Yuan Yin, Elias Ramzi, Ellington Kirby, Alexandre Boulch, and Matthieu Cord. Test-time trajectory optimization for autonomous driving. *arXiv preprint arXiv:2606.07170*, 2026.

Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Jain, Jeff Schneider, and Katerina Fragkiadaki. Diffusion-es: Gradient-free planning with diffusion for autonomous driving and zero-shot instruction following. *arXiv preprint arXiv:2402.06559*, 2024a.

Huitong Yang, Zhuoxiao Chen, Fengyi Zhang, Zi Huang, and Yadan Luo. Codemerge: Codebook-guided model merging for robust test-time adaptation in autonomous driving. *arXiv preprint arXiv:2505.16524*, 2025a.

Jiazhi Yang, Kashyap Chitta, Shenyuan Gao, Long Chen, Yuqian Shao, Xiaosong Jia, Hongyang Li, Andreas Geiger, Xiangyu Yue, and Li Chen. Resim: Reliable world simulation for autonomous driving. In *Advances in Neural Information Processing Systems*, 2025b.

Zetong Yang, Li Chen, Yanan Sun, and Hongyang Li. Visual point cloud forecasting enables scalable autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024b.

Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. In *Proceedings of the European Conference on Computer Vision*, pp. 659–675. Springer, 2022.

Bozhou Zhang, Nan Song, Jingyu Li, Xiatian Zhu, Jiankang Deng, and Li Zhang. Future-aware end-to-end driving: Bidirectional modeling of trajectory planning and scene evolution. In *Advances in Neural Information Processing Systems*, 2025.

Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021.

Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *Proceedings of the European Conference on Computer Vision*, pp. 1–21. Springer, 2022.

Yupeng Zheng, Pengxuan Yang, Zebin Xing, Qichao Zhang, Yuhang Zheng, Yinfeng Gao, Pengfei Li, Teng Zhang, Zhongpu Xia, Peng Jia, et al. World4drive: End-to-end autonomous driving via intention-aware physical latent world model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 28632–28642, 2025.

Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17863–17873, 2023.

A APPENDIX

We include additional technical details in the following appendices.

- Section A.1 (Comparison on Clean Conditions after Adaptation with SimTTA). Checks that adapting to a corruption leaves no lasting damage once the input turns clean, and that our SimTTA still improves the clean performance.
- Section A.2 (Effect of the TVE Loss on the BEV Representation). Measures the BEV perception quality (mIoU, calibration error, and boundary F1) of the adapted model against a batch statistics baseline across corruption types.
- Section A.3 (Method Agnosticism across Architectures). Reports results on TransFuser and UniAD to verify that our SimTTA generalizes across diverse end-to-end planners.
- Section A.4 (Robustness across Corruption Severity). Evaluates the robustness of our SimTTA across static severity levels (1, 3, 5) and a Gradual setting in which the corruption severity repeatedly cycles between 1 and 5 over the test stream.
- Section A.5 (Robustness to Compound Corruptions). Evaluates the robustness of our SimTTA on compound corruption settings where two corruption types are chained on the same sample.
- Section A.6 (Oracle Analysis with PDMS). Quantifies the upper bound of our SimTTA by replacing the SSV verification signal with PDMS, which uses ground-truth scene context to verify trajectory candidates.
- Section A.7 (Preliminaries). Formalizes the problem setting of test-time adaptation for trajectory planning and defines the simulation-based PDMS that underlies SimTTA.
- Section A.8 (More Details of Scene Context Reconstruction). Details the agent matching and velocity estimation procedure of Scene Context Reconstruction in our SimTTA, including the ego-motion compensation, the nearest-neighbor matching constraint, and the simulation step size.
- Section A.9 (Implementation Details). Describes the full experimental setup, including hyperparameter configurations and the BN inference behavior of our SimTTA.
- Section A.10 (Qualitative Examples of Corrupted Samples). Provides paired camera and BEV visualizations of the eight corruption types on a representative scene from the NAVSIM navtest split built on OpenScene.
- Section A.11 (Additional Visualization Results). Presents qualitative comparisons with existing TTA methods on additional corruption types from the NAVSIM navtest split and the nuScenes-C benchmark, and the PDMS landscape over the GTR search space on all eight corruption types.
- Section A.12 (Failure Case Analysis). Analyzes failure cases in which severe perception collapse prevents further trajectory improvement by our SimTTA.
- Section A.13 (Source Code). Source code will be released after the review process is over.

A.1 COMPARISON ON CLEAN CONDITIONS AFTER ADAPTATION WITH SIMTTA

An adaptation method is only useful in deployment if it does not damage the model once the corruption disappears. Table 7 tests this with DiffusionDrive (Liao et al., 2025a) on the navtest split. For each corruption type we run the full stream with adaptation enabled, reaching a mean PDMS of 85.4 on the corrupted input, and then evaluate the resulting model on the clean split without resetting its parameters. The clean performance returns to 88.0 on average, within 0.1 of the 88.1 obtained by the model that rarely adapted, and the recovery holds for every corruption type, with the lowest values on the two sensor corruptions that degrade the perception the most (FOV Lost 87.8 and Gaussian Noise 87.9). Running the full SimTTA on the clean stream raises the result to 88.5, above the unadapted model, since the verification still finds a better candidate when the input is not corrupted. The adaptation therefore leaves no lasting damage, which is consistent with restricting the update to the BN affine parameters and to the cells that pass the temporal verification.

Table 7: Clean performance after test-time adaptation to a corruption. Clean and Corr. denote the clean and the corrupted dataset, Train training, Eval. evaluation without adaptation, and Adapt. test-time adaptation. The first row rarely sees a corruption, so it is reported only under Mean.

