AUGUSTUS: AUTONOMOUS DRIVING VLA MODEL WITH UNIFIED GENERATION AND UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models have shown strong potential for end-to-end autonomous driving by leveraging world knowledge and understanding to achieve great performance in long-tail scenarios. However, current VLAs suffer from two major limitations: 1) Degeneration into vision-action (VA) models. To achieve low latency, regression-based and flow-based methods directly generate actions, yet they do not sufficiently leverage the VLM's understanding, leading to unsafe and unexplainable actions. 2) Autoregressive high latency. Autoregressive methods sequentially generate chain-of-thought (CoT) and action tokens, fully leveraging the VLM's understanding, but high latency makes deployment difficult. To enable low-latency deployment while properly using CoT to assist action generation, we propose **AUGUstus**, an **A**utonomous-driving VLA model with **U**nified **G**eneration and **U**nderstanding. Specifically, we first unify diffusion-based action generation and autoregressive CoT understanding in a single model, where understanding is optimized only to serve generation. More concretely, joint CoT supervision preserves the VLM's spatial understanding in the shared representation and keeps it useful for safer and more accurate trajectories. Second, we propose **AUGUstus-mode**, which orders the action generation before the CoT so that the CoT supervises the reliable action during training while inference needs only the action output, enabling low latency. Finally, we introduce **AUGUstus-GRPO**, a unified GRPO algorithm that jointly optimizes the action and CoT, further improving action performance. Extensive experiments demonstrate that **AUGUstus** achieves 94.6 PDMS on NAVSIMv1 and 90.7 EPDMS on NAVSIMv2, establishing state-of-the-art (SOTA) planning performance without scorer module. On nuScenes, our method achieves an average L2 error of 0.27 m and an average collision rate of 0.05%, both the best among the compared methods. To validate the practicality of **AUGUstus**, we train it on our in-house dataset using 512 NVIDIA H800 GPUs for approximately two days, then evaluate it on an internal 1,000 km dataset and preliminarily deploy it on a real vehicle, achieving 12 Hz operation.

1 INTRODUCTION

Autonomous driving has gradually moved from modular stacks for perception, prediction, and planning toward end-to-end models to reduce error accumulation, including ST-P3 (Hu et al., 2022), UniAD (Hu et al., 2023), and VAD (Jiang et al., 2023). Yet their limited model capacity and training data restrict the world knowledge available to the policy, making long-tail scenes and semantically complex interactions difficult to handle reliably.

To address long-tail scenarios, many recent studies have explored large vision-language models (VLMs) as a promising foundation model for autonomous driving, leveraging their large-scale pretraining to supply broad visual concepts, language priors, and reasoning capabilities (Zhao et al., 2026; Xiong et al., 2026; Zhou et al., 2026b; Ye et al., 2026b). However, current mainstream autonomous driving VLA architectures still exhibit three limitations: **1) Regression head:** Despite low latency, these methods (Zhao et al., 2026; Li et al., 2025) produce no action distribution, making RL post-training difficult; moreover, they do not fully leverage VLM knowledge, resulting in low-quality generated actions (Fig. 1a). **2) Diffusion or flow-matching expert:** To address the lack

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.

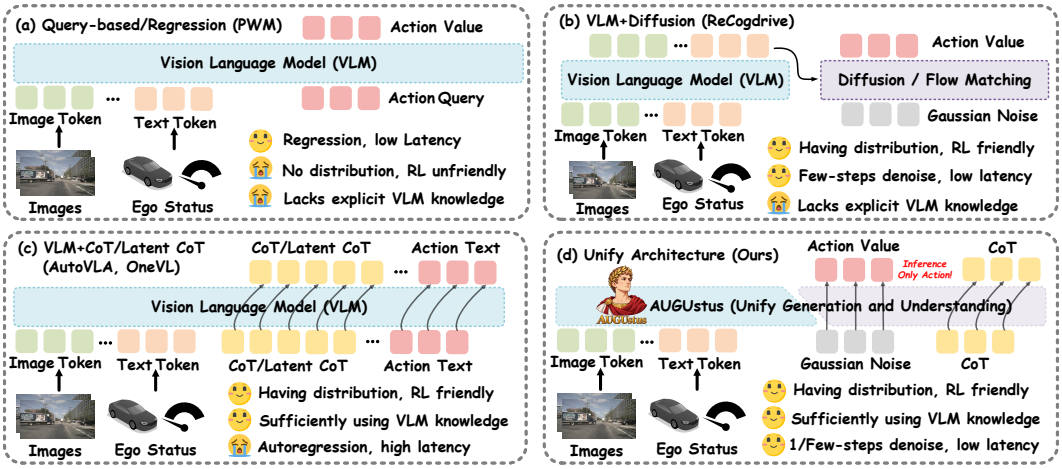

Figure 1: **The difference between AUGUstus and other autonomous-driving VLA paradigms.** Our method unifies generation and understanding under AUGUstus-mode, so that joint CoT supervision preserves the VLM's understanding in the shared representation, yielding safer, more accurate actions with an RL-friendly action distribution and low-latency deployment.

of action distribution for RL algorithm, some VLA methods (Xiong et al., 2026) adopt diffusion or flow-matching experts for action generation, but they still do not fully exploit the VLM's spatial understanding, thereby producing unsafe and unexplainable actions (Fig. 1b). **3) Autoregressive decoding:** To make full use of VLM understanding for action generation, some VLAs (Zhou et al., 2026b; Ye et al., 2026b) serially generate CoT and action, which incurs high latency and makes real-time deployment difficult (Fig. 1c). *Overall, current VLAs struggle to reconcile the VLM's understanding with low-latency control. This raises a key question: can we design an architecture that keeps this action-relevant understanding while remaining low-latency and easy to deploy?*

We argue that these limitations lie in the separation between VLM-based understanding and action generation, rather than treating them as a unified process. Autonomous driving requires understanding and action generation to work in synergy, where accurate spatial understanding enables reliable action generation. However, conventional approaches decouple them: direct regression and diffusion/flow experts degenerate into Vision-Action (VA) models that produce unsafe and unexplainable actions, whereas autoregressive CoT leverages understanding but serializes reasoning before action, causing high latency. Accordingly, we reconsider the role of CoT understanding in autonomous driving. Rather than treating CoT as an independent reasoning stage or a serial prefix to actions, CoT should be jointly optimized with action generation within a unified architecture, where action generation is the core and CoT serves to improve it. Such a unified formulation keeps VLM spatial understanding coupled to and useful for the action policy, while inference only outputs actions, preserving an RL-friendly action distribution and low latency.

To address above fundamental limitations, we propose **AUGUstus**[1], an **A**utonomous-driving VLA model with **U**nified **G**eneration and **U**nderstanding (Fig. 1d). Our objective is action-centric: understanding is not trained for its own sake but to generate a safer and more accurate trajectory. Specifically, (1) we unify diffusion-based action generation and autoregressive CoT understanding in a single model, where joint CoT supervision preserves the VLM's spatial understanding in the shared representation for better action generation; (2) we propose **AUGUstus-mode**, an action-first and CoT-second ordering. This forces the model to learn a CoT that benefits action performance, while inference only needs the action output, thereby maintaining low latency. We explore various CoT designs and find that a CoT describing the specific positions and future orientations of other agents in the scene is most effective for action; and (3) we also introduce **AUGUstus-GRPO**, a unified GRPO algorithm that jointly optimizes the continuous action and CoT streams so that reasoning serves planning, further improving planning capabilities.

[1] AUGUstus refers to Augustus (Octavian), the first Roman emperor, whose consolidation of Roman authority symbolizes unification. Here, it denotes a unified generation and understanding VLA model.

We extensively evaluate **AUGUstus** on closed-loop NAVSIM and open-loop nuScenes. **AUGUstus** achieves 94.6 PDMS on NAVSIMv1 and 90.7 EPDMS on NAVSIMv2, without a scorer module. On nuScenes, our method achieves an average L2 error of 0.27 m and an average collision rate of 0.05%, both the best among the compared methods. To validate the practicality of **AUGUstus**, we train it on our in-house dataset using 512 NVIDIA H800 GPUs for approximately two days, then evaluate it on an internal 1,000 km dataset and preliminarily deploy it on a real vehicle, achieving 12 Hz operation. These results show that the shared model preserves the VLM's spatial understanding, improving action generation while meeting the latency requirements of real-time planning. The main contributions of this paper are summarized as follows:

1. We propose **AUGUstus**, a unified VLA that integrates diffusion-based action generation and autoregressive CoT understanding in one transformer. The shared model preserves the VLM's spatial understanding for action generation, avoiding VLA-to-VA degeneration. **AUGUstus-mode** orders action before CoT to ensure low-latency inference.
2. We introduce **AUGUstus-GRPO**, a unified GRPO algorithm that jointly optimizes action and CoT. It combines JaT-GRPO, which uses closed-form Gaussian importance sampling ratios and KL terms, with CoT-GRPO, which uses token-level GRPO. The reasoning stream is rewarded for serving planning, thereby further improving action performance.
3. We demonstrate that **AUGUstus** achieves state-of-the-art performance on multiple autonomous driving benchmarks, underscoring the effectiveness and generality of our approach. It also demonstrates practical scalability through large-scale training, validation on an internal driving dataset, and real-vehicle deployment.

2 Related Work

VLA models for autonomous driving. Recent driving VLAs follow three designs. *Query-based policies* (Zhao et al., 2026; Liu et al., 2026b; Shao et al., 2026) are low-latency but without action distribution for RL and weakly leveraging VLM semantics, which can yield unsafe and unexplainable actions. *External flow-matching experts* (Xiong et al., 2026) model continuous action distributions but keep the generator outside the VLM, weakening VLM semantics. *Autoregressive policies* (Zhou et al., 2026b; Ye et al., 2026b; Wang et al., 2026) expose semantic reasoning but serialize CoT and action tokens, incurring high latency. These designs struggle to trade off VLM understanding and low latency. **AUGUstus** unifies continuous action tokens and CoT understanding in one transformer under **AUGUstus-mode**, where understanding is trained to serve action generation rather than to maximize its own output quality, and deployment outputs actions directly.

Reinforcement learning for driving VLAs. Driving-oriented VLMs increasingly adopt GRPO-style post-training, such as AutoVLA, AutoDrive-R^2, AutoDrive-P^3, and RecogDrive (Zhou et al., 2026b; Yuan et al., 2026; Ye et al., 2026b; Xiong et al., 2026). RecogDrive directly supervises actions but lacks explicit CoT supervision, making it prone to reward hacking. Other autoregressive methods supervise token-level outputs, which slows both training and inference. In contrast, **AUGUstus-GRPO** jointly optimizes CoT and action under a shared group-relative advantage, with CoT further improving action performance.

3 Methodology

This section introduces **AUGUstus** framework, which unifies generation and understanding within a single transformer. First, we unify continuous, flow-based action generation with autoregressive, token-based CoT understanding in one transformer. Second, we propose **AUGUstus-mode**, an action-first, CoT-second training scheme in which the CoT stream is optimized to supervise and improve action generation. At inference, only the action stream is evaluated, ensuring low-latency deployment. Finally, we introduce **AUGUstus-GRPO**, a unified reinforcement learning algorithm that further improves the VLA's planning capability.

3.1 Unified framework

As shown in Fig. 2, for vision-based end-to-end driving, each sample contains camera video S, ego state E, route command C, and history trajectory H, which we write compactly as the driving

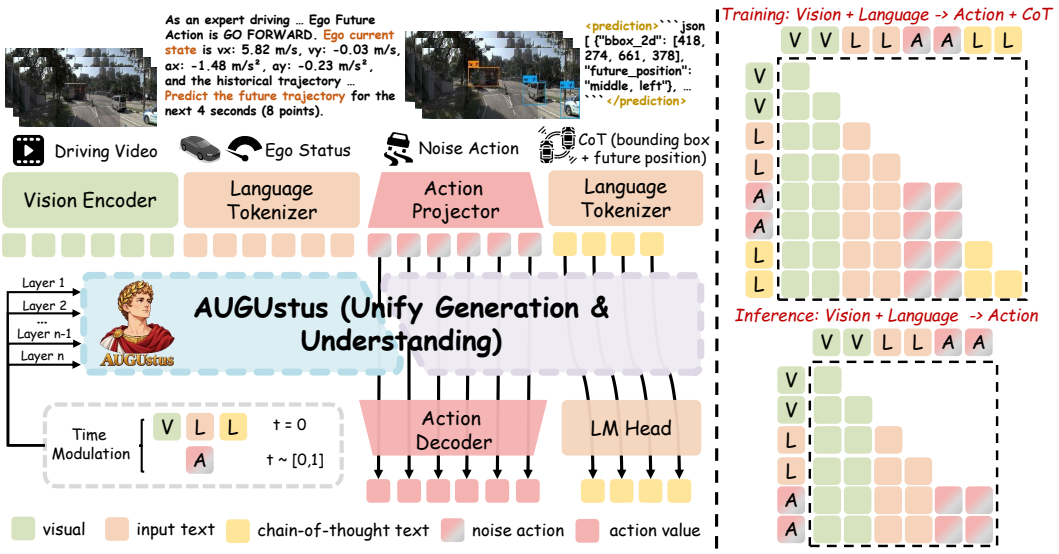

Figure 2: **Overview of AUGUstus.** AUGUstus takes video and ego status as input and uses one shared transformer to first generate clean action and then understand a CoT containing key-object bounding boxes and future position. Action tokens use bidirectional attention and cannot attend to the CoT tokens. At inference, AUGUstus directly outputs the action to ensure low latency.

