REFINEDRIVE: RELIABLE FAILURE-GUIDED LEARNING FOR VISION-LANGUAGE-ACTION DRIVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models for autonomous driving rely heavily on successful expert demonstrations, leaving model-specific failures underexploited. Learning from these failures is hindered by unreliable diagnoses, poorly matched correction targets, and coarse rewards. We propose **RefineDrive**, a failure-guided post-training framework that learns from self-generated failures through targeted supervision and safety-aware reinforcement learning. Reliable Diagnosis derives structured, verifiable feedback on collisions and drivable-area violations directly from simulator states. Minimum-Correction Target Retrieval searches a clustered human trajectory bank for nearby corrections that satisfy hard-safety constraints in the current scene, prioritizing preservation of the failed prediction's motion pattern. Conditioned on the driving context and failed trajectory, Correction SFT learns to generate the diagnosis followed by the retrieved correction as a training-only auxiliary task. We then apply GRPO with a Safety-Layered Reward that strictly prioritizes hard-safe trajectories, retains continuous safety feedback for both unsafe and hard-safe trajectories, and rewards driving progress only after hard safety is satisfied. At inference, the policy directly predicts trajectories from the driving context without an explicit diagnosis or repair stage. On NAVSIM v1, RefineDrive improves the 4B base SFT policy from 87.7 to 91.7 PDMS. Using the same checkpoint without additional training, RefineDrive achieves 89.4 EPDMS on the original NAVTEST scenes evaluated with NAVSIM v2 extended metrics. Controlled ablations support the benefits of structured diagnosis supervision, retrieved corrections, and safety-layered optimization for direct planning.

1 INTRODUCTION

Vision-Language-Action (VLA) models have shown strong potential for end-to-end autonomous driving, benefiting from large-scale vision-language pretraining and expert demonstrations. However, most approaches still rely primarily on imitation learning from successful human behavior, while failures exposed by the model's own rollouts—such as collisions and departures from the drivable area—remain underexploited. For a policy with strong basic driving capabilities, further improvement may depend less on imitating more expert behavior and more on identifying and correcting the remaining weaknesses revealed by its own failures.

Recent studies have begun to explore learning from failures, but three key limitations remain, as illustrated in Fig. 1. First, **failure diagnosis can be unreliable**. Methods that rely on large VLMs or teacher models to explain failures may produce open-ended diagnoses that are difficult to verify, with inaccurate event localization, incorrect causal attribution, or inconsistency with the simulated outcome. Second, **the correction target may be poorly matched to the failure**. Since a failed trajectory is often only partially incorrect, directly replacing it with human ground truth can introduce unnecessary behavioral changes and lead to over-correction. Third, **coarse reward signals provide limited optimization guidance**. An aggregated driving score may insufficiently distinguish the extent and timing of safety violations among unsafe trajectories or safety margins among safe ones, limiting fine-grained safety optimization.

To address these limitations, we propose **RefineDrive**, a failure-guided post-training approach with three components. **Reliable Diagnosis** derives structured, simulator-verifiable feedback for collisions and drivable-area violations. **Minimum-Correction Target Retrieval** searches a clustered

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.

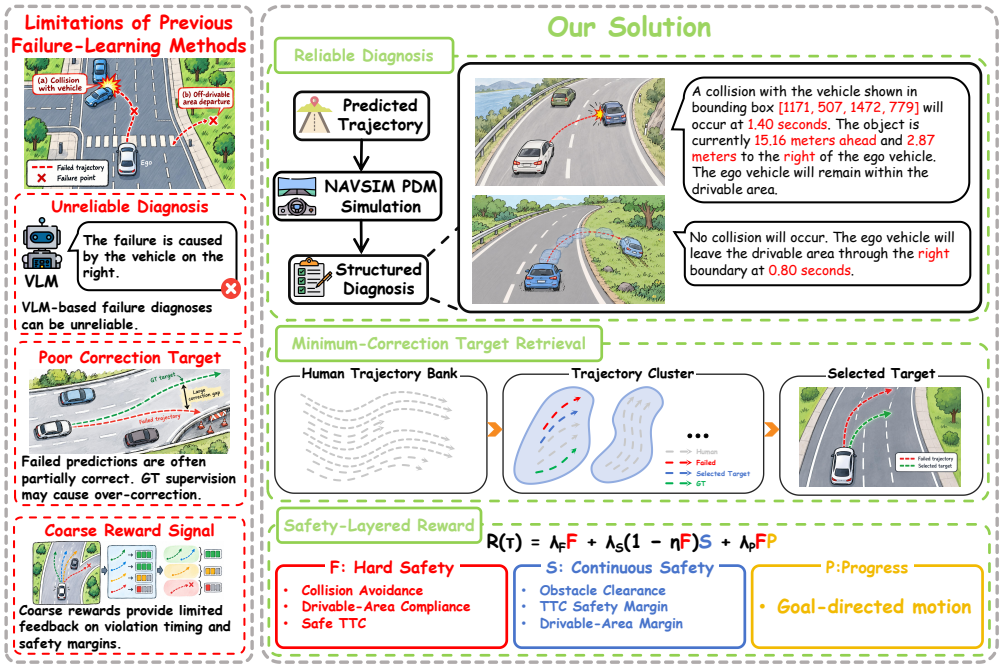


Figure 1: **Motivation and overview of RefineDrive.** Existing failure-learning methods can suffer from unreliable VLM-based diagnoses, over-corrective human ground-truth supervision, and coarse aggregated rewards. We address these limitations with Reliable Diagnosis, which grounds structured feedback in NAVSIM PDM simulation; Minimum-Correction Target Retrieval, which searches a clustered human trajectory bank for safe corrections close to failed predictions; and Safety-Layered Reward, which provides hierarchical signals over hard safety, continuous safety, and driving progress.

human trajectory bank for a nearby correction that satisfies hard-safety constraints in the current scene, prioritizing the motion pattern of the failed prediction. During training, Correction SFT conditions on the driving context and a failed trajectory to learn structured diagnosis and correction. At inference, RefineDrive directly predicts the final trajectory from the driving context, without an explicit diagnosis or repair stage.

Finally, **Safety-Layered GRPO** combines hard safety, continuous safety, and driving progress. It strictly prioritizes hard-safe trajectories, retains continuous safety feedback for both unsafe and hard-safe trajectories, and rewards progress only after hard safety is satisfied.

RefineDrive uses self-generated rollouts to expose residual weaknesses, derives simulator-verifiable diagnoses, retrieves minimally modified safe corrective targets, and further refines the policy with safety-layered reinforcement learning. Rather than repeatedly learning the full distribution of expert behavior, RefineDrive focuses post-training on the specific deficiencies currently exhibited by the policy.

Our main contributions are summarized as follows:

- We introduce **RefineDrive**, which explicitly exploits failures from the model's own rollouts to address residual weaknesses in autonomous driving VLAs.
- We propose **Reliable Diagnosis and Minimum-Correction Target Retrieval**. Simulator-derived structured diagnoses provide reproducible and verifiable failure feedback, while cluster-based retrieval selects safe corrective trajectories close to failed predictions, reducing unnecessary behavioral changes and over-correction.
- We develop a **Safety-Layered Reward** that prioritizes hard safety, provides continuous safety feedback for both unsafe and hard-safe trajectories, and rewards driving progress

only after hard safety is satisfied. Experiments on NAVSIM demonstrate the effectiveness of RefineDrive.