Setting	Weather			Sensor		Motion		Alignment	Mean
	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{FL}$	$\mathcal{G}$	$\mathcal{MO}$	$\mathcal{MB}$	$\mathcal{T}$	
Clean Train → Clean Eval.	–	–	–	–	–	–	–	–	88.1
Clean Train → Corr. Adapt.	85.0	88.2	87.5	82.8	81.4	88.4	87.8	82.1	85.4
Clean Train → Corr. Adapt. → Clean Eval.	88.1	88.1	88.1	87.8	87.9	88.1	88.1	88.1	88.0
Clean Train → Corr. Adapt. → Clean Adapt.	88.6	88.6	88.6	88.3	88.5	88.6	88.6	88.6	88.5

Table 8: BEV perception quality before and after the test-time adaptation of SimTTA. BN stats denotes the batch-statistics BN forward without our adaptation. The ECE uses 10 bins and the boundary F1 a tolerance of 2 cells.

Metric	Method	Weather			Sensor		Motion		Alignment	Mean
		$\mathcal{S}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{FL}$	$\mathcal{G}$	$\mathcal{MO}$	$\mathcal{MB}$	$\mathcal{T}$	
mIoU ↑	BN stats	0.203	0.213	0.191	0.091	0.192	0.216	0.208	0.151	0.183
	Ours	**0.250**	**0.268**	**0.247**	**0.124**	**0.226**	**0.274**	**0.259**	**0.187**	**0.230**
ECE ↓	BN stats	0.148	0.141	0.151	0.277	0.159	0.138	0.145	0.203	0.170
	Ours	**0.116**	**0.105**	**0.113**	**0.246**	**0.134**	**0.101**	**0.113**	**0.183**	**0.139**
Boundary F1 ↑	BN stats	0.346	0.358	0.324	0.165	0.330	0.362	0.354	0.279	0.315
	Ours	**0.395**	**0.419**	**0.379**	**0.208**	**0.367**	**0.427**	**0.410**	**0.320**	**0.366**

Table 9: Comparison of PDMS on NAVSIM navtest split using **TransFuser** as the base planner. We report PDMS for each of the eight corruption types and the mean across all corruptions.

Methods	Weather			Sensor		Motion		Alignment	Mean
	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{FL}$	$\mathcal{G}$	$\mathcal{MO}$	$\mathcal{MB}$	$\mathcal{T}$	
Source (*No Adapt.*)	67.1	75.2	74.3	69.0	60.6	80.9	76.6	70.5	71.8
BN Adapt.	72.2	73.9	72.1	62.6	71.3	73.4	73.1	67.1	70.7
Tent(ICLR'21)	61.9	64.8	62.5	59.8	60.6	63.9	64.5	60.2	62.3
MonoTTA(ECCV'24)	71.1	71.8	69.9	62.1	70.6	70.9	71.2	65.6	69.2
MOS(ICLR'25)	52.9	50.6	60.8	54.8	40.8	61.6	66.6	54.8	55.4
SSA(ICLR'25)	72.3	74.1	72.2	62.6	71.4	73.4	73.3	67.2	70.8
SPA(ICML'25)	70.6	72.2	69.8	57.0	71.9	70.9	70.0	65.6	68.5
CodeMerge(NeurIPS'25)	59.7	71.8	66.2	68.8	59.9	79.3	76.4	70.5	69.1
LearnableBN(AAAI'26)	67.1	75.3	74.6	69.0	59.5	80.9	76.5	70.5	71.7
SimTTA (Ours)	**82.7**	**84.7**	**82.7**	**82.5**	**79.8**	**85.1**	**84.7**	**78.5**	**83.0**

A.2 Effect of the TVE Loss on the BEV Representation

The TVE loss is designed to recover the shared BEV representation under corruption. The quality of this representation is the direct signature of adaptation, since the candidate selection of SSV and GTR operates in the trajectory space and cannot modify the BEV prediction, so any gain in perception quality comes from the adaptation alone. We therefore measure three metrics over the full stream of each corruption type and compare against a baseline that replaces our adaptation with plain batch statistics normalization on identical inputs. The mIoU measures prediction accuracy, the ECE measures how trustworthy the confidence is, which matters because both the TVE mask and the reliability gate select cells by confidence, and the boundary F1 measures the structural quality near class boundaries, where drivable area violations are decided. Table 8 shows an improvement on all three metrics for every corruption type, with the mean mIoU rising from 0.183 to 0.230, the ECE falling from 0.170 to 0.139, and the boundary F1 rising from 0.315 to 0.366. This recovery translates into the PDMS gains of the main ablation (Table 4), where the TVE loss alone lifts the source from 78.9 to 80.9 and adds a further gain from 84.1 to 85.4 on top of the candidate selection.

A.3 Method Agnosticism across Architectures

A test-time adaptation framework that depends on the internal structure of a particular planner cannot be deployed on the diverse end-to-end models used in autonomous driving. We therefore evaluate our SimTTA on two additional architectures, TransFuser (Prakash et al., 2021) and UniAD (Hu et al., 2023), and report the results in Table 9 and Table 10. Our SimTTA outperforms every TTA baseline on all eight corruption types for both architectures, achieving mean PDMS of 83.0 on

Table 10: Comparison of PDMS on NAVSIM navtest split using **UniAD** as the base planner. We report PDMS for each of the eight corruption types and the mean across all corruptions.

Methods	Weather			Sensor		Motion		Alignment	Mean
	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{FL}$	$\mathcal{G}$	$\mathcal{MO}$	$\mathcal{MB}$	$\mathcal{T}$	
Source (*No Adapt.*)	63.9	73.8	73.9	69.0	58.1	78.7	73.8	70.0	70.2
BN Adapt.	68.1	70.2	68.5	61.1	64.3	69.7	69.2	65.2	67.0
Tent(ICLR'21)	62.4	64.7	62.7	59.7	58.8	64.5	63.5	62.3	62.3
MonoTTA(ECCV'24)	61.9	63.1	61.0	55.6	58.1	62.4	63.3	59.1	60.6
MOS(ICLR'25)	52.7	57.3	58.2	56.4	50.7	63.0	54.8	55.3	56.1
SSA(ICLR'25)	68.0	70.2	68.1	61.1	64.2	69.8	69.3	65.2	67.0
SPA(ICML'25)	63.5	64.9	62.6	55.8	63.5	64.7	63.3	61.6	62.5
CodeMerge(NeurIPS'25)	60.0	66.9	67.3	68.2	58.1	78.7	73.6	69.8	68.7
LearnableBN(AAAI'26)	64.0	73.7	74.1	69.1	62.0	78.7	73.3	70.0	70.6
SimTTA (Ours)	**79.9**	**83.2**	**80.7**	**81.8**	**76.3**	**83.7**	**83.4**	**80.1**	**81.1**

Table 11: PDMS results under five static corruption severity levels (1–5) and the Gradual setting, where the corruption severity repeatedly cycles through every level over the test stream ($1\to2\to3\to4\to5\to4\to3\to2\to\cdots$). All results are evaluated on the NAVSIM navtest split built on OpenScene using DiffusionDrive.