Algorithm 1 AUGUstus training

Require: Context c, trajectory x_0, CoT y.
Ensure: $\hat{x}_0$, CoT probabilities, updated θ.
1: Noise the target: $x_t = (1-t)x_0 + t\varepsilon$.
2: Joint forward: context, noisy action, teacher-forced CoT.
3: Predict $\hat{x}_0$ and the next CoT tokens.
4: Update with flow-matching and CoT losses.

Algorithm 2 AUGUstus inference (1-step denoise)

Require: Context c, noise x_1, step budget N.
Ensure: Trajectory $\hat{x}_0$; optional post-hoc CoT.
1: Cache the causal context prefix.
2: Denoise the action block in N evaluations.
3: For $N = 1$, return $\hat{x}_0$ directly.
4: Skip CoT at inference; decode it afterward only if requested (e.g., during RL rollout sampling).

context $c = (S, E, C, H)$. During training, as summarized in Algorithm 1, the model additionally receives a noised trajectory x_t and the teacher-forced prefix $y_{<k}$ CoT and predicts both a clean trajectory estimate and the next CoT token:

$$\big(c,\ x_t,\ y_{<k}\big) \longmapsto \Big(\hat{x}_0,\ p_\theta\big(y_k \mid c, x_t, y_{<k}\big)\Big). \tag{1}$$

Here $x_0 \in \mathbb{R}^{T\times 3}$ is the continuous target trajectory, with each waypoint represented by planar position and heading. y is an autoregressive token sequence describing key objects with bounding box and future position. At inference, as summarized in Algorithm 2, only context c is provided. The initial noise state x_1 is denoised in one or few steps denoising to generate $\hat{x}_0$, and CoT decoding is optional to ensure low latency. We denote this action-first, CoT-second ordering as **AUGUstus-mode**. As shown in Fig. 2, its attention design is: context tokens are causal; action tokens form a bidirectional block attending to the full context and to each other; and CoT tokens are causal and additionally attend to the complete action block, while action tokens cannot attend to CoT tokens. Complete procedures are provided in Appendix D.3.

3.2 Supervised Fine-Tuning for AUGUstus

As summarized in Algorithm 1, supervised fine-tuning for **AUGUstus** jointly trains conditional flow matching for actions and teacher-forced language modeling for CoT. Let x_0 be the clean trajectory, $\varepsilon \sim \mathcal{N}(0, I)$ the initial noise, and $t \in [0, 1]$ the flow time:

$$x_t = (1-t)x_0 + t\varepsilon, \qquad \hat{x}_0 = f_\theta(x_t, t, c). \tag{2}$$

Here f_θ is **AUGUstus** model, which directly predicts the clean trajectory $\hat{x}_0$ rather than a velocity, similar to (Li & He, 2025) which predicts the clean image in the image domain. This x_0 parame-

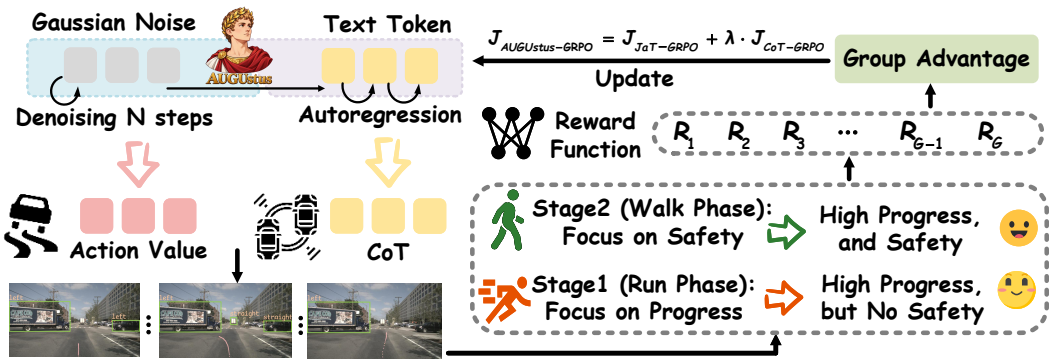

Figure 3: **The pipeline of AUGUstus-GRPO.** Paired x_0 actions and autoregressive CoTs share a group-relative advantage, with closed-form JaT-GRPO and token-level CoT-GRPO. Stage I focuses on progress via PDMS reward, while Stage II focuses on safety via terminal and safety rewards.

Algorithm 3 AUGUstus-GRPO: Stage I

Require: SFT policy and contexts.
Ensure: Progress-tuned policy π_{I}^*.
1: Sample paired action and CoT rollouts.
2: Compute $R_{\mathrm{pred},i}$; set $R_{\mathrm{plan},i} = \mathrm{PDMS}_i$.
3: Normalize the shared group advantage A_i.
4: Jointly update JaT-GRPO and CoT-GRPO.

Algorithm 4 AUGUstus-GRPO: Stage II

Require: Stage-I policy π_{I}^* and contexts.
Ensure: Safety-refined policy π_{II}^*.
1: Sample paired action and CoT rollouts.
2: Keep $R_{\mathrm{pred},i}$; set $R_{\mathrm{plan},i} = R_{\mathrm{end},i} + R_{\mathrm{safe},i}$.
3: Normalize the shared group advantage A_i.
4: Jointly update JaT-GRPO and CoT-GRPO.

terization makes the deployed endpoint explicit, since one Euler step at $t = 1$ returns $\hat{x}_0$ exactly, supporting both few-step sampling and a genuine one-step policy.

The supervised objective combines flow matching for continuous action reconstruction with teacher-forced CoT entropy minimization under the same context c and noise action x_t:

$$\mathcal{L}_{\mathrm{SFT}} = \lambda_{\mathrm{FM}} \underbrace{\mathbb{E}_{t,\varepsilon}[\|\hat{x}_0 - x_0\|_1]}_{\text{flow matching loss}} - \lambda_{\mathrm{CoT}} \underbrace{\mathbb{E}\left[\sum_{k=1}^{L} \log p_\theta(y_k \mid y_{<k}, c, x_t)\right]}_{\text{CoT likelihood}}. \tag{3}$$

During SFT, we note that CoT for prediction, i.e., predicting the bounding boxes and future positions of key objects, is more beneficial for safe and efficient actions than CoT for perception or planning reasoning; additional CoT experiments are provided in the supplementary material.

3.3 Reinforcement Learning for AUGUstus

To further improve the VLA's planning capability, we introduce **AUGUstus-GRPO**, a unified RL method for continuous action and autoregressive CoT. As illustrated in Fig. 3, each rollout produces a paired action and CoT, and the two streams share a normalized group-relative advantage. JaT-GRPO applies GRPO to the continuous action branch under the x_0 parameterization, while CoT-GRPO optimizes the CoT branch with token-level likelihoods.

Unified GRPO objective. For each context, the behavior policy samples G paired rollouts. In stage $s \in \{1, 2\}$, $R_{\mathrm{pred},i}$ evaluates the CoT and $R_{\mathrm{plan},i}^{(s)}$ denotes the corresponding planning reward. We define the shared reward and normalized advantage as

$$R_i^{(s)} = R_{\mathrm{pred},i} + w_{\mathrm{plan}} R_{\mathrm{plan},i}^{(s)}, \qquad A_i = \frac{R_i^{(s)} - \mathrm{mean}_j(R_j^{(s)})}{\mathrm{std}_j(R_j^{(s)}) + \epsilon_0}. \tag{4}$$

For each branch, we compute the importance sampling ratio between π_θ and π_{old}:

$$r_{i,q}^{(\mathrm{action})} = \frac{\pi_\theta(x_{i,t-\Delta t} \mid x_{i,t}, c_i)}{\pi_{\mathrm{old}}(x_{i,t-\Delta t} \mid x_{i,t}, c_i)}, \qquad r_{i,q}^{(\mathrm{CoT})} = \frac{\pi_\theta(y_{i,k} \mid y_{i,<k}, c_i, x_i)}{\pi_{\mathrm{old}}(y_{i,k} \mid y_{i,<k}, c_i, x_i)}. \tag{5}$$

Both branches are optimized with the same clipped GRPO objective,

$$\mathcal{J}_m = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{I}_i^m|}\sum_{q\in\mathcal{I}_i^m}\min\Big(r_{i,q}^{(m)}A_i, \mathrm{clip}(r_{i,q}^{(m)}, 1-\epsilon, 1+\epsilon)A_i\Big)\right], \tag{6}$$

where $m \in \{\text{action}, \text{CoT}\}$, and $\mathcal{I}_i^{\text{action}}$ and $\mathcal{I}_i^{\text{CoT}}$ index the denoising steps and CoT token positions, respectively. Combining the two branches, the overall objective is

$$\mathcal{L}_{\text{AUGUstus-GRPO}} = \underbrace{-\mathcal{J}_{\text{action}} + \beta_{\text{action}}\text{KL}_{\text{act}}}_{\text{JaT-GRPO on } x_0} + \alpha_{\text{CoT}}\underbrace{\left(-\mathcal{J}_{\text{CoT}} + \beta_{\text{CoT}}\text{KL}_{\text{CoT}}\right)}_{\text{CoT-GRPO on language}}. \tag{7}$$

Closed-form JaT-GRPO. The action-policy derivation starts from the linear conditional-flow path $x_t = (1-t)x_0 + t\varepsilon$, with $\varepsilon \sim \mathcal{N}(0, I)$. Following Flow-GRPO (Liu et al., 2026a), we adopt its reverse transition:

$$x_{t-\Delta t} = x_t - \left[v_\theta + \frac{g(t)^2}{2t}\big(x_t + (1-t)v_\theta\big)\right]\Delta t + g(t)\sqrt{\Delta t}\, z, \qquad z \sim \mathcal{N}(0, I). \tag{8}$$

Here v_θ is computed by our predicted clean trajectory $\hat{x}_0^\theta(x_t, t, c)$ through:

$$v_\theta(x_t, t, c) = \frac{x_t - \hat{x}_0^\theta(x_t, t, c)}{t}. \tag{9}$$

Using Eq. (9) and $g(t)^2 = \lambda t$ in Eq. (8) yields a transition with mean $\mu_\theta = c_x x_t + c_0\hat{x}_0^\theta$ and variance $\sigma_s^2 I$. Only $\hat{x}_0^\theta$ is policy-dependent, and the covariance is fixed. Thus, the π_θ, π_{old}, and π_{ref} policies induce Gaussian transitions with the same covariance $\sigma_s^2 I$ and means μ_θ, μ_{old}, and μ_{ref}, respectively. With $\delta_{\text{old}} = \hat{x}_0^\theta - \hat{x}_0^{\text{old}}$ and $\delta_{\text{ref}} = \hat{x}_0^\theta - \hat{x}_0^{\text{ref}}$, the importance ratio and KL simplify to:

$$\log r_t^{(\text{action})} = \tfrac{c_0}{\sigma_s}\langle\delta_{\text{old}}, z\rangle - \tfrac{c_0^2}{2\sigma_s^2}\|\delta_{\text{old}}\|_2^2, \quad \text{KL}_{\text{act}} = \tfrac{c_0^2}{2\sigma_s^2}\|\delta_{\text{ref}}\|_2^2. \tag{10}$$

With the closed-form solution, we plug resulting ratios and KL terms into Eq. (6), enabling optimization with few-step and single-step denoising. The complete derivation is in Appendix B.

Two-stage RL. AUGUstus-GRPO uses a two-stage training procedure inspired by Run-then-Walk scheduling (Ye et al., 2026a). As summarized by Algorithms 3 and 4, Stage I starts from the SFT policy and, following Eq. 4, sets the planning reward $R_{\text{plan},i}^{(1)} = \text{PDMS}_i$ while keeping $R_{\text{pred},i}$. This encourages assertive progress and aligns the policy with the benchmark objective. Stage II starts from the Stage-I checkpoint and sets $R_{\text{plan},i}^{(2)} = R_{\text{end},i} + R_{\text{safe},i}$, preserving the same $R_{\text{pred},i}$. It therefore retains the progress learned in Stage I while refining the trajectory toward accurate endpoints and safe behavior. In both stages, the shared reward and advantage in Eq. 4 jointly update JaT-GRPO and CoT-GRPO; only planning reward is different.