2 Related Work

2.1 Vision-Language-Action Models for Autonomous Driving

Vision-language models (VLMs) have been applied to autonomous driving for scene understanding, reasoning, and decision making Xu et al. (2024); Mao et al. (2023); Shao et al. (2024); Sima et al. (2024); Nie et al. (2024). DriveVLM Tian et al. (2024) emphasizes language-space reasoning for hierarchical planning, whereas VLA models more tightly integrate multimodal reasoning and action generation. OpenDriveVLA Zhou et al. (2026b) aligns instance-aware visual and language representations for autoregressive actions, while AutoVLA Zhou et al. (2026c) combines semantic reasoning, discretized action tokens, and reinforcement fine-tuning. ReCogDrive Xiong et al. (2026) integrates a cognitive VLM with a diffusion planner, and Qwen-Drive Zhou et al. (2026a) explores unified representations for perception, understanding, and planning.

ReflectDrive Li et al. (2026a) introduces discrete-diffusion planning with inference-time safety reflection: a safety scorer identifies unsafe waypoints, local search finds safe anchors, and inpainting regenerates the surrounding trajectory. Although both methods repair unsafe predictions, ReflectDrive performs waypoint-level inference editing, whereas we retrieve complete, scene-validated trajectories and pair them with structured diagnoses as offline Correction SFT supervision.

2.2 Learning from Failures

Learning from failures complements imitation from successful demonstrations. ChauffeurNet Bansal et al. (2018) perturbs expert trajectories to synthesize unsafe situations. Recent VLA methods exploit failures more directly: ELF-VLA Luo et al. (2026) uses teacher-generated diagnoses for trajectory refinement, SafeAlign-VLA Tian et al. (2026) constructs failure descriptions and counterfactual positives for supervised and reinforcement learning, and FIRE-VLA Dou (2026) employs failure-triggered self-distillation with privileged future information. Rather than relying on open-ended teacher explanations, we derive structured diagnoses directly from simulator states, grounding violation times, involved objects, and boundary-exit directions in verifiable geometric and kinematic evidence.

The concurrent R^2LPL framework Gong et al. (2026) retrieves feasible trajectory anchors from recoverable closed-loop states and updates an anchor-scoring planner through replay-based lifelong learning. Its target scores combine rule-based planning quality with route and expert consistency using state-dependent weights. Both methods construct corrective supervision from policy failures, but differ in the correction setting and target-selection criterion. We formulate our correction stage as trajectory-conditioned behavior repair as a training-only auxiliary task: the failed prediction serves as both the reference for retrieving a nearby, scene-verified safe trajectory from a clustered human trajectory bank and an explicit conditioning input for Correction SFT. Rather than supervising anchor scores, we train the VLA to autoregressively generate a simulator-grounded diagnosis followed by the corrected trajectory. Safety-Layered GRPO subsequently refines the policy.

3 Method

3.1 Overview

We propose RefineDrive that enables an already capable driving VLA to improve from safety-critical failures exposed by its own behavior. As illustrated in Fig. 2, our framework consists of four stages: **(1) Failure Discovery**, **(2) Structured Failure Correction**, **(3) Correction SFT**, and **(4) Safety-Layered Reinforcement Learning**.

The resulting framework progressively improves the policy by focusing learning on weaknesses revealed by its own rollouts rather than repeatedly imitating already-mastered expert behavior.

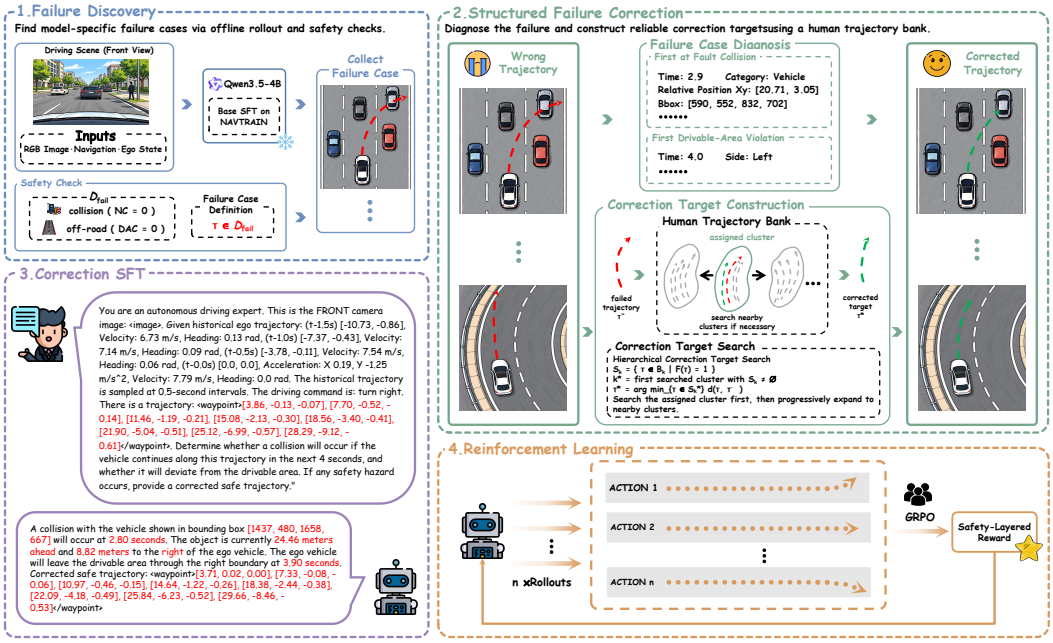


Figure 2: **Training pipeline of RefineDrive.** Offline rollouts and simulator-based safety checks expose model-specific failures. For each failure, we construct a structured diagnosis and retrieve a nearby safe correction from a human trajectory bank. Conditioned on the driving context and failed trajectory, Correction SFT learns to generate the diagnosis followed by the correction. Finally, GRPO refines the policy with the Safety-Layered Reward, providing fine-grained feedback beyond aggregated driving scores. Diagnosis and correction are training-only tasks; inference directly maps the driving context to the final trajectory.

3.2 Failure Discovery

We start from a supervised fine-tuned driving policy and perform offline rollouts on the training scenarios. Each predicted trajectory is re-executed using the NAVSIM PDM Simulator to obtain reproducible safety outcomes.

We focus on two directly observable safety-critical failures: **at-fault collisions** and **departures from the drivable area**. Accordingly, we construct the model-specific failure set as

$$\mathcal{D}_{\text{fail}} = \left\{ (q, \tau^-) \;\middle|\; \text{NC}(\tau^-) = 0 \;\vee\; \text{DAC}(\tau^-) = 0 \right\}, \tag{1}$$

where q denotes the driving context, τ^- is the trajectory predicted by the current policy, and NC and DAC denote No At-Fault Collisions and Drivable Area Compliance, respectively.

Unlike artificially synthesized negative samples, the trajectories in $\mathcal{D}_{\text{fail}}$ expose failure modes actually produced by the current policy. They therefore provide targeted cases for subsequent diagnosis and correction.