Severity	Weather			Sensor		Motion		Alignment	Mean
	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{FL}$	$\mathcal{G}$	$\mathcal{MO}$	$\mathcal{MB}$	$\mathcal{T}$	
1	86.8	88.3	88.2	87.7	87.3	88.4	88.2	87.1	87.8
2	86.7	88.5	88.2	87.2	86.1	88.5	88.3	84.9	87.3
3	86.0	88.2	87.7	85.0	84.1	88.3	88.1	82.0	86.2
4	85.3	88.6	87.8	83.7	82.7	88.6	88.2	82.5	85.9
5	85.0	88.2	87.5	82.8	81.4	88.4	87.8	82.1	85.4
Gradual	86.0	88.5	88.2	85.3	84.6	88.6	88.3	83.9	86.7

TransFuser (+11.2 over the Source baseline) and 81.1 on UniAD (+10.9 over the Source baseline). The improvement pattern is also consistent across the two architectures, with the largest gains on Gaussian Noise and Snow, the two corruptions on which the Source degrades the most, and the smallest gains on Moving Object, the corruption that affects the Source the least. This consistency demonstrates that our SimTTA is architecture-agnostic and can be applied to diverse end-to-end planners without any modification to the underlying model.

A.4 Robustness across Corruption Severity.

We evaluate our SimTTA across five static severity levels (1–5) and a Gradual setting in which the corruption severity repeatedly cycles through every level over the test stream (Table 11). The mean PDMS of our SimTTA gradually decreases from 87.8 at severity 1 to 85.4 at severity 5, with the largest drops on Gaussian Noise (87.3 to 81.4), Temporal Misalignment (87.1 to 82.1) and FOV Lost (87.7 to 82.8), where the degradation directly affects the perception capability of the base planner, while the remaining corruptions (Snow, Rain, Fog, Moving Object, Motion Blur) remain within 2 points across severities. Under the Gradual setting, our SimTTA reaches a mean PDMS of 86.7, above the static severity 3 result (86.2) even though the stream keeps returning to severity 5, where the static result is 85.4. This pattern is consistent with the design intent of TVE loss, which absorbs the cyclically shifting distribution over the stream, while SSV continues to provide a reliable selection signal across a wide range of corruption strengths.

A.5 Robustness to Compound Corruptions

Real-world driving rarely produces a single isolated corruption, since even overtaking on a snowy road already combines visibility loss with motion blur. We evaluate four compound settings in which two corruption types are chained on the same sample at severity 3 each (S+MB, R+G, F+FL, FL+T) in Table 12. Our SimTTA achieves a mean PDMS of 82.9, improving the Source baseline on all four settings and by 9.6 points on average, with the largest gain on Fog+FOV Lost (66.4 $\to$ 83.2, +16.8). The robustness traces back to the post-hoc verification principle that grounds our design: each candidate trajectory is judged by what would actually happen if it were executed, simulated forward through SSV from the model perception output rather than inferred from the input distribution. GTR enlarges the candidate set so that this judgment has trajectories worth selecting between even when

Table 12: PDMS on compound corruptions in which two corruption types are chained on the same sample at severity 3 each (S+MB, R+G, F+FL, FL+T) on the NAVSIM navtest split built on OpenScene with DiffusionDrive.

Methods	Weather + Sensor/Motion		Weather + Sensor	Sensor + Alignment	Mean
	$\mathcal{S}$+$\mathcal{MB}$	$\mathcal{R}$+$\mathcal{G}$	$\mathcal{F}$+$\mathcal{FL}$	$\mathcal{FL}$+$\mathcal{T}$	
Source (*No Adapt.*)	76.8	75.7	66.4	74.5	73.3
BN Adapt.	80.8	78.4	72.0	72.7	76.0
Tent(ICLR'21)	79.8	77.2	68.1	72.6	74.4
MonoTTA(ECCV'24)	78.4	77.0	70.4	71.2	74.2
MOS(ICLR'25)	51.5	69.8	65.4	73.2	65.0
SSA(ICLR'25)	80.6	78.4	72.0	72.7	75.9
SPA(ICML'25)	77.6	72.1	60.2	70.7	70.1
CodeMerge(NeurIPS'25)	55.2	49.3	40.6	53.0	49.5
LearnableBN(AAAI'26)	77.0	75.6	66.5	74.5	73.4
SimTTA (Ours)	**85.5**	**80.8**	**83.2**	**81.9**	**82.9**

Table 13: Oracle analysis using PDMS as the verification signal in place of SSV on the NAVSIM navtest split with DiffusionDrive. PDMS serves as an upper bound by providing perfect scene context, and the gap between SSV and the oracle reflects the approximation error introduced by scene reconstruction.

Verification	Weather			Sensor		Motion		Alignment	Mean
	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{FL}$	$\mathcal{G}$	$\mathcal{MO}$	$\mathcal{MB}$	$\mathcal{T}$	
PDMS	**92.8**	**93.6**	**93.2**	**88.6**	**91.5**	**93.7**	**93.1**	**89.7**	**92.0**
SSV (Ours)	85.1	88.2	87.4	82.7	81.3	88.4	87.8	82.1	85.4

a single forward pass becomes unreliable, and TVE loss stabilizes the representation that produces those candidates. Because the future outcome of a trajectory does not become harder to evaluate when two corruptions co-occur, the verification signal remains usable on compound shifts where weight-space merging methods, which must absorb both distribution changes into a single set of weights, degrade sharply.