4 Experiments

4.1 Benchmarks and Baselines

Public benchmarks. Following (Zhou et al., 2026b; Xiong et al., 2026; Ye et al., 2026b), we evaluate on the NAVSIM (Dauner et al., 2024) and nuScenes (Caesar et al., 2020) benchmarks. For both benchmarks, the input is a 2-second, 4-frame clip from the front, front-left, and front-right cameras at 2048×512. AUGUstus predicts 8 waypoints over a 4-second horizon for NAVSIM, and 6 waypoints over a 3-second horizon for nuScenes. For NAVSIM, we report the official PDMS on NAVSIMv1 and EPDMS on NAVSIMv2. For nuScenes, following the ST-P3 protocol (Hu et al., 2022), we report L2 error and collision rate.

In-house benchmarks and driving test. To validate the practicality of AUGUstus, we train it on in-house dataset without using any public dataset, using 512 NVIDIA H800 GPUs for approximately two days with a global batch size of 2048. We first evaluate it on in-house 1,000 km driving set covering safety, comfort, efficiency, and compliance metrics, and then deploy the model on a real vehicle for a preliminary driving test in Guangzhou, China.

Table 1: **Performance comparison on NAVSIMv1 benchmark.**

Method	Ref	RL	NC↑	DAC↑	EP↑	TTC↑	Comf.↑	PDMS↑
Human	-	✗	100.0	100.0	87.5	100.0	100.0	94.8
End-to-end small models								
TransFuser (Chitta et al., 2022)	TPAMI'23	✗	97.8	92.6	78.9	92.9	99.9	83.9
PARA-Drive (Weng et al., 2024)	CVPR'24	✗	97.9	92.4	79.3	93.0	99.8	84.0
Hydra-MDP (Li et al., 2024)	CVPR'24	✗	98.3	96.0	78.7	94.6	**100.0**	86.5
WoTE (Guan et al., 2025)	ICCV'25	✗	98.5	96.8	81.9	94.9	**100.0**	88.3
Regression-based VLA planners								
ImagiDrive (Li et al., 2025)	ICRA'26	✗	97.9	95.5	80.7	93.1	99.9	86.4
PWM (Zhao et al., 2026)	NeurIPS'25	✗	98.6	95.9	81.8	95.4	**100.0**	88.1
DriveVLA-W0 (Li et al., 2026)	ICLR'26	✔	98.7	99.1	83.3	95.3	99.3	90.2
Autoregressive VLA planners								
AutoVLA (Zhou et al., 2026b)	NeurIPS'25	✔	98.4	97.5	81.9	98.0	99.9	89.1
AdaThinkDrive (Luo et al., 2025)	ICRA'26	✔	98.4	97.8	84.4	95.2	**100.0**	90.3
AutoDrive-R^2 (Yuan et al., 2026)	ICLR'26	✔	98.5	95.9	82.7	95.4	**100.0**	89.1
AutoDrive-P^3 (Ye et al., 2026b)	ICLR'26	✔	98.9	97.7	83.7	96.6	99.9	90.2
VGGDrive (Wang et al., 2026)	CVPR'26	✗	98.6	96.3	82.9	95.6	**100.0**	88.8
Flow-matching VLA planners								
ReCogDrive (Xiong et al., 2026)	ICLR'26	✔	98.1	97.7	86.5	94.9	**100.0**	90.6
AUGUstus (Ours, 1-step denoising)	-	✔	98.6	98.7	91.4	96.6	**100.0**	94.0
AUGUstus (Ours, 3-step denoising)	-	✔	**98.9**	**98.9**	**91.9**	**97.1**	**100.0**	**94.6**

Table 2: **Performance comparison on NAVSIMv2 benchmark.**

Method	NC↑	DAC↑	DDC↑	TLC↑	EP↑	TTC↑	LK↑	HC↑	EC↑	EPDMS↑ False / True
Human	100.0	100.0	99.8	100.0	87.4	100.0	100.0	98.1	90.1	90.3 / 94.5
Ego Status MLP	93.1	77.9	92.7	99.6	86.0	91.5	89.4	98.3	85.4	64.0 / –
Transfuser (Chitta et al., 2022)	96.9	89.9	97.8	99.7	87.1	95.4	92.7	98.3	87.2	76.7 / 84.0
Hydra-MDP++ (Li et al., 2024)	97.2	97.5	99.4	99.6	83.1	96.5	94.4	98.2	70.9	81.4 / –
DiffusionDrive (Liao et al., 2025)	98.2	96.2	99.5	99.8	87.4	97.3	96.9	**98.4**	**87.7**	84.7 / 88.2
WoTE (Guan et al., 2025)	98.5	96.8	98.8	99.8	86.1	97.9	95.5	98.3	82.9	84.2 / 87.7
ReCogDrive (Xiong et al., 2026)	98.2	97.7	98.4	**100.0**	**89.9**	97.4	90.8	97.7	26.8	82.7 / –
AutoDrive-P^3 (Ye et al., 2026b)	98.9	97.6	98.9	99.8	86.8	**98.5**	95.4	98.3	80.6	85.2 / 88.7
AUGUstus (Ours, 1-step denoising)	**99.0**	**97.9**	**99.7**	99.9	87.3	98.4	**98.0**	98.3	87.2	**87.1 / 90.7**

Baselines. We use Qwen3.5-2B (Qwen Team, 2026) as the base model. On public datasets, we compare against mainstream approaches, including end-to-end small planners and representative autoregressive, regression-based, and diffusion-based VLA planners. On the in-house dataset, we compare AUGUstus with diffusion-based and regression-based architectures. Note that, considering the real-time requirements for deployment, unless otherwise specified, **inference in this paper refers to 1-step denoising without the subsequent CoT generation**.

4.2 Main Results

Public benchmark results. As shown in Table 1, AUGUstus reaches **94.0/94.6 PDMS** with 1/3-step denoising on NAVSIMv1, only 0.2 below human performance (94.8), while only generating a single trajectory without any scorer module. On NAVSIMv2, as shown in Table 2, AUGUstus obtains the SOTA performance, **90.7 EPDMS** with the human-penalty filter (87.1 without it) with only 1-step denoising. On nuScenes, as shown in Table 3, our method with only 1-step denoising achieves **0.27 m** average L2 error and **0.05%** average collision rate.

In-house benchmarks and real driving test. Beyond public benchmarks, we evaluate VLM+flow matching, VLM+regression, and AUGUstus on an internal 1,000 km driving data. We report macro-average across safety, comfort, efficiency, and compliance, where positive values indicate worse

Table 3: **Performance comparison on nuScenes benchmark.**

Method	Ref	L2 (m) ↓				Collision (%) ↓			
		1s	2s	3s	Avg.	1s	2s	3s	Avg.
ST-P3 (Hu et al., 2022)	ECCV'22	1.33	2.11	2.90	2.11	0.23	0.62	1.27	0.71
VAD (Jiang et al., 2023)	ICCV'23	0.17	0.34	0.60	0.37	0.07	0.10	0.24	0.14
UniAD (Hu et al., 2023)	CVPR'23	0.44	0.67	0.96	0.69	0.04	0.08	0.23	0.12
GPT-Driver (Mao et al., 2023)	arXiv'23	0.20	0.40	0.70	0.44	0.04	0.12	0.36	0.17
DriveVLM (Tian et al., 2024)	CoRL'25	0.18	0.34	0.68	0.40	0.10	0.22	0.45	0.27
RDA-Driver (Huang et al., 2024)	arXiv'24	0.17	0.37	0.69	0.40	0.01	0.05	0.26	0.10
OmniDrive (Wang et al., 2024)	CVPR'25	**0.14**	0.29	0.55	0.33	0.01	0.04	0.27	0.11
OpenDriveVLA (Zhou et al., 2026a)	AAAI'26	**0.14**	0.30	0.55	0.33	0.02	0.07	0.22	0.10
AutoVLA (Zhou et al., 2026b)	NeurIPS'25	0.25	0.46	0.73	0.48	0.07	0.07	0.26	0.13
AutoDrive-R^2 (Yuan et al., 2026)	ICLR'26	0.35	0.49	0.62	0.49	0.13	0.44	0.87	0.48
AutoDrive-P^3 (Ye et al., 2026b)	ICLR'26	0.16	0.31	0.56	0.34	**0.00**	0.04	0.20	0.08
AUGUstus (Ours, 1-step denoising)	-	**0.14**	**0.24**	**0.42**	**0.27**	0.01	**0.02**	**0.13**	**0.05**

Table 4: **In-house 1,000 km evaluation.**

Category	VLM+Flow	VLM+Reg.	AUGUstus
Safety	+18.4%	+15.1%	**0.0%**
Comfort	+8.2%	+8.2%	**0.0%**
Efficiency	+3.0%	+3.9%	**0.0%**
Compliance	**-0.5%**	+0.1%	0.0%

Table 5: **Inference time.**

Module/system	Time (ms)
Vision Encoder Stage	40.2
VLA Stage (1-step denoising)	33.2
Total (Locally, no CoT)	**73.6 (13.6 Hz)**
Real vehicle total	**83.0 (12.0 Hz)**

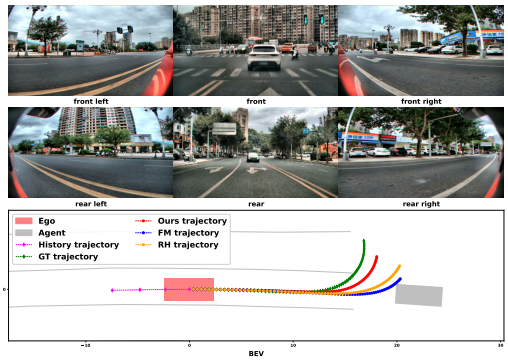

Figure 4: **In-house evaluation visualization.**

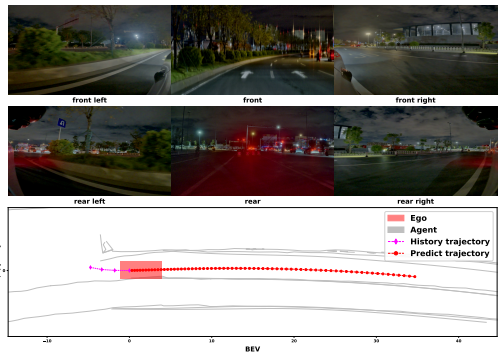

Figure 5: **Real driving test visualization.**

performance and AUGUstus is fixed at 0%. AUGUstus performs best on all four categories, with the largest gaps in safety and comfort. As shown in Table 5, AUGUstus achieves 13.6 Hz on a single H800 and 12.0 Hz on the real vehicle in our preliminary assisted driving tests. Figs. 4 and 5 further visualize its qualitative behavior on the in-house evaluation and real driving test. Further details and supplementary videos are provided in the supplementary material.

4.3 Ablation Study

Ablation Study of CoT and AUGUstus-mode. As shown in Table 6, we ablate CoT and AUGUstus-mode on SFT with NAVSIMv1. With AUGUstus-mode, adding CoT to training makes driving safer and further improves performance. Compared with CoT-first, AUGUstus-mode achieves comparable performance while preserving real-time inference, making the architecture deployable. Overall, our proposed AUGUstus-mode and unified CoT training prove effective.

Ablation Study of AUGUstus-GRPO. As shown in Table 7, with our proposed AUGUstus-GRPO, jointly optimized with JaT-GRPO and CoT-GRPO, stage I improves PDMS by +4 and stage II further improves it by +0.6, achieving a SOTA PDMS of 94.6. These results demonstrate the effectiveness of our proposed unified AUGUstus-GRPO algorithm.

Table 6: **Ablation Study of CoT and AUGUstus-mode.**

Setting	PDMS↑	NC↑	DAC↑	DDC↑	EP↑	TTC↑	Comf.↑	ADE (m)↓	Time (ms)↓
No CoT, Only Action	89.2	98.8	96.8	98.4	83.5	95.8	**100.0**	0.530	**73.60**
CoT first, Action second	**90.6**	99.1	97.8	**98.6**	**84.2**	**96.6**	**100.0**	0.521	4587
AUGUstus-mode (Ours)	**90.6**	**99.0**	**97.9**	98.5	**84.2**	96.5	**100.0**	**0.520**	**73.60**

Table 7: **Ablation Study of AUGUstus-GRPO.**

Setting	PDMS↑	NC↑	DAC↑	EP↑	TTC↑	Comf.↑
SFT with AUGUstus-mode	90.6	**99.0**	97.9	84.2	96.4	**100.0**
+ AUGUstus-GRPO Stage I	94.0	98.6	98.7	91.4	96.6	**100.0**
+ AUGUstus-GRPO Stage II	**94.6**	98.9	**98.9**	**91.9**	**97.1**	**100.0**
+ JaT-GRPO, without CoT-GRPO	92.6	98.0	97.6	91.8	94.1	**100.0**

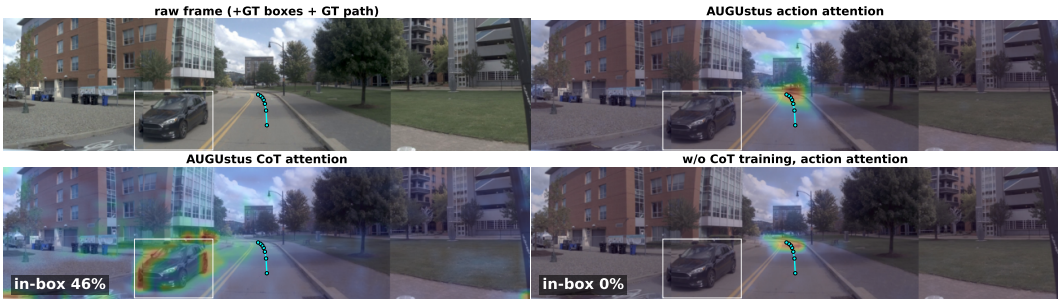

Figure 6: **CoT attention visualization.** CoT supervision anchors the action-relevant attention on the nearby risk object (46% in-box), whereas the no-CoT model misses it (0%).