3.3 Structured Failure Correction

For each discovered failure, we construct correction supervision that answers two complementary questions: *why does the current prediction fail?* and *what is the smallest behavioral change that makes it safe?* We address them through verifiable failure diagnosis and counterfactual minimal correction.

Verifiable Failure Diagnosis. Rather than relying on a teacher VLM to produce open-ended explanations, we derive failure feedback directly from simulator states.

For an at-fault collision, we identify the **first ego at-fault collision** and extract its occurrence time, object category, and relative position with respect to the ego vehicle. The corresponding traffic participant is further projected onto the current front-view image to obtain its 2D bounding box. The resulting feedback explicitly indicates when the collision occurs, which object is involved, and where the object appears in the current visual observation.

For a drivable-area violation, we identify the first time at which the predicted trajectory leaves the valid driving region and determine whether the ego vehicle exits through the left or right boundary.

When the collision object cannot be reliably localized in the current visual observation, we discard the corresponding sample to avoid introducing textual supervision that cannot be grounded in the model input.

The resulting diagnosis is therefore derived from reproducible geometric and kinematic verification rather than subjective model interpretation.

Counterfactual Minimal Correction. For an already capable driving policy, a failed trajectory's maneuver intention and most of its future motion may remain reasonable, with only a limited deviation causing the safety violation. Direct replacement with the human trajectory may introduce unnecessary behavioral changes; we instead search for a **nearby feasible correction** from a real human trajectory bank.

We construct the bank from the training set and group trajectories by motion pattern using K-Means. Given a failed trajectory τ^-, we identify its assigned cluster in the same feature space and search it first to preserve the original motion pattern. If no feasible correction is found, we search the remaining clusters in ascending order of the distance from their centers to the feature representation of τ^-.

Importantly, human trajectories are not regarded as valid correction targets by default. Since trajectories in the bank may originate from different scenes, every candidate is re-evaluated in the current failure scene. We define hard-safety feasibility as

$$F(\tau) = \mathbb{I}[\mathrm{NC}(\tau) = 1 \wedge \mathrm{DAC}(\tau) = 1 \wedge \mathrm{TTC}(\tau) = 1], \tag{2}$$

where NC, DAC, and TTC denote no-at-fault collision, drivable-area compliance, and time-to-collision safety, respectively.

To measure deviation from the original prediction, we use the same trajectory representation as K-Means clustering. We unwrap each trajectory's heading sequence and express it relative to the first waypoint's heading. Longitudinal positions, lateral positions, and relative headings are normalized using trajectory-bank statistics and flattened across all future waypoints into $\phi(\tau)$. The trajectory distance is

$$d(\tau_i, \tau_j) = \|\phi(\tau_i) - \phi(\tau_j)\|_2 . \tag{3}$$

Let $\mathcal{B}_k$ denote the set of human trajectories belonging to cluster k. For each searched cluster, we define its feasible candidate set as

$$\mathcal{S}_k = \{\tau \in \mathcal{B}_k \mid F(\tau) = 1\}. \tag{4}$$

If $\mathcal{S}_k$ is non-empty, the closest feasible trajectory within that cluster is selected as

$$\tau_k^* = \arg\min_{\tau \in \mathcal{S}_k} d(\tau, \tau^-). \tag{5}$$

The final correction target τ^* is given by τ_k^* from the first searched cluster that contains at least one feasible candidate. In practice, candidates within each cluster are evaluated in ascending order of $d(\tau, \tau^-)$, so the search terminates as soon as the first safe candidate is found.

If no feasible trajectory is found after all available candidates have been considered, or if the predefined search budget is reached before finding one, no corrected trajectory is assigned to the corresponding failure case.

3.4 CORRECTION SFT

We use Correction SFT as a training-only auxiliary task that exposes the policy to self-generated failures, their simulator-grounded diagnoses, and corresponding safe corrections.

For each training sample, the input contains the driving context q and the failed trajectory τ^-. The target response y^* consists of the structured diagnosis followed by the retrieved corrective trajectory τ^*. We optimize the policy using the standard autoregressive objective:

$$\mathcal{L}_{\text{corr}} = -\sum_{t=1}^{|y^*|} \log \pi_\theta \left(y_t^* \mid q, \tau^-, y_{<t}^* \right). \tag{6}$$

The purpose of this task is to improve direct planning through failure-specific supervision, rather than to define an inference-time repair procedure. At inference, the policy directly generates the final trajectory conditioned only on the driving context q. It is not given a failed trajectory and does not perform an explicit diagnosis-and-correction pass.

After Correction SFT, we further refine the policy with Safety-Layered GRPO.

3.5 Safety-Layered Reward

We refine the corrected policy with GRPO using a reward that combines *hard safety*, *continuous safety assessment*, and *driving progress*. Hard-safety feasibility is determined by $F(\tau) \in \{0, 1\}$ in Eq. 2, which requires NC, DAC, and TTC to be satisfied simultaneously.

Continuous Safety Score. Binary feasibility alone cannot distinguish failure severity among unsafe trajectories or safety margins among feasible ones. We therefore construct a continuous safety score from simulator-derived margins and violation times. For each safety factor $k \in \{\text{clr}, \text{ttc}, \text{road}\}$, corresponding to obstacle clearance, TTC safety, and drivable-area compliance, let $m_k(\tau)$ denote its minimum signed margin and $t_k(\tau)$ its first violation time. We normalize them as

$$\bar{m}_k(\tau) = \text{clip}\left(\frac{m_k(\tau) - l_k}{u_k - l_k}, 0, 1 \right), \qquad \bar{t}_k(\tau) = \text{clip}\left(\frac{t_k(\tau)}{T}, 0, 1 \right), \tag{7}$$

where $[l_k, u_k]$ is the factor-specific normalization interval and T is the simulation horizon. We set $t_k(\tau) = T$ when the corresponding violation does not occur. The continuous safety score is

$$S(\tau) = \sum_k w_k \left[(1 - \rho_k) \bar{m}_k(\tau) + \rho_k \bar{t}_k(\tau) \right], \qquad \sum_k w_k = 1, \tag{8}$$

where $w_k \geq 0$ and $\rho_k \in [0, 1]$, ensuring $S(\tau) \in [0, 1]$. Higher scores favor larger safety margins and later violations. Thus, S provides graded feedback on failure severity for unsafe trajectories while continuing to distinguish safety margins among hard-safe trajectories.

Safety-Layered Composition. We reward driving progress only after hard safety is satisfied. The progress score is defined using Ego Progress (EP):

$$P(\tau) = \text{EP}(\tau) \tag{9}$$

The final reward is

$$R(\tau) = \lambda_F F(\tau) + \lambda_S \big[1 - \eta F(\tau) \big] S(\tau) + \lambda_P F(\tau) P(\tau), \tag{10}$$

where $\eta \in [0, 1]$ controls the attenuation of the continuous safety reward once hard safety is satisfied. Unsafe trajectories receive only $\lambda_S S(\tau)$. Hard-safe trajectories receive the feasibility bonus, retain a reduced safety-score weight of $\lambda_S(1 - \eta)$, and additionally receive the progress reward.