A.6 Oracle Analysis with PDMS

To quantify the upper bound of our SimTTA, we replace the SSV verification signal with the PDMS that uses ground-truth scene context. As shown in Table 13, the PDMS oracle reaches a mean PDMS of 92.0 across the eight corruption types, while our SimTTA with SSV reaches 85.4, a gap of 6.6 PDMS that reflects the approximation error introduced by reconstructing the scene context from the perception outputs of the model. The oracle stays above SSV on every corruption type, and the gap ranges from 5.3 on Moving Object and Motion Blur to 10.2 on Gaussian Noise. Since the two signals largely agree on the candidate ranking (Table 1), this gap comes from the reconstruction and motion approximation errors rather than from a different notion of trajectory quality, leaving room for further improvement by a tighter scene reconstruction and a better motion model.

A.7 Preliminaries

In this section, we formalize the problem of test-time adaptation (TTA) for trajectory planning and introduce the key concepts underlying our analysis and method.

Task Formulation. We consider a pre-trained trajectory planning model f_Θ that maps a sensor input $x \in \mathcal{X}$ to a trajectory $\hat{\tau} \in \mathbb{R}^{T\times 3}$, where T is the number of future waypoints and each waypoint holds a planar position and a heading ψ. The model contains L batch normalization (BN) layers $\{\text{BN}_l\}_{l=1}^{L}$, each with running statistics (μ_l, σ_l^2) from the source domain and affine parameters (γ_l, β_l) learned during pre-training. BN layers reside mainly in the feature backbone while planning heads use layer normalization (LN), so adapting BN running statistics or affine parameters modifies the backbone features that the planning head consumes.

In the TTA setting, f_Θ is trained on a clean source domain $\mathcal{D}_{\text{src}}$ and evaluated on a target domain $\mathcal{D}_{\text{tgt}} = \{x_t\}_{t=1}^{N}$ subject to distribution shifts caused by sensor corruptions (Chen et al., 2024b; 2025b; Yang et al., 2025a; Liao et al., 2026), where t indexes the sample in the test stream. TTA

improves predictive quality on the target stream under three constraints: (*i*) **online streaming**, where samples are processed sequentially with no access to future samples; (*ii*) **single-sample batch**, an inherent condition of autonomous driving deployment; and (*iii*) **no source data and no ground-truth trajectory** on the target stream.

Predictive Driver Model Score (PDMS). On the NAVSIM (Dauner et al., 2024) benchmark, trajectory quality is measured by the PDMS, a simulation-based composite metric. Given a predicted trajectory $\tau \in \mathbb{R}^{T\times 3}$ and a scene context c_t that describes the surrounding environment, including HD map (drivable area, lane structure), states and recorded future trajectories of surrounding agents, and initial ego state, PDMS is a score $S : (\tau, c) \to [0, 1]$ computed by a forward simulator:

$$S(\tau, c) = s_{\text{NC}} \cdot s_{\text{DAC}} \cdot \frac{5\, s_{\text{TTC}} + 5\, s_{\text{EP}} + 2\, s_{\text{C}}}{12}, \tag{9}$$

where s_{NC} (no collision) and s_{DAC} (drivable area compliance) are discrete penalties that zero out the score upon safety violation, while s_{TTC} (time-to-collision), s_{EP} (ego progress), and s_{C} (comfort) evaluate continuous aspects of driving quality through a weighted average. All sub-scores are normalized to $[0, 1]$, and we follow NAVSIM (Dauner et al., 2024) for the weighting coefficients (5, 5, 2). Two structural aspects of S underlie our SimTTA. First, S is computed by forward simulation rather than by comparison against a reference trajectory, so its value depends entirely on the inputs τ and c. Second, although c is provided as a ground-truth annotation in NAVSIM (Dauner et al., 2024), no such annotation is available at deployment time. We therefore reconstruct c from the perception outputs of the model in Section 3.2, turning S into a prediction-based score with which SimTTA can evaluate trajectories at test time.

A.8 More Details of Scene Context Reconstruction

This appendix details the agent matching and velocity estimation procedure summarized in Section 3.2.

When the previous-sample agent set $\mathcal{A}_{t-1} = \{a_{t-1}^{(j)}\}_{j=1}^{N_{t-1}}$ is available, where j indexes an agent in the previous sample, SCR estimates per-agent velocities by first compensating for the ego translation and then performing nearest-neighbor matching from current to previous agents:

$$\tilde{\mathbf{p}}_{t-1}^{(j)} = \mathbf{p}_{t-1}^{(j)} - \mathbf{v}_{\text{ego}} \cdot \Delta t, \tag{10}$$

$$j^*(i) = \arg\min_{j\in\mathcal{U}} \|\mathbf{p}_t^{(i)} - \tilde{\mathbf{p}}_{t-1}^{(j)}\|_2, \quad \text{s.t.} \quad \|\mathbf{p}_t^{(i)} - \tilde{\mathbf{p}}_{t-1}^{(j^*(i))}\|_2 \le r_{\text{match}}, \tag{11}$$

where $\mathbf{v}_{\text{ego}}$ is the ego velocity obtained from the ego state, $\mathcal{U}$ is the set of previous-sample agent indices that have not yet been matched, $\Delta t = 0.5$ s is the time interval between samples, and $r_{\text{match}} = 5.0$ m is the matching radius. The velocity of a successfully matched agent is then computed as:

$$\mathbf{v}^{(i)} = (\mathbf{p}_t^{(i)} - \tilde{\mathbf{p}}_{t-1}^{(j^*(i))})/\Delta t. \tag{12}$$

An agent that finds no match within r_{match} is treated as co-moving with the ego, $\mathbf{v}^{(i)} = \mathbf{v}_{\text{ego}}$. The matching and the velocity estimation above operate in the ego-local detection frame of the model, while the simulator evaluates the candidates in the global map frame. SCR therefore transforms the position, the heading $\theta^{(i)}$, and the velocity of every reconstructed agent into the global frame before the constant velocity extrapolation is simulated:

$$\bar{\mathbf{p}}^{(i)} = \mathbf{p}_{\text{ego}} + R(\theta_{\text{ego}})\, \mathbf{p}_t^{(i)}, \quad \bar{\mathbf{v}}^{(i)} = R(\theta_{\text{ego}})\, \mathbf{v}^{(i)}, \quad \bar{\theta}^{(i)} = \theta^{(i)} + \theta_{\text{ego}}, \tag{13}$$

where $R(\cdot)$ is the two-dimensional rotation matrix and $(\mathbf{p}_{\text{ego}}, \theta_{\text{ego}})$ is the pose of the ego vehicle in the global map frame, which is an input provided by localization at test time rather than a label. The simulation step size is set to $\Delta t_{\text{sim}} = 0.1$ s over the $H = 51$ step horizon.