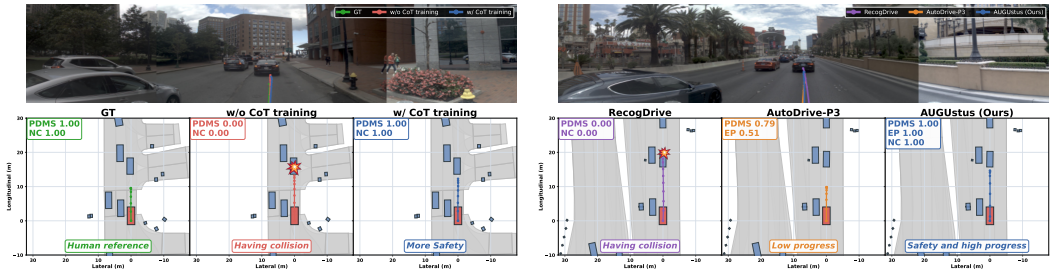

Figure 7: **Visualization.** *Left:* Trajectory visualization with and without CoT. *Right:* Trajectory visualization compared with other methods.

4.4 Visualization

As shown in Fig. 7, the no-CoT model fails to yield to the nearby agent and collides, whereas CoT supervision produces a more safe trajectory. Compared with other methods on the same scene, AUGUstus avoids collision while maintaining high progress.

5 Conclusion and Future Work

In this work, we propose **AUGUstus**, a deployable unified VLA that integrates diffusion-based action generation and autoregressive CoT understanding in a single transformer, with **AUGUstus-mode** ordering for low-latency inference and **AUGUstus-GRPO** for joint optimization. Experiments on NAVSIMv1/v2 and nuScenes show state-of-the-art performance: **AUGUstus** achieves 94.6 PDMS on NAVSIMv1 and 90.7 EPDMS on NAVSIMv2 without a scorer module, and on nuScenes it achieves an average L2 error of 0.27 m and an average collision rate of 0.05%, both the best among the compared methods. To validate its practicality, we train it on our in-house dataset with a large-scale GPU cluster, then evaluate it on an internal driving dataset and preliminarily deploy it on a real vehicle, achieving real-time operation. In future work, we will conduct more extensive real-vehicle tests to further validate the architecture and attempt to apply it to robot VLA.

AI USE STATEMENT

In this work, we used generative AI tools for editing the manuscript and for assisting in literature search to identify relevant papers. We have not used generative AI tools for other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for language polishing. We have reviewed all AI-assisted work: the AI-suggested references were verified through a manual literature survey, and the AI-edited text was checked by the authors for correctness and clarity. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics and all authors have read and adhered to the Code of Ethics. In this study, no human subjects is involved. The use of all datasets, including nuScenes (Caesar et al., 2020) and NAVSIM (Dauner et al., 2024; Cao et al., 2025), follows the relevant usage guidelines and public licenses, ensuring no violation of privacy. We have been careful to avoid any biased or discriminatory results during our research process. No personally identifiable information is used, and no privacy or security concerns will be raised due to our experiments. We are committed to maintaining transparency and integrity throughout the research process.

REPRODUCIBILITY STATEMENT

The training objective and reward definitions appear in Section 3 and Appendix D.1; the algorithm and implementation details are in Appendices D.3 and D.2.

REFERENCES

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.

Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, et al. Pseudo-simulation for autonomous driving. *arXiv preprint arXiv:2506.04218*, 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.

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.

Yanchen Guan, Haicheng Liao, Chengyue Wang, Xingcheng Liu, Jiaxun Zhang, and Zhenning Li. World model-based end-to-end scene generation for accident anticipation in autonomous driving. *Communications Engineering*, 4(1):144, 2025.

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 *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.

Zhijian Huang, Tao Tang, Shaoxiang Chen, Sihao Lin, Zequn Jie, Lin Ma, Guangrun Wang, and Xiaodan Liang. Making large language models better planners with reasoning-decision alignment. In *European Conference on Computer Vision*, pp. 73–90. Springer, 2024.

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.

Jingyu Li, Bozhou Zhang, Xin Jin, Jiankang Deng, Xiatian Zhu, and Li Zhang. Imagidrive: A unified imagination-and-planning framework for autonomous driving. *arXiv preprint arXiv:2508.11428*, 2025.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

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, 2026.

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*, 2024.

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.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *Advances in neural information processing systems*, 38:40783–40818, 2026a.

Qiqi Liu, Huan Xu, Jingyu Li, Bin Sun, Zhihui Hao, Dangen She, Xiatian Zhu, and Li Zhang. Uni-world vla: Interleaved world modeling and planning for autonomous driving. *arXiv preprint arXiv:2603.27287*, 2026b.

Yuechen Luo, Fang Li, Shaoqing Xu, Zhiyi Lai, Lei Yang, Qimao Chen, Ziang Luo, Zixun Xie, Shengyin Jiang, Jiaxin Liu, et al. Adathinkdrive: Adaptive thinking via reinforcement learning for autonomous driving. *arXiv preprint arXiv:2509.13769*, 2025.

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

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

Hao Shao, Letian Wang, Yang Zhou, Yuxuan Hu, Zhuofan Zong, Steven L Waslander, Wei Zhan, and Hongsheng Li. Lmgendrive: Bridging multimodal understanding and generative world modeling for end-to-end driving. *arXiv preprint arXiv:2604.08719*, 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.

Jie Wang, Guang Li, Zhijian Huang, Chenxu Dang, Hangjun Ye, Yahong Han, and Long Chen. Vggdrive: Empowering vision-language models with cross-view geometric grounding for autonomous driving. *arXiv preprint arXiv:2602.20794*, 2026.

Shihao Wang, Zhiding Yu, Xiaohui Jiang, Shiyi Lan, Min Shi, Nadine Chang, Jan Kautz, Ying Li, and Jose M Alvarez. Omnidrive: A holistic llm-agent framework for autonomous driving with 3d perception, reasoning and planning. *CoRR*, 2024.

Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15449–15458. IEEE, 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.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Yuqi Ye, Shangkun Sun, Junhong Lin, Jiayi Zhao, Changhao Peng, Wei Zheng, Guoqing Liu, Tiesong Zhao, and Wei Gao. Sometimes you gotta run before you can walk: Run-then-walk scheduling strategy for vlm autonomous driving, 2026a. URL https://arxiv.org/abs/2609.25831.

Yuqi Ye, Zijian Zhang, Junhong Lin, Shangkun Sun, Changhao Peng, and Wei Gao. $autodrive\text{-}p^3$: Unified chain of perception-prediction-planning thought via reinforcement fine-tuning, 2026b. URL https://arxiv.org/abs/2603.28116.

Zhenlong Yuan, Chengxuan Qian, Jing Tang, Rui Chen, Zijian Song, Lei Sun, Xiangxiang Chu, Yujun Cai, Dapeng Zhang, and Shuo Li. Autodrive-r^2: Incentivizing reasoning and self-reflection capacity for vla model in autonomous driving. In *International Conference on Learning Representations*, volume 2026, pp. 97934–97955, 2026.

Zhida Zhao, Talas Fu, Yifan Wang, Lijun Wang, and Huchuan Lu. From forecasting to planning: Policy world model for collaborative state-action prediction. In *Advances in Neural Information Processing Systems*, volume 38, pp. 134585–134611, 2026.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. In *International Conference on Learning Representations*, volume 2026, pp. 134129–134150, 2026.

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, 2026a.

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, 2026b.

A Supplementary Architecture Details

A.1 Backbone and action projections

Action-module data path. Given a normalized noisy trajectory $x_t \in \mathbb{R}^{8\times 3}$ and flow time $t \in [0,1]$, the module produces a clean trajectory estimate $\hat{x}_0 \in \mathbb{R}^{8\times 3}$ through three stages: an Action Projector, the shared transformer action-token block, and an Action Decoder. The eight rows always correspond to the eight future waypoints; all linear maps are shared across waypoint positions, while positional information is carried by the backbone's continued token positions.

Action Projector. In the left block of Fig. 8, each waypoint is first mapped from three coordinates to the hidden width by a linear layer. In parallel, the scalar flow time is encoded with a sinusoidal sine/cosine embedding. The two representations are concatenated row-wise and passed through a Linear–SiLU–Linear MLP, yielding action-token embeddings $e_{\text{act}} \in \mathbb{R}^{8\times D}$. Thus the time condition is available at the input of every action token without changing the dimensionality of the pretrained language-vision stream.

Shared action-token block. The embeddings e_{act} are inserted as an eight-token action block after the multimodal context. In AUGUstus-mode, the action block reads the cached video, ego-state, route, and language context and is bidirectional within itself; the subsequent causal CoT block may read the action states, but action tokens cannot read CoT tokens. The action states therefore receive scene information from the shared trunk while remaining independent of the optional autoregressive decoding path. During training, this is also the route by which the CoT loss can shape the action representation; during deployment, only the action block is evaluated.

Segmented input sequence and transformer layer. The center of Fig. 8 expands one forward pass into its token segments: visual tokens V, input-language tokens L, noisy action tokens A, and causal chain-of-thought tokens L. The action segment is a single bidirectional block, while the language and CoT segments retain their autoregressive masks. Every decoder layer applies the same residual pattern, $\text{AdaRMSNorm} \to \text{Attention} \to \text{Add} \to \text{AdaRMSNorm} \to \text{MLP} \to \text{Add}$, to the shared hidden states. The time condition is segment-specific: visual and input-language tokens use $t{=}0$, the noisy action tokens use the sampled $t \in [0,1]$, and the CoT tokens again use $t{=}0$. This keeps the diffusion state from contaminating the language stream while allowing every action layer to adapt to the denoising point.

Action Decoder. The right block of Fig. 8 maps the final action hidden states $h_{\text{act}} \in \mathbb{R}^{8\times D}$ back to waypoint coordinates. It applies LayerNorm, a linear projection, SiLU, a second LayerNorm, and a final linear projection to three channels, producing $\hat{x}_0 \in \mathbb{R}^{8\times 3}$. The output is inverse-normalized to metric x, y, and heading coordinates before being sent to the planner. At $t = 1$, the x_0-parameterization makes this decoder output the deployed one-step trajectory directly; for a multi-step ODE/SDE rollout, the output is re-embedded as the next noisy action state and the same shared module is applied again. No separate action transformer or scorer is introduced.

Flow time is injected with per-segment AdaRMSNorm. For hidden state h and condition q, we use

$$\text{AdaRMSNorm}(h;q) = \text{RMSNorm}(h) \odot (1+w) \odot (1+s(q)) + b(q), \qquad [s(q), b(q)] = W_q q, \tag{11}$$

where w is the pretrained RMSNorm weight. Context, language, and boundary tokens use a zero-time condition, while action tokens use the sampled flow time. The positional indices continue from the multimodal context into the action and CoT blocks, so the pretrained positional statistics are preserved.