We set $(\lambda_F, \lambda_S, \lambda_P) = (2.0, 1.0, 0.25)$ and $\eta = 0.5$, retaining half of the continuous safety weight for hard-safe trajectories. Consequently, unsafe trajectories have $R(\tau) \in [0, 1]$, whereas hard-safe trajectories have $R(\tau) \in [2, 2.75]$. This guarantees a strict reward preference for hard-safe trajectories while preserving fine-grained safety feedback within both groups.

4 Experiments

4.1 Experimental Setup

Dataset and Evaluation Protocol. We conduct experiments on NAVSIM Dauner et al. (2024), a planning-oriented autonomous driving benchmark built on OpenScene/nuPlan. Following the official NAVSIM v1 protocol, we use NAVTRAIN for supervised training, failure mining, and reinforcement learning, and reserve NAVTEST exclusively for evaluation. All corrective trajectories

Table 1: Comparison with state-of-the-art methods on NAVSIM v1. All metrics are higher-is-better.

Method	Params.	NC	DAC	TTC	C	EP	PDMS
End-to-End Methods							
UniAD Hu et al. (2023)	–	97.8	91.9	92.9	100	78.8	83.4
TransFuser Chitta et al. (2022)	–	97.7	92.8	92.8	100	79.2	84.0
DiffusionDrive Liao et al. (2025)	–	98.2	96.2	94.7	100	82.2	88.1
Driving VLA Methods							
Base SFT (Qwen3.5-4B) Qwen Team (2026)	4B	98.6	95.8	95.4	100	81.1	87.7
AutoVLA Zhou et al. (2026c)	3B	98.4	95.6	98.0	99.9	81.9	89.1
SafeAlign-VLA Tian et al. (2026)	7B	98.6	97.2	98.1	100	81.7	89.1
DriveVLA-W0 Li et al. (2026b)	8B	98.7	99.1	95.3	99.3	83.3	90.2
ReCogDrive Xiong et al. (2026)	2B	97.9	97.3	94.9	100	87.3	90.8
ELF-VLA Luo et al. (2026)	8B	98.9	98.1	96.0	100	85.3	91.0
DriveTeach-VLA Yang et al. (2026)	3B	98.5	96.9	97.9	98.2	88.5	90.4
DriveMA Zheng et al. (2026)	4B	98.7	97.8	95.5	99.9	86.7	91.2
Qwen-Drive-1.0-RL Zhou et al. (2026a)	4B	98.6	98.2	95.9	100	84.8	90.7
ReflectDrive Li et al. (2026a)	8B	97.7	99.3	93.5	100	86.9	91.1
DriveFine Dang et al. (2026)	8B	98.6	97.9	95.2	99.9	85.5	90.7
RefineDrive	4B	98.7	98.2	95.9	100	87.2	91.7

are constructed from the training split only. We additionally evaluate the same checkpoint used for NAVSIM v1 on the original NAVTEST scenes using the extended metrics of NAVSIM v2 Cao et al. (2025), without additional training. This is a single-stage, original-scene evaluation and does not include synthetic second-stage observations or two-stage pseudo-simulation aggregation.

Metrics. We report Predictive Driver Model Score (PDMS) for overall benchmark comparison. To evaluate our safety-oriented objective, we analyze No At-Fault Collision (NC), Drivable Area Compliance (DAC), and Time-to-Collision (TTC) as the key safety outcomes, alongside Ego Progress (EP) and Comfort (C). We interpret safety and progress jointly: similar aggregate PDMS values need not imply similar safety performance. All metrics are higher-is-better.

Model and Training Details. We instantiate RefineDrive with Qwen3.5-4B and first perform trajectory SFT on NAVTRAIN. The resulting policy is rolled out to mine safety-critical failures, from which we construct the diagnosis-and-correction dataset. We then perform Correction SFT followed by Safety-Layered GRPO. All ablations start from the same base SFT checkpoint. During evaluation, every checkpoint, including those evaluated immediately after Correction SFT, directly predicts eight $(x, y, \text{heading})$ waypoints at 0.5 s intervals over a 4 s horizon from the current front-view image, ego state, and historical trajectory. No failed trajectory is supplied, and no additional diagnosis or trajectory-repair pass is performed. Training hyperparameters and implementation details are provided in Appendix C, while trajectory-bank clustering and retrieval settings are described in Appendix B.2.

4.2 Main Results

Results on NAVSIM v1. Table 1 compares our method with representative end-to-end and VLA planners on NAVSIM v1. Our 4B model achieves 91.7 PDMS, together with 98.7 NC, 98.2 DAC, and 95.9 TTC, demonstrating strong overall and safety performance without relying on a substantially larger foundation model.

Extended-Metric Evaluation on NAVTEST. Using the extended metrics of NAVSIM v2 on the original NAVTEST scenes, RefineDrive achieves 89.4 EPDMS with the same checkpoint used for NAVSIM v1 and no additional training. This complements the v1 results with a broader assessment of planning quality on the same test split.

Table 2: Comparison on the original NAVTEST scenes using NAVSIM v2 extended metrics. Our result uses the same checkpoint as the NAVSIM v1 evaluation, without additional training. All metrics are higher-is-better. * indicates the exact NAVSIM v2 evaluator revision is not specified in the paper. † indicates results evaluated on the bug-fixed version of NAVSIM.

Method	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	EPDMS
ReCogDrive Xiong et al. (2026)	98.3	95.2	99.5	99.8	87.1	97.5	96.6	98.3	86.5	83.6
DiffusionDriveV2 Zou et al. (2025)*	97.7	96.6	99.2	99.8	88.9	97.2	96.0	97.8	91.0	85.5
DriveVLA-W0 Li et al. (2026b)*	98.5	99.1	98.0	99.7	86.4	98.1	93.2	97.9	58.9	86.1
DriveSuprim (ViT-L) Yao et al. (2026)	98.4	98.6	99.6	99.8	90.5	97.8	97.0	98.3	78.6	87.1
ExploreVLA Sheng et al. (2026)*	98.8	96.2	99.6	99.8	87.1	98.2	97.8	98.3	86.8	88.8
DriveFine† Dang et al. (2026)	98.7	97.3	99.5	99.8	88.7	97.8	97.7	98.4	83.8	89.7
RefineDrive†	98.7	98.2	98.1	99.7	90.5	98.2	94.4	97.8	82.7	89.4

Table 3: Ablation of failure-aware Correction SFT. Human GT and Minimal use the same failure cases and identical structured diagnosis supervision. Minimal w/o Diagnosis retains the same corrective trajectory targets as Minimal but removes diagnosis supervision.

Correction Strategy	NC	DAC	EP	TTC	C	PDMS
Base SFT	98.6	95.8	81.1	95.4	**100**	87.7
Human GT	98.6	96.1	81.6	95.7	**100**	88.2
Minimal w/o Diagnosis	98.5	96.2	81.7	95.7	**100**	88.3
Minimal	**98.7**	**96.4**	**82.0**	**95.9**	**100**	**88.6**

4.3 Ablation Studies

We conduct controlled ablations on NAVSIM v1 to study: (1) learning from self-generated failures; (2) minimal correction versus Human GT; (3) the benefit of structured failure-diagnosis supervision; and (4) Safety-Layered Reward versus direct PDMS optimization. All variants share the same base SFT model and evaluation protocol, and Human-GT and minimal-correction variants use the same failure cases.