A.9 Implementation Details

For batch normalization (BN) at inference, the normalization statistics are a uniform blend of the source-domain running statistics and the per-channel mean and variance computed over the current input, $(1 - \alpha)$ of the former and α of the latter with $\alpha = 0.5$, which keeps a stable source prior while still following the corrupted input. The affine transformation (γ_l, β_l) is then applied with the

parameters updated by the Temporally-Verified Entropy loss on the previous sample, and no other parameter of the planner is modified. We optimize the BN affine parameters with the Adam (Kingma & Ba, 2014) optimizer at a learning rate of 3.0×10^{-5} on a single NVIDIA A6000 GPU. The TVE loss uses a confidence threshold $\delta = 0.9$, a collapse penalty weight $\eta = 2.0$, and an entropy floor $H_{\min} = 1.0$ nat. The hyperparameters of SSV are set with $r_{\text{match}} = 5.0$ m, $\Delta t_{\text{sim}} = 0.1$ s, and $H = 51$, and the reliability gate uses $a_0 = 0.95$ and $w = 5$, while GTR uses $N_s = N_r = N_j = 10$, $\sigma_s = 0.05$, $\sigma_r = 0.10$ rad, with scale clipping $[s_{\min}, s_{\max}] = [0.8, 1.2]$ and rotation clipping $[r_{\min}, r_{\max}] = [-0.3, 0.3]$ rad. All experiments are implemented in PyTorch (Paszke et al., 2017).

A.10 Qualitative Examples of Corrupted Samples

We illustrate the eight corruption types used in our experiments on a representative scene from the NAVSIM (Dauner et al., 2024) navtest split, showing both sensor inputs of the platform, the camera images and the LiDAR points projected onto the BEV view of the scene, as shown in Figure 6. The eight corruptions span four real-world categories: weather (Snow, Rain, Fog) reproduces the appearance of adverse atmospheric conditions, sensor (FOV Lost, Gaussian Noise) simulates partial camera failure and pixel-level sensor noise, motion (Moving Object, Motion Blur) models scene-level dynamics that the perception module is not trained to handle, and alignment (Temporal Misalignment) captures the latency between perception and planning that arises in real deployment. The red borders in Figure 6 mark the sensor inputs that each corruption affects. The corruption enters through the camera, the LiDAR, or both, and Temporal Misalignment affects the LiDAR alone, so a deployed planner cannot expect the hazard to be confined to a single modality. Compared to the Clean reference (top-left), every corruption visibly degrades the affected input, removing the evidence on which the trajectory prediction of the base planner depends. These qualitative examples confirm that the eight corruption types reflect realistic deployment hazards rather than synthetic noise, motivating the need for an adaptation framework that maintains trajectory quality under such conditions.

A.11 Additional Visualization Results

OpenScene Dataset. We provide visualization results comparing our SimTTA with existing TTA methods (Wang et al., 2021; Chen et al., 2025b; Yang et al., 2025a; Liao et al., 2026) on the NAVSIM (Dauner et al., 2024) navtest split built on OpenScene (Yang et al., 2024b), across all eight corruption types, including those not covered in the main paper (Snow, Fog, Gaussian Noise, Moving Object, Motion Blur, and Temporal Misalignment). The results are shown in Figure 7. When the predicted trajectory requires heading rotation or anticipation of dynamic agents, the existing TTA methods struggle to produce safe trajectories, frequently violating the drivable area boundary or colliding with leading vehicles (Table 2 of our main paper). In contrast, our SimTTA consistently produces trajectories that stay within the drivable area and align with the future motion of surrounding agents in such challenging cases. This improvement comes from (1) TVE loss, which adapts the BN affine parameters on the temporally verified cells and restores the degraded BEV representation, (2) GTR, which expands the prediction into candidates covering the output directions that weight-space adaptation cannot reach, and (3) SSV, which scores every candidate by simulating its future and keeps the safest one.

nuScenes Dataset. We further visualize trajectory predictions on the nuScenes-C (Xie et al., 2025) corruption benchmark in Figure 8, demonstrating the cross-dataset generalization of our SimTTA. Across all eight corruption types, our SimTTA consistently produces trajectories that stay within the drivable area and align with surrounding agent motion, while existing TTA methods frequently fail to recover trajectory quality under the same conditions (Table 3 of our main paper). The improvement holds despite the different scene composition, vehicle dynamics, and sensor configurations between OpenScene and nuScenes, confirming that the SSV verification signal is dataset-agnostic and that the GTR exploration mechanism transfers effectively to unseen distributions.

PDMS Landscape over the GTR Search Space. Figure 3 of the main paper shows the landscape on three corruption types and one scene each. Figure 9 extends it to all eight corruption types on four scenes, where every row is the same scene and every column is one corruption. The prediction of the model, marked by the red square, mostly sits away from the high-scoring region, while the GTR candidates spread over the sampling box and reach it. Among them, the candidate that SSV selects lands inside the high-scoring region and next to the trajectory of the human driver on most

panels, including the rows where the prediction itself scores close to zero. SSV therefore identifies the candidate with the high PDMS from the reconstructed context alone, which is what turns the candidates of GTR into an improvement over the prediction.