B Algorithmic Details for JAT-GRPO

B.1 Reverse-time SDE kernel in x0-parameterization

With the convention $t{=}1$ pure noise, $t{=}0$ data, the forward CFM interpolation $x_t = (1-t)x_0 + t\varepsilon$ defines the probability-flow ODE $\dot{x}_t = v_\theta(x_t, t)$ in the direction of decreasing t. Adding Langevin diffusion with coefficient $g(t)$ (the Flow-GRPO construction (Liu et al., 2026a)) gives the reverse-

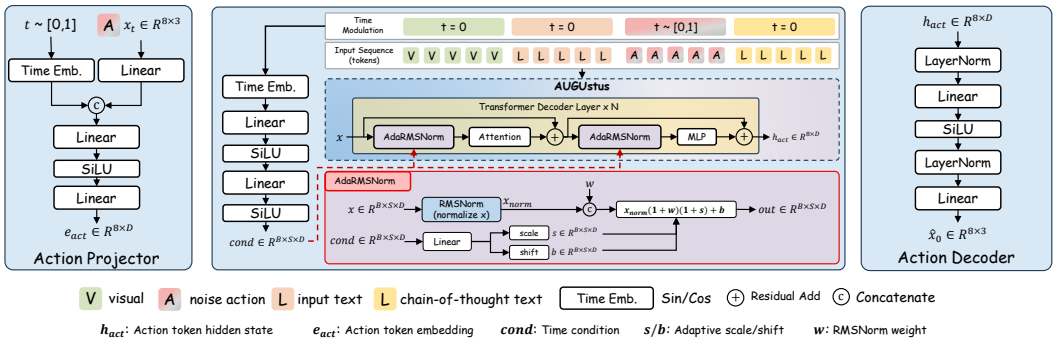

Figure 8: **Detailed AUGUstus action module.** *Left: Action Projector.* A normalized noisy trajectory $x_t \in \mathbb{R}^{8\times 3}$ and a sinusoidal time embedding are projected, concatenated, and passed through Linear–SiLU layers to form $e_{\text{act}} \in \mathbb{R}^{8\times D}$. *Center: shared decoder.* Visual tokens V, input text L, noisy action tokens A, and CoT text share the transformer; the action block is bidirectional, and the segment-specific time modulation uses $t{=}0$ for context/CoT and the sampled t for action tokens. Each layer contains attention, MLP, residual additions, and AdaRMSNorm. The inset expands the adaptive scale/shift s, b and the pretrained RMSNorm weight w. *Right: Action Decoder.* LayerNorm–Linear–SiLU–LayerNorm–Linear decoding maps $h_{\text{act}} \in \mathbb{R}^{8\times D}$ to $\hat{x}_0 \in \mathbb{R}^{8\times 3}$.

Table 8: **Kernel coefficients per schedule** (Eq. (8); one reverse step $t \to t-\Delta t$). The first row names each method and the second row gives its diffusion schedule $g(t)^2$; the remaining rows give the induced kernel coefficients and endpoint behavior. †Flow-GRPO implementations substitute a clamp $\sigma_{\max}$ for σ_s near $t{=}1$.

	Flow-GRPO (Liu et al., 2026a)	DanceGRPO (Xue et al., 2025)	JaT-GRPO (ours)
$g(t)^2$	$a^2\frac{t}{1-t}$	η^2	λt
c_x	$1-\frac{\Delta t}{t}\left(1+\frac{a^2}{2(1-t)}\right)$	$1-\frac{\Delta t}{t}\left(1+\frac{\eta^2}{2t}\right)$	$1-\frac{\Delta t}{t}\left(1+\frac{\lambda}{2}\right)$
c_0	$\frac{\Delta t}{t}\left(1+\frac{a^2}{2}\right)$	$\frac{\Delta t}{t}\left(1+\frac{\eta^2(1-t)}{2t}\right)$	$\frac{\Delta t}{t}\left(1+\frac{\lambda(1-t)}{2}\right)$
σ_s	$a\sqrt{\frac{t\,\Delta t}{1-t}}$	$\eta\sqrt{\Delta t}$	$\sqrt{\lambda t\,\Delta t}$
$t{=}1,\ \Delta t{=}1$	$c_x \to -\infty, \sigma_s \to \infty$ †	$c_x{=}-\frac{\eta^2}{2},\ c_0{=}1,\ \sigma_s{=}\eta$	$c_x{=}-\frac{\lambda}{2},\ c_0{=}1,\ \sigma_s{=}\sqrt{\lambda}$
single-step policy	*undefined*	$\mathcal{N}(\hat{x}_0-\frac{\eta^2}{2}x_1, \eta^2 I)$	$\mathcal{N}(\hat{x}_0-\frac{\lambda}{2}x_1, \lambda I)$
terminal state $t{=}0$	clean	$+\mathcal{N}(0, \eta^2\Delta t\, I)$ corruption	clean

time SDE; its Euler–Maruyama step of size $\Delta t > 0$ (from t to $t-\Delta t$) is

$$x_{t-\Delta t} = x_t - \left[v_\theta + \frac{g(t)^2}{2t}\big(x_t + (1-t)v_\theta\big)\right]\Delta t + g(t)\sqrt{\Delta t}\, z, \tag{12}$$

where $-\frac{1}{t}(x_t+(1-t)v)$ is the score $\nabla \log p_t$ of the linear interpolation path. Substituting the x_0-parameterized velocity $v_\theta = (x_t - \hat{x}_0^\theta)/t$ and collecting x_t and $\hat{x}_0^\theta$ yields the kernel of Eq. (8) with

$$c_x = 1-\frac{\Delta t}{t}\left(1+\frac{g(t)^2}{2t}\right), \qquad c_0 = \frac{\Delta t}{t}\left(1+\frac{g(t)^2\,(1-t)}{2t}\right), \qquad \sigma_s = g(t)\sqrt{\Delta t}.$$

For $g(t)^2 = \lambda t$ these reduce to $c_x = 1-\frac{\Delta t}{t}(1+\frac{\lambda}{2})$, $c_0 = \frac{\Delta t}{t}(1+\frac{\lambda(1-t)}{2})$, $\sigma_s = \sqrt{\lambda t \Delta t}$; at $t{=}1, \Delta t{=}1$: $c_x = -\lambda/2$, $c_0 = 1$, $\sigma_s = \sqrt{\lambda}$, hence Eq. (16).

The action-transition schedules used by GRPO are summarized in Table 8. The proposed schedule is finite at the pure-noise endpoint and vanishes at the clean endpoint, so it supports both one-step deployment and a clean terminal trajectory. The schedule enters the exact ratio and KL only through c_0 and σ_s, so the same implementation covers all three alternatives.

B.2 Closed-form log-ratio and KL

Both kernels share variance $\sigma_s^2 I$, so with $z = (x_{t-\Delta t} - \mu_{\text{old}})/\sigma_s$:

$$\log \rho_t = -\frac{1}{2\sigma_s^2}\Big(\big\|x_{t-\Delta t} - \mu_\theta\big\|^2 - \big\|x_{t-\Delta t} - \mu_{\text{old}}\big\|^2\Big) \tag{13}$$

$$= \frac{1}{\sigma_s^2}\langle \mu_\theta - \mu_{\text{old}},\, x_{t-\Delta t} - \mu_{\text{old}}\rangle - \frac{1}{2\sigma_s^2}\big\|\mu_\theta - \mu_{\text{old}}\big\|^2 = \frac{c_0}{\sigma_s}\langle \delta, z\rangle - \frac{c_0^2}{2\sigma_s^2}\|\delta\|^2. \tag{14}$$

Since $\mathbb{E}[z] = 0$, $\mathbb{E}[\log \rho_t] = -\frac{c_0^2}{2\sigma_s^2}\|\delta\|^2 = -\mathrm{KL}(\pi_{\text{old}}\|\pi_\theta)$ (equal-covariance Gaussian KL). The same derivation applies per coordinate d when σ_s and z are diagonal per-dimension quantities – the form used in our implementation with multivariate noise levels $\lambda_{x,y,\psi}$.

The corresponding reference-policy KL and deployed one-step policy are

$$\mathrm{KL}_{\text{act}} = \frac{c_0^2}{2\sigma_s^2}\|\hat{x}_0^\theta - \hat{x}_0^{\text{ref}}\|_2^2. \tag{15}$$

$$\pi_\theta(x_0 \mid x_1, c) = \mathcal{N}\left(\hat{x}_0^\theta - \frac{\lambda}{2}x_1, \lambda I\right) \quad (t = 1, \Delta t = 1). \tag{16}$$

Scalar parametric form. Writing $m = \delta/t$ (the velocity-space prediction difference) and defining the schedule coefficient $\kappa \triangleq c_0\, t/\sigma_s$, Eq. (14) factorizes per element into the compact quadratic form

$$\log \rho = \kappa\langle m, z\rangle - \tfrac{1}{2}\kappa^2\|m\|^2, \qquad \mathrm{KL} = \tfrac{1}{2}\kappa^2\|m\|^2, \qquad \rho \le \exp\big(\|z\|^2/2\big),$$

with equality of the bound at $m = -z/\kappa$ – the quadratic dependence of $\rho_{\max}$ on the exploration noise is a second, independent motivation for two-sided PPO clipping. For the constant-η schedule, $\kappa = \sqrt{\Delta t}\,(\,\eta^{-1} + \frac{\eta(1-t)}{2t}\,)$; at t=1 this stays finite ($\kappa = \sqrt{\Delta t}/\eta$).

B.3 Gradient decomposition and comparison

Writing $\phi(\theta) \triangleq \log \rho_t$ and differentiating Eq. (14):

$$\nabla_\theta \phi = \frac{c_0}{\sigma_s}\Big(\langle \nabla_\theta \hat{x}_0^\theta,\, z\rangle - \frac{c_0}{\sigma_s}\langle \nabla_\theta \hat{x}_0^\theta,\, \delta\rangle\Big), \tag{17}$$

i.e. an advantage-weighted *noise-following* term (move $\hat{x}_0$ along z when $A_i > 0$) and a *trust-region-like* term pulling $\hat{x}_0$ back toward $\hat{x}_0^{\text{old}}$, which vanishes at the behavior policy; on-policy, $\mathbb{E}_z \nabla\phi = A_i \frac{c_0}{\sigma_s} z$, equivalent to minimizing $\mathcal{L}_{\text{equiv}} = -\sum_t A_i \frac{c_0}{\sigma_s}\langle z, \hat{x}_0^\theta\rangle$. This contrasts with DiffusionNFT (Zheng et al., 2026), whose gradient follows the *velocity difference* direction with a built-in trust region of radius $\propto A_i/\beta$; Flow-GRPO coincides direction-wise with our first term but has no closed-form KL and a diverging t=1 schedule.

B.4 Necessity of $\lambda > 0$ for Single-Step GRPO

When $N = 1$, all rollouts in a group share the same initial condition x_1, so they can differ only in the noise variable z. If $\lambda = 0$, the kernel collapses to a point mass. Consequently, all G samples are identical, $\mathrm{std}_G(R) = 0$, and the advantages are undefined or zero. Therefore, $\lambda > 0$ is necessary. Sufficiency follows because $\mathcal{N}(\mu, \lambda I)$ has full support.

B.5 Exploration Noise Engineering

Per-dimension levels and trajectory smoothing. Exploration noise is applied in the quantile-normalized action space, with per-dimension levels $(\lambda_x, \lambda_y, \lambda_\psi)$. The injected noise is smoothed along the trajectory, keeping perturbed trajectories kinematically plausible. The group shares the initial x_1 state so that advantages compare like-for-like trajectories.

σ-sensitivity of heteroscedastic noise. For fixed prediction shift δ_d in dimension d, Eq. (15) gives per-step KL $\propto \delta_d^2 \lambda_d^{-1}$, while the induced update drift of dimension d scales as $\nabla \propto \delta_d \lambda_d^{-1}$ (gradient)

Table 9: **CoT content ablation (complete metrics and supervision formats).** All variants share the Section 4.1 protocol and are evaluated with the deterministic one-step evaluator on the full navtest split; only the CoT content changes. PDMS is reported together with its full sub-metric breakdown: NC, DAC, DDC, EP, TTC, Comfort, and ADE. NC: no-collision, DAC: drivable-area compliance, DDC: driving-direction compliance, EP: ego progress, TTC: time-to-collision, Comfort: comfort, ADE: average L2 displacement. The default prediction CoT (risk-object boxes + future positions) is **bolded**; the last row is the CoT-first ordering control (identical recipe and prediction content, reverse token order). Inference time is reported once for the shared 73.60 ms action-only path and separately for the CoT-first path.

CoT type	Template	PDMS↑	NC↑	DAC↑	DDC↑	EP↑	TTC↑	Comf.↑	ADE↓	Time (ms)
No CoT	action-only training	89.2	98.8	96.8	98.4	83.5	95.8	100.0	0.530	
Planning	`slow, keep lane, pass after crossing`	90.1	98.9	97.4	98.5	83.8	96.1	100.0	0.522	
Reasoning	perception + future bins + maneuver	89.6	98.7	96.9	98.4	83.4	96.1	100.0	0.534	
Perception	`car [x1,y1,x2,y2]; ped. [x1,y1,x2,y2]`	90.3	98.9	97.5	**98.6**	**84.3**	96.1	100.0	0.517	**73.60**
Perception + Prediction	boxes + future bins, no maneuver sentence	90.3	98.9	97.6	**98.6**	84.1	96.3	100.0	**0.513**	
Short-prediction	atomic vocabulary encoding	89.1	98.7	96.7	98.3	83.1	95.4	100.0	0.559	
Prediction	`car [box] -> near-front; ped. [box] -> far-left`	**90.6**	**99.0**	**97.9**	98.5	84.2	**96.5**	**100.0**	0.520	
CoT first	same as Prediction, CoT tokens precede action	90.6	99.1	97.8	98.6	84.2	96.6	100.0	0.521	4597

and the post-update KL as $\propto \lambda_d^{-2}$; with per-dimension noise levels spanning $\sim 2.5\times$ and time-scaled reweighting across waypoints, the effective step-size ratio across dimensions blows up to $\sim 10^3$ in drift terms, collapsing the effective sample size of the group to a few near-deterministic slots. As a result, we set noise level 5×10^{-2} and groups with $G{=}8$.