Failure-Aware Correction. Using self-generated failures with Human GT supervision improves NAVTEST PDMS from 87.7 to 88.2. With the failure cases and structured diagnosis supervision held fixed, replacing Human GT with retrieved minimal corrections further increases PDMS to 88.6, with gains in NC, DAC, EP, and TTC. Across the 32,650 paired Correction-SFT records, however, the retrieved targets have a slightly lower mean PDMS than the corresponding Human GT trajectories (95.4 vs. 95.8). Thus, the downstream improvement cannot be explained by higher average target PDMS. Furthermore, removing diagnosis supervision while keeping the corrective trajectory targets unchanged reduces PDMS from 88.6 to 88.3. This suggests that structured failure-diagnosis supervision benefits the resulting planning policy beyond corrective trajectory supervision alone.

Safety-Layered Reinforcement Learning. Table 4 isolates the RL objective from the same minimal-correction SFT checkpoint. Direct PDMS optimization increases EP from 82.0 to 87.2, but lowers NC from 98.7 to 98.2 and TTC from 95.9 to 94.9. Our Safety-Layered Reward reaches the same reported mean EP (87.2), while retaining the pre-RL NC and TTC scores and increasing DAC to 98.2. Relative to PDMS Reward, it improves NC, DAC, and TTC by 0.5, 0.4, and 1.0 percentage points, respectively, and PDMS from 91.1 to 91.7. This comparison supports improved safety without a reduction in reported mean progress.

Effect on the Final Policy. Table 5 evaluates Correction SFT variants followed by the same Safety-Layered GRPO. Full Correction SFT attains the highest NC, DAC, and TTC among all compared variants. Relative to omitting Correction SFT, the gains are 0.9, 0.5, and 1.8 percentage points, respectively, with EP decreasing from 88.8 to 87.2. Relative to Human-GT Correction SFT, it increases the three safety metrics by 0.6, 0.5, and 1.2 points, while EP decreases from 89.0 to 87.2. Thus, their similar PDMS values (91.7 vs. 91.6) accompany different safety–progress profiles and do not imply similar safety performance. Removing diagnosis while retaining the same corrective

Table 4: Ablation of the RL objective from the same minimal-correction SFT checkpoint.

RL Objective	NC	DAC	EP	TTC	C	PDMS
No RL	**98.7**	96.4	82.0	**95.9**	**100**	88.6
PDMS Reward	98.2	97.8	**87.2**	94.9	**100**	91.1
Safety-Layered Reward	**98.7**	**98.2**	**87.2**	**95.9**	**100**	**91.7**

Table 5: Effect of Correction SFT under the same Safety-Layered GRPO objective. Full Correction SFT achieves the highest NC, DAC, and TTC among the compared variants, with lower EP.

Training Variant	NC	DAC	EP	TTC	C	PDMS
w/o Correction SFT	97.8	97.7	88.8	94.1	**100**	91.2
Minimal-Correction SFT w/o Diagnosis	98.2	97.8	87.5	95.3	99.9	91.3
Human-GT Correction SFT	98.1	97.7	**89**	94.7	99.9	91.6
Full Correction-SFT	**98.7**	**98.2**	87.2	**95.9**	**100**	**91.7**

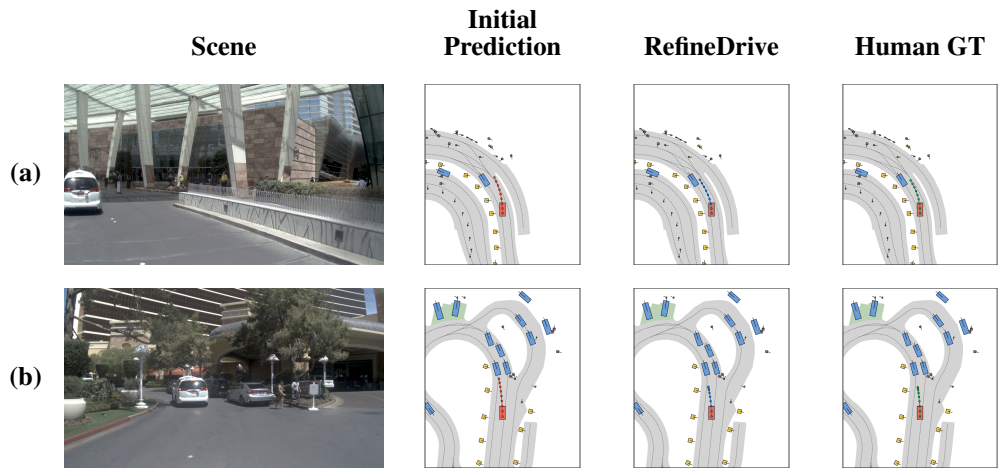


Figure 3: Qualitative comparison of direct trajectory predictions on NAVTEST: (a) off-road cases and (b) collision cases. Red, blue, and green denote the base-policy prediction, RefineDrive prediction, and human trajectory, respectively. Both policies predict independently from the same driving context; RefineDrive is not conditioned on the base-policy prediction.

targets lowers NC, DAC, and TTC by 0.5, 0.4, and 0.6 points, with EP increasing from 87.2 to 87.5. Together, these comparisons support a safety-oriented contribution from diagnosis-and-correction supervision after RL, with an explicit progress trade-off.

4.4 Qualitative Analysis

Fig. 3 compares the base SFT policy and RefineDrive on the same NAVTEST scenes. Both policies directly predict trajectories from the driving context. In these examples, RefineDrive avoids the off-road and collision behaviors exhibited by the base policy.

5 Conclusion

We present RefineDrive, a failure-guided post-training framework integrating Reliable Diagnosis, Minimum-Correction Target Retrieval, and Safety-Layered GRPO, with no inference-time diagnosis or repair. Controlled ablations show that diagnosis-and-correction supervision improves NC, DAC, and TTC after RL at some cost to progress, while Safety-Layered Reward improves all three at the same mean EP as direct PDMS optimization. RefineDrive achieves 91.7 PDMS on NAVSIM v1 and, with the same checkpoint and no further training, 89.4 EPDMS on the original NAVTEST scenes under NAVSIM v2 extended metrics.

AI USE STATEMENT

In this work, we used generative AI tools, including ChatGPT, to provide feedback on research methodology and experimental design, assist with method implementation and code development, and support translation. We additionally used generative AI tools for brainstorming, literature search and summarization, drafting and revising parts of the manuscript, improving readability, and editing research code.

Generative AI tools were not used to formulate or prove mathematical claims, nor were they used to autonomously conduct experiments or make final research decisions. All AI-assisted code, analyses, and manuscript content were reviewed by the authors. Experimental results were obtained through our own experimental pipeline, and technical claims, reported results, and cited literature were manually checked against the corresponding experimental outputs and original sources.

The authors take full responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the assistance of generative AI.

REFERENCES

Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-simulation for autonomous driving. In *Conference on Robot Learning (CoRL)*, 2025.

Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE transactions on pattern analysis and machine intelligence*, 45(11):12878–12895, 2022.

Chenxu Dang, Sining Ang, Yongkang Li, Haochen Tian, Jie Wang, Guang Li, Hangjun Ye, Jie Ma, Long Chen, and Yan Wang. Drivefine: Refining-augmented masked diffusion vla for precise and robust driving. *arXiv preprint arXiv:2602.14577*, 2026.

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.

Hao Dou. Fire-vla: Failure-informed self-evolution for vision-language-action models in autonomous driving. *arXiv preprint arXiv:2608.13395*, 2026.

Cheng Gong, Haoyang Wang, Chao Lu, Zirui Li, and Jianwei Gong. Learning from mistakes: Rollout-retrieval lifelong policy learning for autonomous driving. *arXiv preprint arXiv:2606.30537*, 2026.

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 *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17853–17862. IEEE, 2023.

Pengxiang Li, Yinan Zheng, Yue Wang, Huimin Wang, Hang Zhao, Jingjing Liu, Xianyuan Zhan, Kun Zhan, and Xianpeng Lang. Discrete diffusion for reflective vision-language-action models in autonomous driving. In *International Conference on Learning Representations*, volume 2026, pp. 14647–14666, 2026a.

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, et al. Drivevla-w0: World models amplify data scaling law in autonomous driving. In *International Conference on Learning Representations*, volume 2026, pp. 7890–7911, 2026b.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12037–12047. IEEE, 2025.

Yuechen Luo, Fang Li, Qimao Chen, Shaoqing Xu, Jiaxin Liu, Ziying Song, Zhi-xin Yang, and Fuxi Wen. Unleashing vla potentials in autonomous driving via explicit learning from failures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24833–24842, 2026.

Jiageng Mao, Yuxi Qian, Junjie Ye, Hang Zhao, and Yue Wang. Gpt-driver: Learning to drive with gpt. *arXiv preprint arXiv:2310.01415*, 2023.

Ming Nie, Renyuan Peng, Chunwei Wang, Xinyue Cai, Jianhua Han, Hang Xu, and Li Zhang. Reason2drive: Towards interpretable and chain-based reasoning for autonomous driving. In *European Conference on Computer Vision*, pp. 292–308. Springer, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Hao Shao, Yuxuan Hu, Letian Wang, Guanglu Song, Steven L Waslander, Yu Liu, and Hongsheng Li. Lmdrive: Closed-loop end-to-end driving with large language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15120–15130. IEEE, 2024.

Zihao Sheng, Xin Ye, Jingru Luo, Sikai Chen, and Liu Ren. Explorevla: Dense world modeling and exploration for end-to-end autonomous driving. *arXiv preprint arXiv:2604.02714*, 2026.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. In *European conference on computer vision*, pp. 256–274. Springer, 2024.

Kefei Tian, Yuansheng Lian, Kai Yang, Xiangdong Chen, and Shen Li. Safealign-vla: A negative-enhanced safe alignment framework for risk-aware autonomous driving. *arXiv preprint arXiv:2605.19524*, 2026.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, Kun Ma, et al. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. In *International Conference on Learning Representations*, volume 2026, pp. 157518–157556, 2026.

Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee K Wong, Zhenguo Li, and Hengshuang Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robotics and Automation Letters*, 9(10):8186–8193, 2024.

Yuguang Yang, Canyu Chen, Zhewen Tan, Yizhi Wang, Zichao Feng, Chunyang Liu, Kehua Sheng, Juan Zhang, Linlin Yang, Baochang Zhang, et al. Teaching vision-language-action models what to see and where to look. In *European Conference on Computer Vision*, pp. 317–335. Springer, 2026.

Wenhao Yao, Zhenxin Li, Shiyi Lan, Zi Wang, Xinglong Sun, Jose M Alvarez, and Zuxuan Wu. Drivesuprim: Towards precise trajectory selection for end-to-end planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 11910–11918, 2026.

Weicheng Zheng, Yixin Huang, Qiao Sun, Derun Li, et al. Drivema: Rethinking language interfaces in driving vlas with one-step meta-actions. *arXiv preprint arXiv:2605.21273*, 2026.

Xin Zhou, Zongchuang Zhao, Zhibo Yang, Mingsheng Li, Humen Zhong, Shuai Bai, Du Chu, Ruizhe Chen, Zhaohai Li, Jun Tang, et al. Qwen-drive-1.0: An initial step towards a vision-language foundation model for autonomous driving. *arXiv preprint arXiv:2609.00111*, 2026a.

Xingcheng Zhou, Xuyuan Han, Feng Yang, Yunpu Ma, Volker Tresp, and Alois Knoll. Opendrivevla: Towards end-to-end autonomous driving with large vision language action model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 13782–13790, 2026b.

Zewei Zhou, Tianhui Cai, Seth Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *Advances in Neural Information Processing Systems*, 38: 27920–27956, 2026c.

Jialv Zou, Shaoyu Chen, Bencheng Liao, Zhiyu Zheng, Yuehao Song, Lefei Zhang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Diffusiondrivev2: Reinforcement learning-constrained truncated diffusion modeling in end-to-end autonomous driving. *arXiv preprint arXiv:2512.07745*, 2025.

A Detailed Reward Computation

A.1 Trajectory Sampling and Simulator Quantities

The policy predicts eight ego-relative waypoints $\tau = \{(x_j, y_j, h_j)\}_{j=1}^{8}$ at $0.5\,\mathrm{s}$ intervals, corresponding to a $4\,\mathrm{s}$ prediction horizon. Trajectory scoring uses the NAVSIM PDM simulator with $\Delta t = 0.1\,\mathrm{s}$ and 40 future simulation steps. The continuous safety computations include the initial state, giving the evaluation grid $\mathcal{T} = \{0, 0.1, \ldots, 4.0\}\,\mathrm{s}$. All geometric quantities below are computed from the simulated ego states and the scene-specific cached map and object occupancies, rather than only from the eight predicted waypoints.

Let E_t denote the ego footprint polygon at time t and $\mathcal{O}_t$ the retained obstacle polygons at that time. The obstacle set excludes red-light proxy geometries and tracks in the simulator-provided collided-track exclusion list. Hard safety is obtained from the NAVSIM NC, DAC, and TTC metrics. For numerical robustness, the reward implementation checks whether each of these metrics is at least $1 - 10^{-6}$ when evaluating $F(\tau)$. The continuous margins do not replace these benchmark safety checks.