A.12 Failure Case Analysis

Our SimTTA depends on the perception outputs of the base planner for the SSV scene reconstruction and on its prediction for the candidates that GTR expands. When the input corruption is severe enough to disrupt the perception of the base planner itself, the GTR candidates collapse together along the same failure mode, so that even the highest-scoring candidate selected by SSV remains unsafe. Figure 10 shows three such failure cases drawn from different corruption types. In these cases all baselines (Source, BN Adapt. (Schneider et al., 2020), CodeMerge (Yang et al., 2025a), LearnableBN (Liao et al., 2026)) and our SimTTA fail in a similar manner, producing trajectories that depart from the drivable area or no longer align with the lane direction shown in the GT BEV. This indicates that the failure mode originates from the perception robustness of the base planner rather than from the adaptation strategy itself, and that improving performance under such severe perception failure requires perception-level countermeasures that are orthogonal to the verification-driven adaptation we propose.

A.13 Source Code

The source code of our work will be released after the review process is over, including the corruption synthesis and the evaluation scripts needed to reproduce the results reported in this paper.

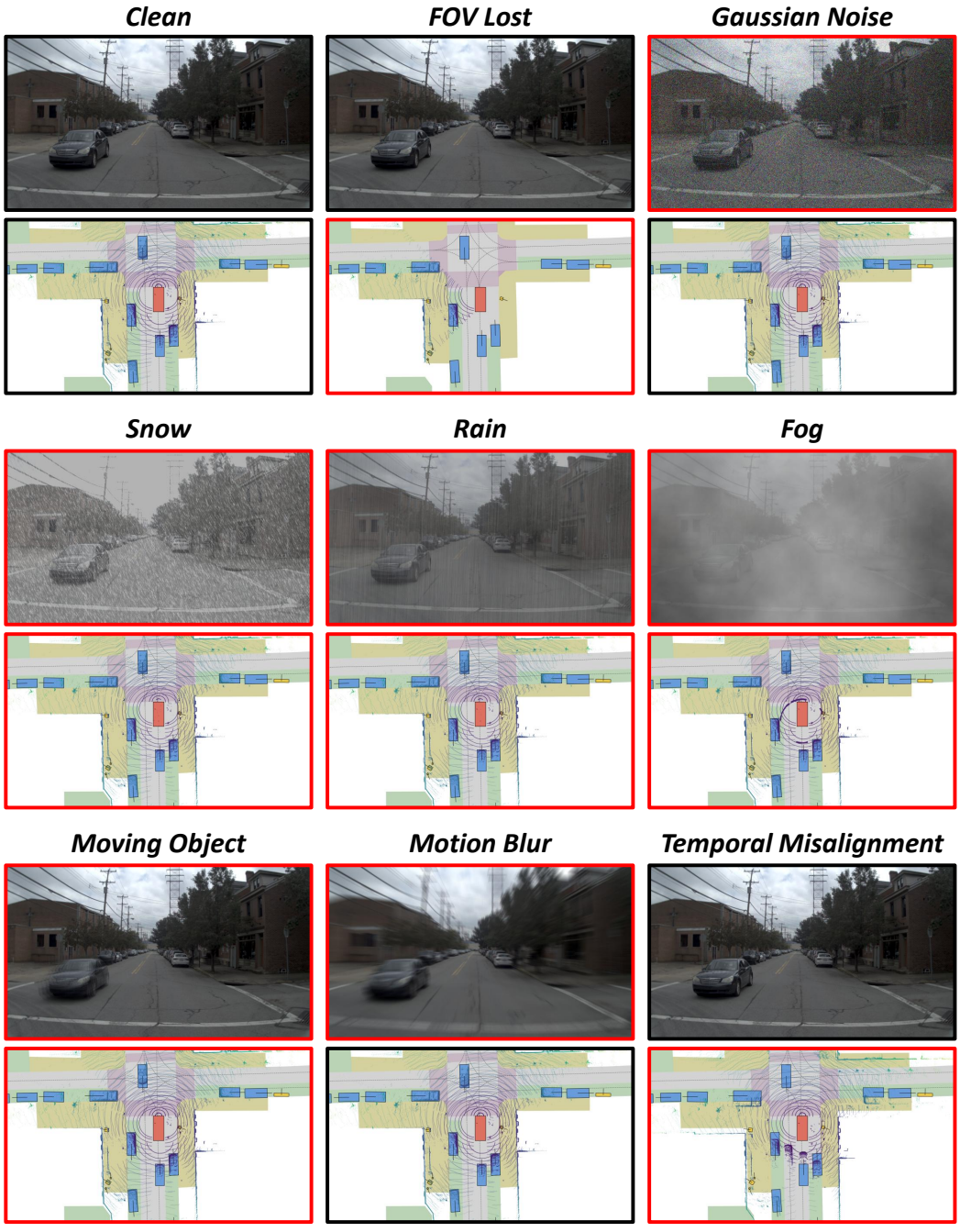


Figure 6: Qualitative examples of the eight corruption types used in our experiments on a representative sample from the NAVSIM navtest split. Each example shows the camera images above and the LiDAR points projected onto the BEV view below. Red borders indicate the sensor inputs that each corruption affects, the camera, the LiDAR, or both.