C More detail about CoT

This appendix complements the main-text CoT ablation with the complete CoT-content metrics and supervision templates (Table 9) and additional CoT-vs-no-CoT attention statistics. To avoid duplicate reporting, the metric table and the format table are merged: each CoT variant is listed once with its supervision template and full PDMS sub-metric breakdown.

CoT content (Table 9). Removing CoT costs 1.4 PDMS (89.2 vs. 90.6), concentrated in drivable-area compliance (-1.1 DAC) and ego progress (-0.7 EP). This ablation directly tests the VLA-to-VA hypothesis: CoT is useful only when its content is action-relevant and its supervision reaches the shared action representation. Among CoT types, *prediction* beats *perception* ($+0.3$), *planning* ($+0.5$), and their union ($+0.3$): future-position labels carry the operative information, and stacking perception on top adds tokens without information. Free-form *reasoning* is worse (89.6), and a 2573-token-vocabulary atomic *short-prediction* encoding (~ 14 tokens per scene, $7\times$ shorter than the default) even falls below the no-CoT baseline (89.1 vs. 89.2) – aggressive CoT compression trades

away the planning gain entirely. Order analysis confirms the "pre > per/perpre > pla" ranking. The CoT-first ordering control (last row) reaches the same 90.6 PDMS as the action-first default, confirming that the quality gain comes from the CoT supervision itself rather than the ordering; the ordering decides only the deployment cost, as reported in Fig. 6 (left).

C.1 CoT CONTENT AND SUPERVISION FORMATS

C.1.1 WHAT EACH CoT TEACHES

Our CoT ablation is a comparison of three *alternative* supervision targets, not a three-stage pipeline: every variant is trained on the identical scene and action, and only the content of the single CoT block differs. We take the three task-oriented CoT *definitions* from AutoDrive-P^3 (Ye et al., 2026b) – *Perception CoT* (the 2-D boxes of relevant agents), *Prediction CoT* (each box plus a discretized future-position label, such as `near-front` or `far-left`), and *Planning CoT* (textual ego waypoints or maneuvers) – and ask which one makes the action head plan best, since only one of them is emitted per sample. We do not chain them, and the model never consumes its own perception output to produce a prediction or a plan; the three streams are competing ways of describing the same scene, and Table 9 reports the outcome (prediction CoT wins). A fourth variant, *reasoning CoT*, concatenates the three signals into a compact approximately 100-token explanation: it first names the relevant agents, then states their future bins, and finally gives the resulting ego maneuver. This is intentionally shorter and more structured than unconstrained free-form reasoning.

Figure 9 instantiates the three variants on one NAVSIM right-turn scene, using the ground-truth annotation of the released evaluation split. The three panels share the identical scene and the identical agent ordering, so the figure isolates exactly what each CoT content contributes. All coordinates below are reported on the deployment canvas (2048×512), matching the model input coordinate system.

(a) Perception CoT: *what is where*. This variant emits a 2-D bounding box and a semantic class for every agent that can influence ego motion. Here the annotation contains five agents—two vehicles (one in the left lane, one far ahead in the center) and three vulnerable road users (two pedestrians and one cyclist on the right sidewalk)—each drawn with a class-colored box and a label tag in Figure 9(a). The complete answer field is

```
<perception>[{"bbox_2d":[0,265,146,509],"label":"vehicle"},
 {"bbox_2d":[994,222,1085,329],"label":"vehicle"},
 {"bbox_2d":[1954,259,2018,378],"label":"pedestrian"},
 {"bbox_2d":[1676,262,1740,372],"label":"pedestrian"},
 {"bbox_2d":[1728,283,1807,366],"label":"bicycle"}]</perception>
```

All five boxes are the salient agents that can influence ego motion: the two vehicles lie along the ego path, and the three vulnerable road users stand on the sidewalk that the right turn sweeps past. The ego vehicle is moving at 10.35 m/s, so the pedestrians and the cyclist are the agents whose distance the maneuver changes most rapidly. This variant carries location but no motion: it tells the planner where the conflicts currently are, not where they will be.

(b) Prediction CoT: *where each agent will be*. This variant keeps the same boxes and assigns each of them a coarse future relative position over the next 4 s (*e.g. near, right* / *far, front* / *behind*), drawn as a directional arrow and a text tag on every box in Figure 9(b):

```
<prediction>[{"bbox_2d":[0,265,146,509],"future_position":"behind"},
 {"bbox_2d":[994,222,1085,329],"future_position":"far, front"},
 {"bbox_2d":[1954,259,2018,378],"future_position":"near, right"},
 {"bbox_2d":[1676,262,1740,372],"future_position":"near, right"},
 {"bbox_2d":[1728,283,1807,366],"future_position":"near, right"}]</prediction>
```

The boxes share the same order as the perception variant. The vehicle in the adjacent left lane drops to *behind*, the lead vehicle stays *far, front*, and all three vulnerable road users remain *near, right*—the right-side road users are the safety-critical agents for this maneuver. This variant is the one our default model uses: it adds the future bearing of each conflict to its current location, and Table 9 shows that this extra content, and not merely the presence of language, is what lifts PDMS from 89.2 to 90.6.

(c) Planning CoT: *what the ego does*. This variant instead describes the ego maneuver itself, as eight waypoints (x, y, θ) at $0.5\,\text{s}$ intervals over a $4\,\text{s}$ horizon. Figure 9(c) projects the ground-truth trajectory (green) back onto the camera strip and shows the complete top-down arc in a BEV inset; note that the inset is drawn with ego-left on the left, so the right turn bends to the right of the star:

<planning>[{"x_y_radian":[5.22,0.0,-0.01],"label":"0.5s"},
{"x_y_radian":[10.58,-0.07,-0.02],"label":"1.0s"},
{"x_y_radian":[16.04,-0.28,-0.05],"label":"1.5s"},
{"x_y_radian":[21.55,-0.6,-0.07],"label":"2.0s"},
{"x_y_radian":[27.06,-1.02,-0.09],"label":"2.5s"},
{"x_y_radian":[32.57,-1.55,-0.1],"label":"3.0s"},
{"x_y_radian":[38.09,-2.11,-0.1],"label":"3.5s"},
{"x_y_radian":[43.76,-2.69,-0.1],"label":"4.0s"}]</planning>

The eight waypoints trace a gentle, high-speed right curve that advances from $(5.2, 0.0)\,\text{m}$ to $(43.8, -2.7)\,\text{m}$ while the heading rotates from -0.01 to $-0.10\,\text{rad}$, keeping a lateral offset from the sidewalk agents on the right. Because this variant describes the answer rather than the evidence, it is farther from the visual input than (a) and (b): its 90.1 PDMS sits below prediction CoT, consistent with the view that the most useful CoT content is the one that supplies the action head with the operative scene information.

Prediction CoT is the most useful target because it supplies the action head with both the current conflict location and its likely future bearing. Perception alone gives the object but not its motion, while planning text is farther from the visual evidence. Free reasoning is less stable under teacher forcing, and aggressive atomic compression removes useful spatial context. The supervision templates and the complete PDMS sub-metric breakdown (NC, DAC, DDC, EP, TTC, Comfort, and ADE) are reported together in Table 9.

C.2 CoT ATTENTION VISUALIZATION

We visualize the last-frame attention from the final full-attention layer, as shown in Figure 10. With prediction-CoT supervision, the action queries consistently concentrate on nearby vehicles and vulnerable road users that constrain the maneuver, rather than spreading attention over the whole image. More importantly, we observe a clear qualitative difference between models trained with and without CoT. Without CoT supervision, the attention is dominated by the lead vehicle that the ego car is following, while surrounding vehicles receive little attention. This narrow focus becomes problematic in cluttered or safety-critical scenes: nearby hazardous vehicles can be overlooked, which in turn prevents the policy from reacting safely. In contrast, adding CoT supervision broadens the attention of the VLA to a larger set of task-relevant objects, including adjacent vehicles, pedestrians, and cyclists that may affect the maneuver. As a result, the model acquires more comprehensive spatial awareness and produces safer driving strategies. These observations are consistent with the quantitative gains in NC and DAC reported in Table 9.



Figure 9: **The three CoT contents compared on one NAVSIM right-turn scene (ground-truth annotations).** The three rows are three *alternative* supervision targets, not a serial pipeline: each is trained alone on the same scene and action, and the model never consumes one row's output to produce another's. One pastel outer frame encloses the title, annotated scene, and JSON answer of each CoT variant. **(a) Perception CoT**: a 2-D box and a class label for every salient agent (vehicle: blue, bicycle: orange, pedestrian: violet) on the 2048×512 deployment canvas. **(b) Prediction CoT**: each box is annotated with its future relative position and a motion arrow; this is the default target. **(c) Planning CoT**: the ground-truth $4\,\mathrm{s}$ trajectory (green) projected onto the camera strip, with a BEV inset (ego-left on the left) showing the top-down right-turn arc.

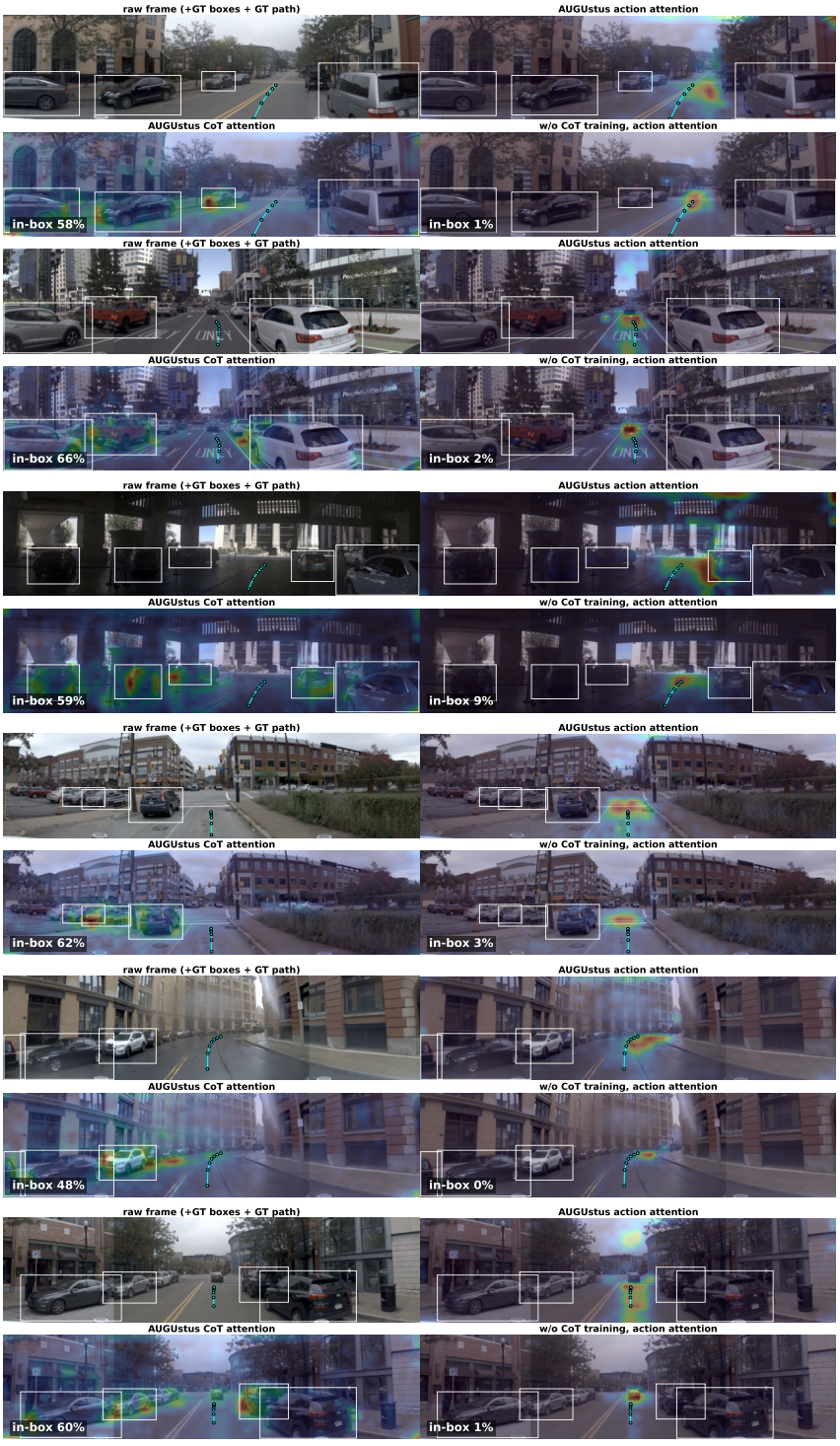

Figure 10: **CoT attention visualization.** After prediction-CoT supervision, action tokens attend more to key objects across these scenes.