A.2 Geometric Safety Margins

Obstacle clearance. For two polygons A and B, the implementation uses the signed clearance surrogate

$$\delta(A, B) = \begin{cases} \mathrm{dist}(A, B), & A \cap B = \varnothing, \\ -\sqrt{\mathrm{area}(A \cap B)}, & A \cap B \neq \varnothing. \end{cases} \tag{11}$$

Here, dist is the minimum Euclidean separation between the polygons. Intersections receive a non-positive value determined by the square root of their overlap area; boundary-only contact has value zero. This is an overlap-area surrogate, not an exact penetration depth. The minimum obstacle-clearance margin is

$$m_{\mathrm{clr}}(\tau) = \min_{t \in \mathcal{T}} \min_{O \in \mathcal{O}_t} \delta(E_t, O). \tag{12}$$

Projected TTC clearance. The continuous TTC margin is a *geometric clearance in meters*, not a time-to-collision value in seconds. At each simulated state, let v_t be the magnitude of the ego velocity and h_t its heading. The ego footprint is translated at constant speed and heading over the projection offsets $\mathcal{H} = \{0, 0.3, 0.6, 0.9\}\,\mathrm{s}$:

$$\widetilde{E}_{t,\Delta} = \mathrm{Translate}\left(E_t,\, \Delta v_t \begin{bmatrix} \cos h_t \\ \sin h_t \end{bmatrix}\right), \qquad \Delta \in \mathcal{H}. \tag{13}$$

The projected clearance is evaluated against obstacle occupancies at $t + \Delta$:

$$m_{\mathrm{ttc}}(\tau) = \min_{\substack{t \in \mathcal{T},\ \Delta \in \mathcal{H} \\ \text{valid projection}}} \min_{O \in \mathcal{O}^{\mathrm{ttc}}_{t,\Delta}} \delta(\widetilde{E}_{t,\Delta}, O). \tag{14}$$

The TTC-specific obstacle set retains objects ahead of the ego vehicle. When the ego vehicle occupies multiple lanes, a non-drivable area, or an intersection, objects that are not behind the ego vehicle are also retained. Objects with unavailable classification metadata are retained rather than silently removed. States below the scorer's stopped-speed threshold are skipped; the extension uses a fallback threshold of $5 \times 10^{-3}\,\mathrm{m/s}$ when that configuration field is absent. Projection offsets without an available future occupancy are skipped.

Drivable-area margin. Let $\mathcal{D}$ be the union of the map polygons belonging to roadblocks, intersections, drivable areas, and carpark areas. For a point p, define

$$\delta_{\mathcal{D}}(p) = \begin{cases} \mathrm{dist}(p, \partial\mathcal{D}), & p \in \mathcal{D}, \\ -\,\mathrm{dist}(p, \mathcal{D}), & p \notin \mathcal{D}. \end{cases} \tag{15}$$

Boundary points are treated as covered by $\mathcal{D}$ and have zero margin. If $\mathcal{C}(E_t)$ denotes the four ego-footprint corners, then

$$m_{\mathrm{road}}(\tau) = \min_{t \in \mathcal{T}} \min_{p \in \mathcal{C}(E_t)} \delta_{\mathcal{D}}(p). \tag{16}$$

Thus, the road margin is evaluated at the four corners, not only at the ego center. Taking the union before measuring the margin avoids treating internal boundaries between adjoining drivable polygons as road edges.

A.3 Violation Times and Score Calibration

The time component is obtained independently from the corresponding simulator safety event. Specifically, t_{clr} is the time of the first at-fault collision, t_{ttc} is the first TTC-infraction time returned by the scorer, and t_{road} is the first simulated time marked as non-drivable. If an event does not occur, its time is set to $T = 4\,\mathrm{s}$. All event times are clipped to $[0, T]$. In particular, t_{ttc} denotes when the TTC safety check first fails along the rollout, not the projected time at which contact would occur. Likewise, t_{clr} uses the at-fault collision criterion, whereas m_{clr} measures geometric clearance to retained obstacles.

For $k \in \{\mathrm{clr}, \mathrm{ttc}, \mathrm{road}\}$, we compute

$$\bar{m}_k = \mathrm{clip}\left(\frac{m_k - l_k}{u_k - l_k}, 0, 1\right), \qquad \bar{t}_k = \mathrm{clip}\left(\frac{t_k}{T}, 0, 1\right). \tag{17}$$

Table 6 gives the normalization intervals, factor weights w_k, and time-mixture coefficients ρ_k.

Table 6: Continuous safety-score parameters. All margin bounds, including the TTC-clearance bounds, are measured in meters.

Safety factor k	l_k	u_k	w_k	ρ_k
Obstacle clearance	-0.50	2.00	0.50	0.25
Projected TTC clearance	-0.25	1.00	0.40	0.25
Drivable-area margin	-0.50	0.50	0.10	0.20

With these settings, the continuous safety score is

$$\begin{aligned} S(\tau) = {} & 0.5\big(0.75\bar{m}_{\mathrm{clr}} + 0.25\bar{t}_{\mathrm{clr}}\big) \\ & + 0.4\big(0.75\bar{m}_{\mathrm{ttc}} + 0.25\bar{t}_{\mathrm{ttc}}\big) \\ & + 0.1\big(0.80\bar{m}_{\mathrm{road}} + 0.20\bar{t}_{\mathrm{road}}\big). \end{aligned} \tag{18}$$

The implementation normalizes the factor weights by their sum, which is already one for the configuration above. Each component and the final safety score are clipped to $[0, 1]$. If no retained obstacle or valid TTC projection yields a finite margin, the resulting positive infinity is replaced by the corresponding upper normalization bound. Negative infinity or a NaN margin is replaced by the lower bound. An empty drivable-area union therefore receives the lower road-margin bound. For a covered point with an empty boundary, the upper road-margin bound is used.

Progress signal. The progress signal is

$$P(\tau) = \mathrm{EP}(\tau). \tag{19}$$

EP is read directly from the candidate's normalized progress metric in the NAVSIM scorer; it is not recomputed as raw displacement or trajectory length. No additional SFT-reference subtraction is applied. This does not remove the internal PDM reference used by the benchmark scoring procedure.

Invalid outputs. The reward parser extracts the first `<waypoint>` block and requires exactly eight finite (x, y, h) states. Automatic heading inference is disabled in the training launch configuration. Under the parser's default coordinate bound, any waypoint with $|x| > 200\,\mathrm{m}$ or $|y| > 200\,\mathrm{m}$ is rejected. Malformed outputs and failed scoring requests receive reward -5.0. This is an invalid-output penalty, separate from the safety score assigned to a valid but unsafe trajectory.

B Failure Diagnosis and Correction-Target Construction

B.1 Structured Diagnosis

The failure-mining script selects rollouts with $\mathrm{NC} = 0$ or $\mathrm{DAC} = 0$. For each selected rollout, collision and drivable-area events are localized on the $0.1\,\mathrm{s}$ simulation grid. The collision diagnosis

records the first ego at-fault collision. The implemented classification includes active front collisions and collisions with stopped tracked objects; active lateral collisions are classified as at-fault when the ego vehicle occupies multiple lanes or a non-drivable area.

The collision object is matched to its annotation in the current frame. Its reported relative position is its *current* position in the ego coordinate system, not its future position at collision time. The current 3D box is transformed into the front-camera frame and projected using the camera calibration. The image-space bounding box is formed from positive-depth corners and clipped to the image extent. A collision sample is discarded when the current annotation, required relative position, camera data, or valid projected box is unavailable. This geometric visibility test does not include an additional occlusion test.

For an off-road event, the exit side is determined by comparing the numbers of left and right ego corners outside the drivable area at the first violating state. The side with more outside corners is selected. When the counts are equal, the implementation uses the sign of the ego's lateral displacement relative to its initial pose; non-negative values are assigned to the left side and negative values to the right side. This tie-breaking rule is distinct from the corner-membership test.