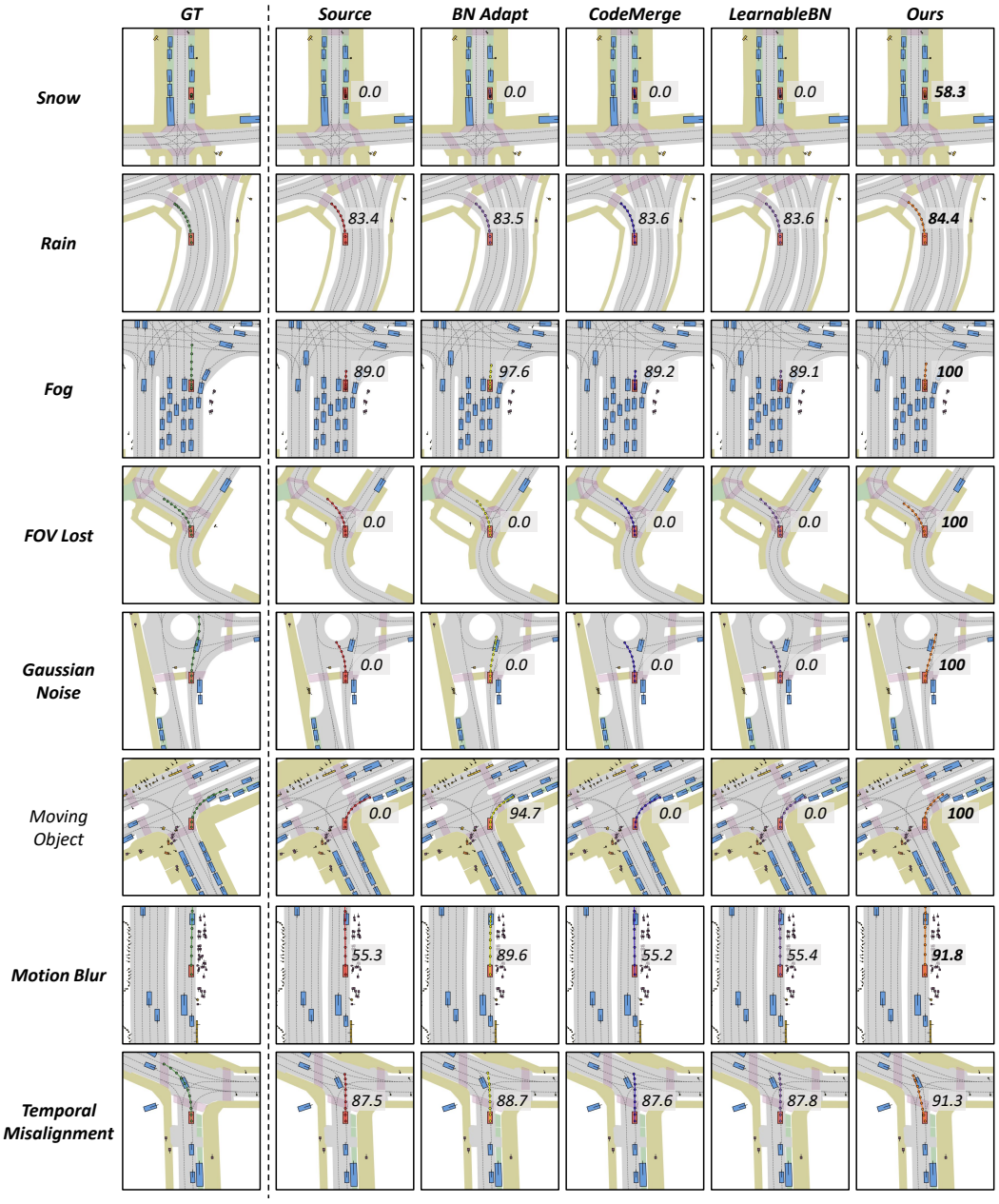


Figure 7: Additional qualitative comparison on the NAVSIM navtest split built on OpenScene. Each row corresponds to one corruption type (Snow, Rain, Fog, FOV Lost, Gaussian Noise, Moving Object, Motion Blur, and Temporal Misalignment), and each column shows the BEV of the ground-truth trajectory and of the trajectories predicted by the Source, existing TTA methods (BN Adapt., CodeMerge, LearnableBN), and our SimTTA.