C.3 TRAJECTORY VISUALIZATION: COLLISION REDUCTION

The attention evidence is reflected in the trajectories. Training CoT reduces collisions and improves the safety terms in Table 9: NC rises from 98.8 to 99.0, DAC from 96.8 to 97.9, and TTC from 95.8 to 96.5. The following three collision cases show the qualitative counterpart: the no-CoT policy enters the conflict region, whereas the CoT-trained policy identifies the nearby agent and keeps a feasible clearance, as shown in 11.

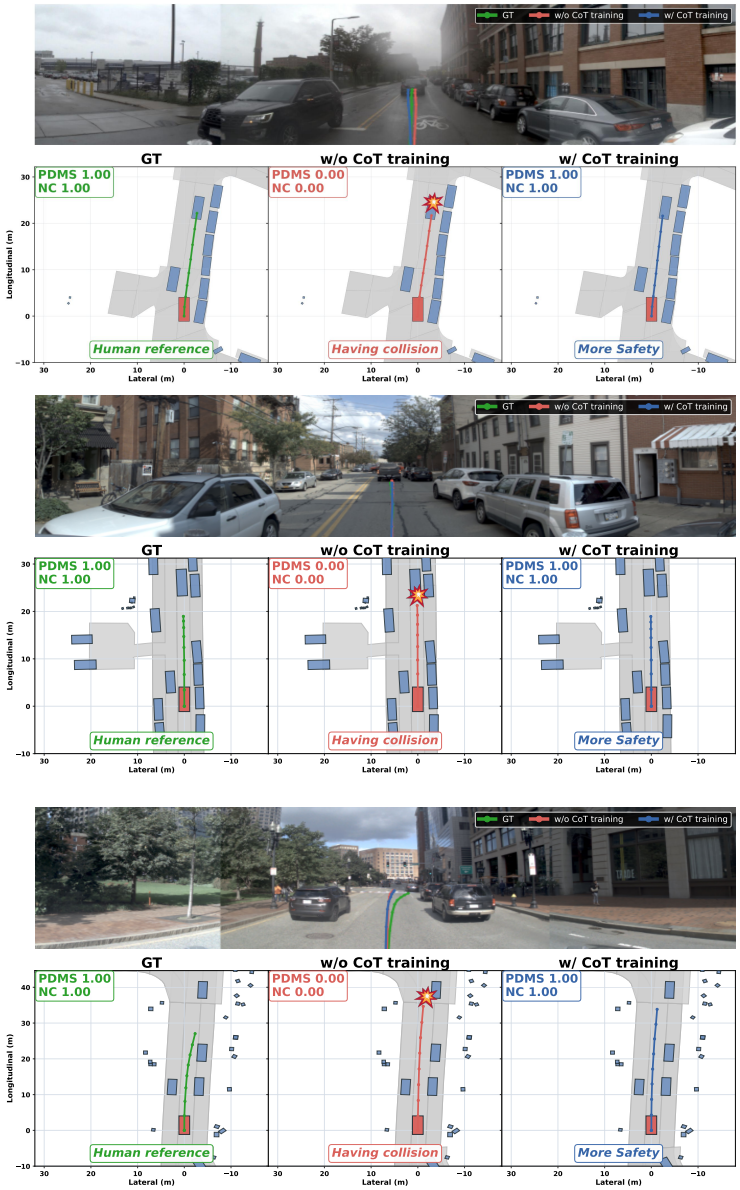

Figure 11: **Collision cases.** Each row compares the no-CoT and prediction-CoT trajectories in the same scene. CoT redirects the action away from the conflict agent, consistent with the improved NC, DAC, and TTC values in Table 9.

C.4 TRAJECTORY VISUALIZATION: PROGRESS PRESERVATION

Safety does not come from stopping. The prediction-CoT model also improves ego progress (EP 83.5 $\rightarrow$ 84.2) and the aggregate PDMS (89.2 $\rightarrow$ 90.6) in Table 9, while retaining full comfort. The three progress cases, as shown in 12, below show that the CoT policy avoids the conflict and continues along the route instead of replacing unsafe motion with an overly conservative halt.

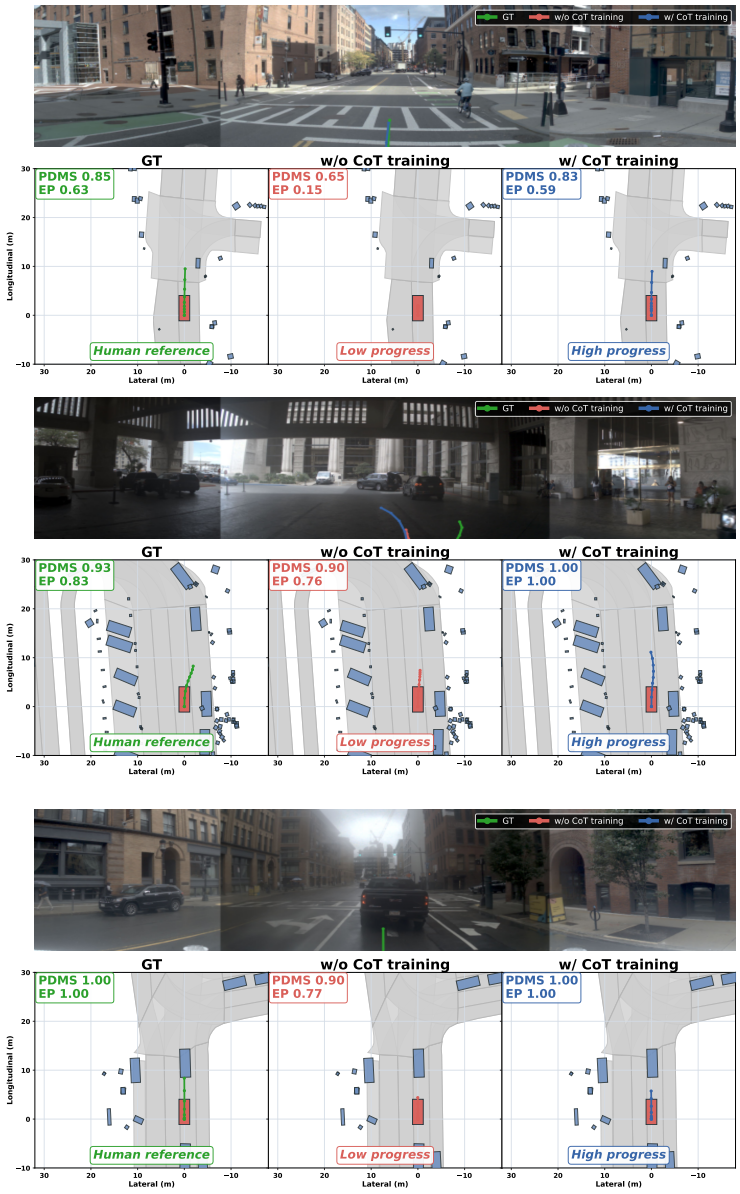

Figure 12: **Progress-preserving cases.** Prediction CoT keeps the route-directed motion while avoiding the nearby agent. The qualitative cases complement the EP and PDMS gains reported in Table 9; they are representative examples rather than a pointwise guarantee.

D ALGORITHMIC DETAILS: SFT, AUGUSTUS-GRPO

D.1 REWARD DESIGN DETAILS AND CURRICULUM RATIONALE

Prediction (CoT) reward. Predicted boxes are greedily matched to ground-truth boxes by descending IoU at threshold 0.5; $r_{\text{det}} = \sum \text{IoU} / \max(\#\text{pred}, \#\text{gt})$ is a soft F1 (penalizing both misses and hallucinations), and r_{fp} is the fraction of matched boxes whose future-position label agrees; $r_{\text{pred}} = r_{\text{det}} r_{\text{fp}}$. A *strict format gate* requires exactly one `<prediction>...</prediction>` block with clean JSON content and no surrounding text or code fences; violations zero r_{pred} (and only r_{pred}).

Planning reward. Stage I uses $R = r_{\text{pred}} + 5 \cdot \text{PDMS}$. Stage II uses $R = r_{\text{pred}} + 5\,(r_{\text{endpoint}} + r_{\text{safe}})$, where r_{endpoint} maps the final-waypoint deviation to $\{1, .8, .6, .4, .2\}$ over progressively looser L1 thresholds (the softest of four tested "soft-error" shapings worked best), and

$$r_{\text{safe}} = \begin{cases} \text{mean}\big(\{\text{TTC}, \text{DAC}\}\big), & \min_{m \in \mathcal{G}} \text{subscore}_m > 0 \\ 0 & \text{otherwise,} \end{cases}$$
$$\mathcal{G} = \{\text{NC}, \text{DAC}, \text{TTC}, \text{EP}\}.$$

with the hard gate additionally zeroing the whole planning term. PDMS is deliberately removed from the stage-II reward for two reasons: (i) late-stage PDMS becomes near-constant within a group (std$\sim$0.02), so its group-relative advantage is dominated by noise, a measured failure mode of dense-score rewards; (ii) direct PDMS optimization slightly trades NC/TTC for EP. The endpoint-plus-shield shaping preserves a dense informative signal (waypoint geometry) but re-imposes safety ceilings; PDMS remains the validation/model-selection metric throughout, so there is no metric/reward mismatch risk in checkpoint choice.

D.2 IMPLEMENTATION DETAILS

D.2.1 SFT TRAINING DETAIL

We adopt Qwen3.5-2B backbone network. The action sequence consists of 8 action tokens with hidden dimension H=2048. The ActionEncoder is a 3-layer SiLU MLP with concatenated sinusoidal time embedding t. The ActionDecoder follows the structure $\text{LN} \to \text{Lin} \to \text{SiLU} \to \text{LN} \to \text{Lin}(3)$ and uses a near-zero final initialization. Each layer applies per-norm modulation projections that map H to $2H$, corresponding to a DiT variant with approximately 8M additional parameters. We apply quantile normalization with (q_{01}, q_{99}) per action dimension. Training uses DeepSpeed ZeRO-2 with bf16 precision. The loss weights are λ_{diff}=10 and λ_{lm}=1. The main model uses dimension weights $(1, 1, 1)$.

For every training example we sample a flow time t and noise ε, construct $x_t = (1-t)x_0 + t\varepsilon$, and run one joint AUGUstus-mode forward. The action head predicts $\hat{x}_0$ while the language head teacher-forces the structured CoT y. The complete objective is

$$\mathcal{L}_{\text{SFT}} = \lambda_{\text{diff}} \mathcal{L}_{x_0} + \lambda_{\text{lm}} \mathcal{L}_{\text{CE}}, \tag{18}$$

where $\mathcal{L}_{x_0}$ is the dimension-normalized flow-matching L1 loss. We use λ_{diff}=10 and λ_{lm}=1 in the default recipe. The CoT labels are generated from the same scene annotation as the action target. Teacher forcing is used only during SFT: the action tokens are visible to the CoT stream, whereas the action tokens never read CoT tokens. Consequently, $\mathcal{L}_{\text{CE}}$ back-propagates through the action states and the shared trunk, but does not create an inference-time dependency on generated text.

D.2.2 RL TRAINING DETAIL

We perform synchronous rollouts based on Verl, where a unified actor serves both rollout generation and policy updates. Each iteration uses 32 prompts with G=8 rollouts per prompt, resulting in 256 sequences. The rollout uses a 3-step SDE with per-dimension diffusion coefficients $g_d(t)^2 = a_d^2 t$, where $a_d \in (0.05, 0.05, 0.02)$. All rollouts in a group share the same initial state x_1. During rollout, the prefix KV is frozen, and the behaviour policy output $\hat{x}_0$ as well as the CoT token log-probabilities are recorded. For the update, we use a single PPO mini-batch that is fully on-policy, with a learning rate of 10^{-6}, a clipping threshold of 0.2 for both heads, and a gradient clipping norm of 1.0.

For a context group c we draw one shared initial noise state x_1 and G independent transition-noise sequences. Each rollout returns an action trajectory x_i and a CoT sequence y_i. Its scalar reward is

$$R_i = R_{\text{pred}}(y_i) + \lambda R_{\text{plan}}(x_i), \qquad A_i = \frac{R_i - \text{mean}_j(R_j)}{\text{std}_j(R_j) + 10^{-6}}, \tag{19}$$

where we set $\lambda = 5$. The same outcome signal trains both modalities. Stage I sets $R_{\text{plan}} = \text{PDMS}$; Stage II replaces this saturating score by the endpoint-plus-safety shaping in Appendix D.1.The action objective is the clipped JaT-GRPO surrogate, and the language objective is token-level clipped CoT-GRPO:

$$\mathcal{J}_a = \mathbb{E}\left[\min(\rho_a A, \text{clip}(\rho_a, 1-\epsilon, 1+\epsilon)A)\right], \tag{20}$$

$$\mathcal{J}_c = \mathbb{E}\left[\frac{1}{|y|}\sum_k \min(\rho_{c,k} A, \text{clip}(\rho_{c,k}, 1-\epsilon, 1+\epsilon)A)\right]. \tag{21}$$

The update maximizes $\mathcal{J}_a + \alpha_c \mathcal{J}_c$ while adding the exact action KL in Eq. (15) and a token-level reference KL for CoT. Thus CoT-GRPO is ordinary PPO on the discrete suffix, but its advantage is tied to the same simulator outcome as the continuous action head.