B.2 Trajectory Features and Search Budget

We construct the human trajectory bank from NAVTRAIN and partition it into $K = 256$ clusters using K-Means. Clustering and correction-target retrieval use the same 24-dimensional trajectory representation defined below. Both the feature normalization statistics and the cluster centers are computed from the training trajectory bank.

Let $\widehat{h}_j$ be the unwrapped heading sequence and define $\widetilde{h}_j = \widehat{h}_j - \widehat{h}_1$. The feature state at waypoint j is $z_j = (x_j, y_j, \widetilde{h}_j)$. Using the coordinate-wise means μ_d, scales σ_d, and feature weights ω_d stored with the clustering model, the trajectory representation is

$$\phi(\tau) = \operatorname{vec}\left[\omega_d \frac{z_{j,d} - \mu_d}{\sigma_d}\right]_{j=1,\ldots,8;\, d=1,2,3} \in \mathbb{R}^{24}. \tag{20}$$

The same three-coordinate scaler and weights are reused at every waypoint and for both the failed trajectory and the candidate bank. The heading reference is the first predicted waypoint; the longitudinal and lateral coordinates are not additionally shifted to the first waypoint before applying the stored scaler. Candidate distances are Euclidean distances in this feature space.

The assigned cluster is searched first. The remaining clusters are ordered by the Euclidean distance from the failed trajectory's feature vector to each cluster center. Within a cluster, candidates are evaluated in ascending feature distance and re-scored in the current failure scene. The retrieval test accepts a candidate only when its NC, DAC, and TTC scores are all equal to one; total PDMS is not used by this retrieval test. The first accepted candidate terminates the search. Consequently, this procedure gives the closest safe candidate in the first successful cluster under the search budget, rather than a globally nearest safe trajectory over the entire bank.

The search launch configuration caps the search at $1{,}000$ candidate evaluations per failed trajectory, shared across all searched clusters rather than allocated separately to each cluster. Attempted evaluations that raise a scoring error also count toward this budget. A case is left without a correction target if the budget is exhausted or all available candidates fail the safety test. The implementation does not exclude a bank trajectory solely because it originates from the same training sample unless the optional exclusion flag is enabled.

B.3 Post-Retrieval Filtering and SFT Serialization

After retrieval, the dataset-construction configuration retains only records with a successfully retrieved candidate. It additionally rechecks NC and DAC against one with an absolute tolerance of 10^{-8}. The TTC constraint is enforced by the retrieval stage described above.

Records are matched by the scene token and rollout index. When duplicate search records exist for the same key, the candidate with the smaller feature distance is retained, with larger PDMS used only to break an exact distance tie. The builder excludes records without a failure description, a valid trajectory, the original prompt, or an input image.

Table 7: Hyperparameters for Base SFT and Correction SFT. Entries spanning both stage columns indicate shared settings.

Hyperparameter	Base SFT	Correction SFT
Learning rate	2×10^{-5}	1×10^{-5}
Training epochs	2	1
Vision encoder	Frozen	
Language model	Trainable	
Multimodal aligner	Trainable	
Numerical precision	bfloat16	
Number of GPUs	8	
Per-device training batch size	8	
Gradient accumulation steps	1	
Nominal effective batch size	64	
Warmup ratio	0.03	
Weight decay	0.1	
Maximum sequence length	4,096	
Gradient checkpointing	Enabled	
Attention implementation	FlashAttention-2	
Distributed training	DeepSpeed ZeRO-3	
Length-based grouping	Enabled	
Dataset preprocessing workers	32	
Data-loader workers	8	

The SFT input preserves the original driving context and appends the failed trajectory together with the collision/off-road assessment and correction request. The target response contains the collision diagnosis, the drivable-area diagnosis, and then the corrected trajectory inside `<waypoint>` tags. Failed and corrected trajectories are serialized with two decimal places per state coordinate. Event times and relative distances are also written with two decimal places, while image bounding-box coordinates are rounded to integers. This input–output format is used only for Correction SFT, not for inference.

C Training Details and Hyperparameters

C.1 Supervised Fine-Tuning

Base SFT initializes from the pretrained Qwen3.5-4B model and learns trajectory prediction on NAVTRAIN. Correction SFT initializes from the resulting base SFT checkpoint and uses the failure-specific supervision described in the main text. Base SFT uses a learning rate of 2×10^{-5} for two epochs, whereas Correction SFT uses 1×10^{-5} for one epoch. The two stages share the remaining optimization and implementation settings listed in Table 7.

Both stages use `swift sft` with full tuning of the unfrozen modules: the vision encoder is frozen, while all parameters of the language model and multimodal aligner are trainable. Training uses eight GPUs with a per-device batch size of eight and one gradient accumulation step, giving a nominal effective batch size of $8 \times 8 \times 1 = 64$ examples per optimizer step.

No validation subset is automatically split from the supplied SFT dataset, and evaluation during SFT is disabled. Training statistics are logged every five steps. Model-only checkpoints are saved at the end of each epoch, with at most one checkpoint retained.

C.2 GRPO

Table 8 summarizes the GRPO configuration. GRPO uses full-parameter optimization in bfloat16. For the Correction-SFT ablation, each variant is subsequently optimized with the same Safety-Layered GRPO objective. Reward evaluation is performed by a separate NAVSIM service using the trajectory sampling and safety-score settings in Appendix A.

Table 8: GRPO hyperparameters. The per-device batch size is the value specified in the training configuration.

Hyperparameter	Value
Trainable parameters	Full-parameter tuning
Numerical precision	bfloat16
Number of GPUs	8
Training epochs	4
Learning rate	5×10^{-7}
Per-device training batch size	16
Gradient accumulation steps	1
Completions per instance G	8
GRPO num iterations	1
KL coefficient β	0.04
Warmup ratio	0.05
Sampling temperature	1.0
Top-p	1.0
Top-k	Disabled (-1)
Maximum completion length	512 tokens
Distributed training	DeepSpeed ZeRO-3
Generation backend	Colocated vLLM
vLLM tensor parallel size	1
vLLM GPU memory utilization	0.3
vLLM maximum model length	10,240 tokens

D LIMITATIONS

Simulator Dependence and Diagnosis Coverage. Reliable Diagnosis and correction verification rely on simulator states, map geometry, and object annotations during training. Their reliability is therefore conditional on the fidelity of these inputs and the simulator's safety criteria. The current diagnosis focuses on the first at-fault collision and drivable-area violation, rather than providing comprehensive explanations of all driving failures. Collision samples without a valid projection of the involved object into the current front-view image are excluded, while the geometric projection itself does not explicitly account for occlusion. Extending this supervision to additional failure types and less directly observable hazards remains future work.

Coverage and Optimality of Correction Retrieval. Minimum-Correction Target Retrieval is constrained by the coverage of the human trajectory bank, the chosen feature representation, and the search budget. Failures without a retrieved target, or whose targets fail post-retrieval filtering, are excluded from Correction SFT. Consequently, corrective supervision covers only a selected subset of the policy's failures, and unsuccessful retrieval does not imply that no feasible correction exists. Expanding candidate coverage and improving search efficiency are potential directions for addressing this limitation.