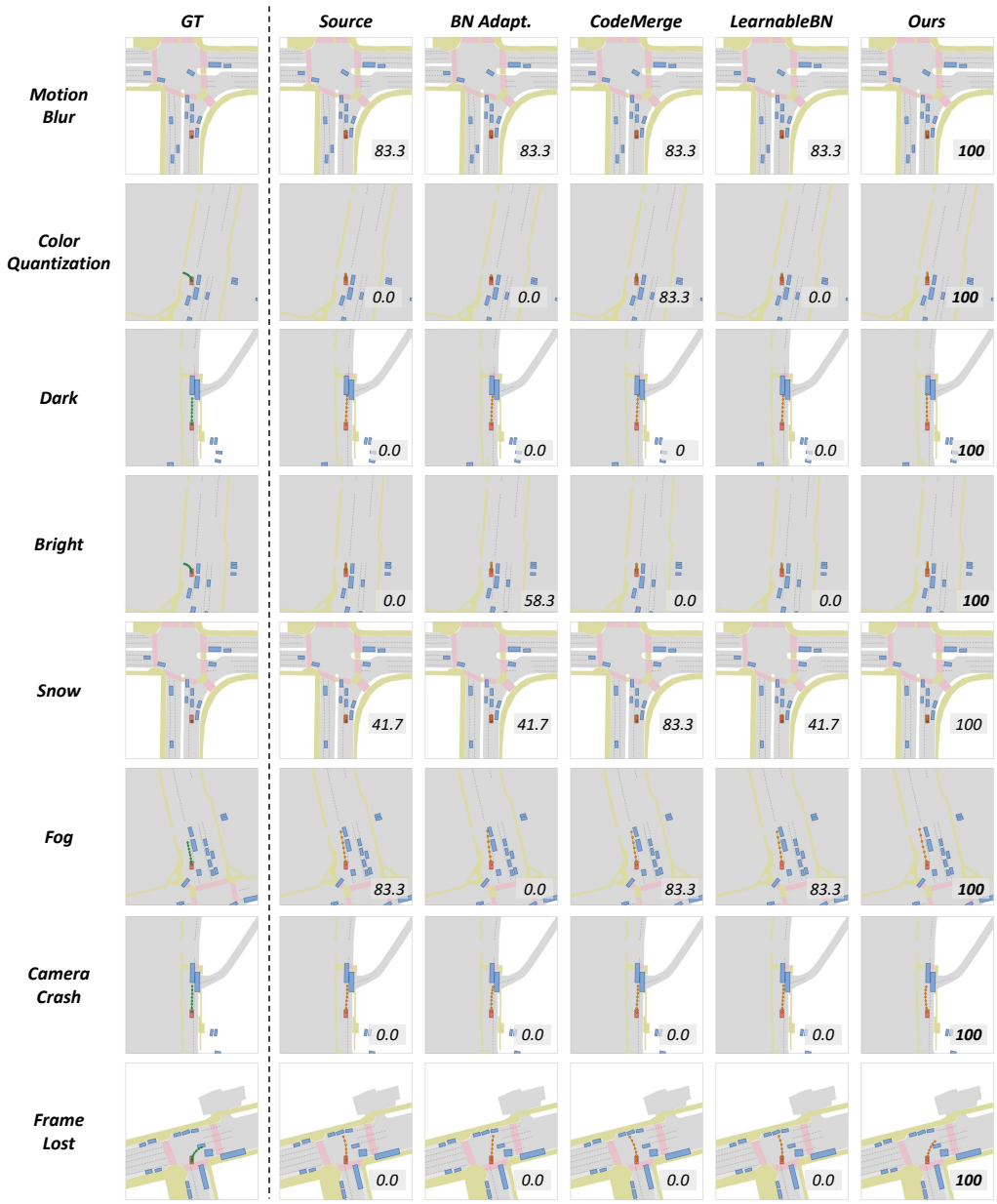


Figure 8: Additional qualitative comparison on the nuScenes-C corruption benchmark. Each row corresponds to one corruption type (Motion Blur, Color Quantization, Dark, Bright, Snow, Fog, Camera Crash, and Frame Lost), and each column shows the BEV of the ground-truth trajectory and of the trajectories predicted by the Source, existing TTA methods (BN Adapt., CodeMerge, LearnableBN), and our SimTTA.

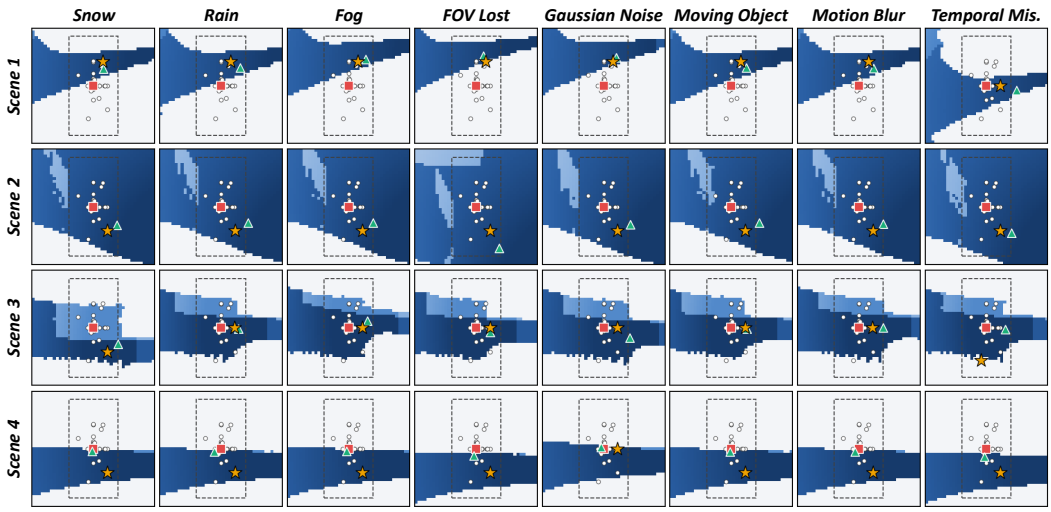


Figure 9: PDMS landscape over the GTR search space on all eight corruption types, with one scene of the NAVSIM navtest split per row and one corruption per column. Axes and color scale follow Figure 3, where the prediction of the model is the red square, the GTR candidates the white dots, the candidate selected by SSV the yellow star, and the human GT the green triangle, and the dashed box is the GTR sampling range.

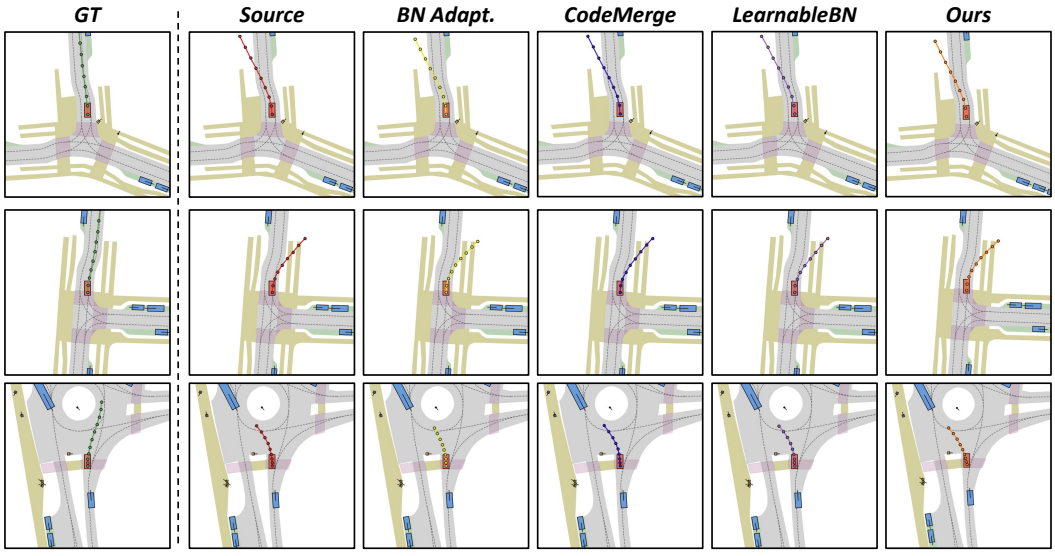


Figure 10: Failure cases of our SimTTA across multiple scenes on the NAVSIM navtest split built on OpenScene. Each row corresponds to one scene under a different corruption type and shows, from left to right, the GT BEV and the BEV of the trajectories predicted by the Source, three existing TTA methods (BN Adapt., CodeMerge, LearnableBN), and our SimTTA.