Algorithm 5 AUGUstus supervised fine-tuning: complete procedure

Require: Pretrained parameters θ; training samples $\mathcal{D} = \{(c, x_0, y)\}$; loss weights and optimizer.
Ensure: Cold-start policy π_{SFT} with action and prediction-CoT outputs.

for each supervised training iteration **do**
  Sample a minibatch of contexts, target trajectories, and prediction CoTs from $\mathcal{D}$.
  Normalize target trajectories using the fixed action statistics.
  Sample flow times t and independent noise $\varepsilon \sim \mathcal{N}(0, I)$.
  Form $x_t = (1-t)x_0 + t\varepsilon$ for each training sample.
  Encode context $c = (S, E, C, H)$, noisy action x_t with time t, and shifted CoT tokens $y_{<k}$.
  Run one shared-transformer forward pass with AUGUstus-mode hybrid attention.
  Read the clean-trajectory prediction $\hat{x}_0$ and the token distribution $p_\theta(y_k \mid y_{<k}, c, x_t)$.
  Compute the joint flow-matching and teacher-forced CoT loss in Eq. (3) and include the auxiliary module losses.
  Backpropagate both losses jointly through the shared transformer and update θ.
  Validate with action-only denoising.
end for
return The selected cold-start policy π_{SFT}.

Algorithm 6 AUGUstus deployment: complete AUGUstus-mode inference

Require: Trained policy π_θ; context $c = (S, E, C, H)$; schedule $1 = t_0 > \cdots > t_N = 0$; optional CoT flag.
Ensure: Future ego trajectory x^{out} and, only if requested, prediction CoT y.

Encode the causal context prefix once and cache its states.
Draw $x \sim \mathcal{N}(0, I)$ in normalized action space.
for $n = 0$ to $N-1$ **do**
  Evaluate the bidirectional action block against the cached context to obtain $\hat{x}_0$.
  Apply the deterministic flow step: $x \leftarrow x - (t_n - t_{n+1})(x - \hat{x}_0)/t_n$.
end for
Invert the fixed action normalization to obtain x^{out}.
if prediction CoT is requested **then**
  Append the denoised action block to the context with zero-time conditioning.
  Autoregressively decode $y_k \sim p_\theta(\cdot \mid y_{<k}, c, x)$ with causal CoT attention.
  return (x^{out}, y).
else
  return x^{out} without CoT decoding.
end if

D.3 Complete Training and Inference Algorithms

Algorithms 5 and 6 give the full cold-start and deployment procedures corresponding to the contrasts in Algorithms 1 and 2. Both use the same AUGUstus-mode attention (action before CoT): context

is causal, actions are bidirectional, and causal CoT tokens can read context and actions but not vice versa. Training uses noisy ground-truth actions and teacher-forced prediction CoT; deployment uses initial Gaussian noise and deterministic action denoising, with no ground-truth action or CoT required. Action normalization, time conditioning, and auxiliary module losses follow Appendix A and Appendix D.2.

D.4 Complete two-stage unified GRPO

Algorithm 7 specifies the complete two-stage post-training procedure. The reference is frozen at SFT throughout both stages; the behavior policy is snapshotted before each on-policy rollout batch. Within each context group, initial flow noise is shared, whereas transition exploration noise and CoT sampling distinguish the rollouts. The standard run uses the noise settings in Appendix B.5, the reward and format gates in Appendix D.1, and the optimization settings in Appendix D.2. The same prediction reward is used in both stages. Stage I uses $R_{\text{pred}} + 5\,\text{PDMS}$, whereas Stage II uses $R_{\text{pred}} + 5(R_{\text{endpoint}} + R_{\text{safe}})$ with the planning safety gate.

Algorithm 7 AUGUstus-GRPO: complete two-stage unified optimization

Require: Cold-start policy π_{SFT}; contexts $\mathcal{D}$; group size G; denoising schedule; clip and KL weights.
Ensure: Two-stage action–CoT policy π^*_{II}.

Freeze $\pi_{\text{ref}} \leftarrow \pi_{\text{SFT}}$ and initialize $\pi_\theta \leftarrow \pi_{\text{SFT}}$.
for stage $s \in \{1, 2\}$ **do**
  if $s = 2$ **then**
    Warm-start $\pi_\theta \leftarrow \pi^*_{\text{I}}$, the best validation-PDMS Stage-I checkpoint.
  end if
  for each on-policy RL iteration **do**
    Snapshot the behavior policy $\pi_{\text{old}} \leftarrow \pi_\theta$ and sample a context minibatch.
    for each context c in the minibatch **do**
      Draw one initial $x_1 \sim \mathcal{N}(0, I)$ shared by the G rollouts.
      for $i = 1$ to G **do**
        Initialize $x_{i,t_0} \leftarrow x_1$.
        for $n = 0$ to $N - 1$ **do**
          Predict $\hat{x}_0^{\text{old}} = f_{\text{old}}(x_{i,t_n}, t_n, c)$.
          Draw kernel noise z and sample $x_{i,t_{n+1}} \leftarrow c_x x_{i,t_n} + c_0 \hat{x}_0^{\text{old}} + \sigma_s z$ via Eq. (8).
          Record both states, $\hat{x}_0^{\text{old}}$, z, and the kernel coefficients for re-scoring.
        end for
        Set $x_i \leftarrow x_{i,0}$; decode action-conditioned CoT y_i and record its token log probabilities.
        Compute $R_{\text{pred},i}$ with the same prediction scoring and format gate in both stages.
        if $s = 1$ **then**
          Set $R_{\text{plan},i} \leftarrow \text{PDMS}_i$.
        else
          Set $R_{\text{plan},i} \leftarrow R_{\text{ep},i} + R_{\text{safe},i}$ and apply the planning safety gate.
        end if
        Set $R_i \leftarrow R_{\text{pred},i} + 5R_{\text{plan},i}$.
      end for
      Normalize group rewards into the shared advantages A_i using Eq. (4).
    end for
    Re-evaluate recorded action states and CoT prefixes under π_θ and the frozen π_{ref}.
    Evaluate JaT-GRPO ratios directly from $\hat{x}_0^\theta - \hat{x}_0^{\text{old}}$ using Eq. (14).
    Evaluate the action and CoT ratios and their native KL terms; keep each rollout's shared A_i.
    Compute the two clipped objectives $\mathcal{J}_a$ and $\mathcal{J}_c$.
    Jointly update θ with $\mathcal{L} = -\mathcal{J}_a + \beta_a \text{KL}_{\text{act}} + \alpha_c(-\mathcal{J}_c + \beta_c \text{KL}_{\text{CoT}})$.
    Synchronize the updated policy for the next rollout batch and retain the best validation checkpoint.
  end for
  Save the selected stage checkpoint as π^*_{I} or π^*_{II}.
end for
return π^*_{II}.

D.5 THREE-STAGE TRAINING TRACE

The SFT checkpoint provides scene grounding and a stable action prior. Stage I uses PDMS to improve progress and establishes the first RL checkpoint. Stage II switches to endpoint and safety shaping to recover within-group variance after PDMS saturates. The qualitative traces in Fig. 13 show the same progression: the predicted path becomes more compliant at Stage I and then adds a larger safety margin at Stage II. Meanwhile, the CoT prediction accuracy improves from 0.75 at SFT to 0.85 after RL, indicating increasingly accurate identification of critical objects and thereby enhancing the model's action planning capability.

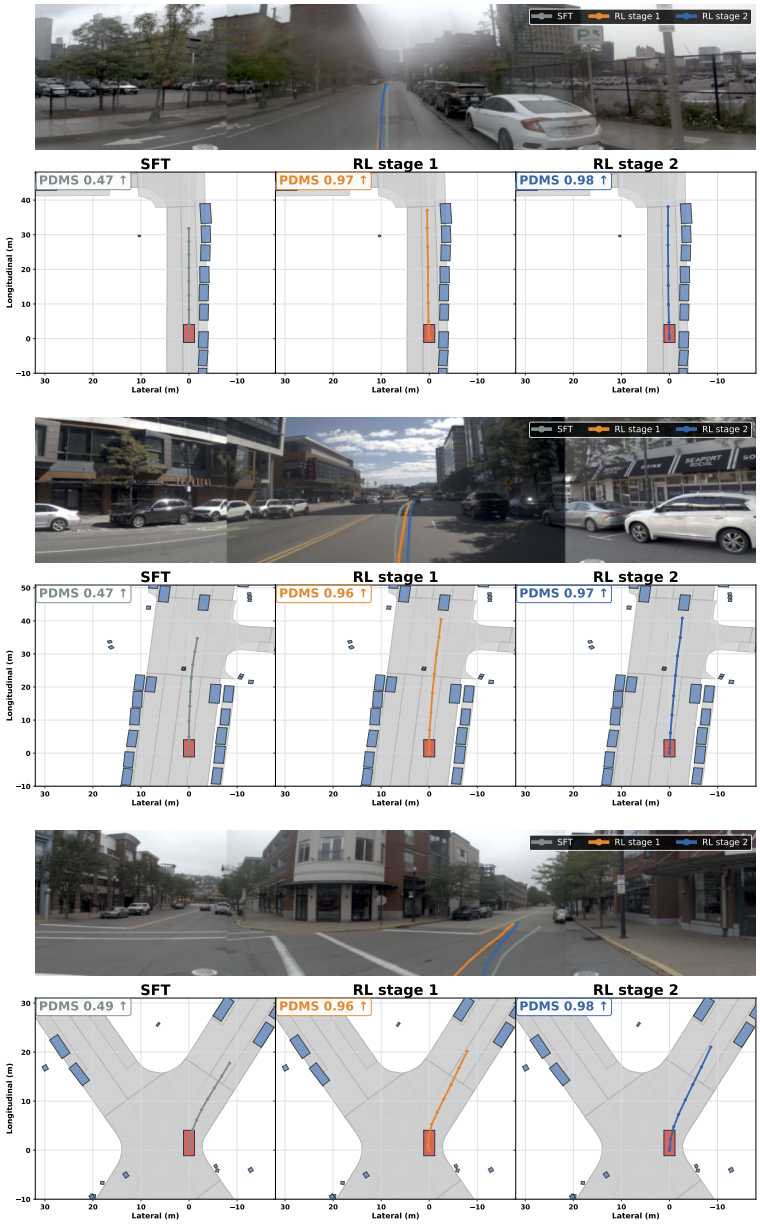

Figure 13: **Three-stage RL visualization.** Each case shows the same scene under SFT, RL Stage I, and RL Stage II. Stage I moves the path toward the desired progress profile; Stage II retains that progress while increasing the safety margin. The displayed case-level PDMS values are illustrative qualitative traces, whereas the aggregate values are in Table 7.

Table 10: **Measured batch-one latency on one H800.**

Path or module	Mean (ms)	Fraction
Vision encoder and M-RoPE	40.20	54.6%
Context prefix, eager	46.04	34.8%
One-step denoising, eager	45.70	34.6%
LLM stage, CUDA-graph replay	33.19	45.1%
End-to-end, eager	131.88	–
End-to-end, CUDA graph	**73.67**	**13.6 Hz**

Table 11: **Multi-seed evaluation of AUGUstus on NAVSIMv1.** Values are reported as mean ± standard deviation over three seeds. Training: three training seeds with the test seed fixed to 1. Testing: three test seeds with the training seed fixed to 1. SFT: supervised fine-tuning with AUGUstus-mode. GRPO: AUGUstus-GRPO Stage II.

Evaluation	Method	NC ↑	DAC ↑	EP ↑	TTC ↑	Comf ↑	PDMS ↑
Training seeds	SFT	99.02 ± 0.02	97.93 ± 0.03	84.18 ± 0.08	96.42 ± 0.03	99.98 ± 0.01	90.58 ± 0.05
	GRPO	98.92 ± 0.03	98.93 ± 0.02	91.92 ± 0.06	97.12 ± 0.02	99.99 ± 0.01	94.62 ± 0.03
Test seeds	SFT	99.01 ± 0.01	97.91 ± 0.02	84.22 ± 0.07	96.41 ± 0.02	99.99 ± 0.01	90.61 ± 0.04
	GRPO	98.91 ± 0.02	98.92 ± 0.01	91.88 ± 0.05	97.11 ± 0.01	99.98 ± 0.01	94.59 ± 0.02

D.6 Deployment Complexity

Latency. The deployed path is a causal visual prefix followed by one bidirectional eight-token action suffix. The vision encoder remains eager because the multimodal preprocessing contains host synchronization; the prefix and denoising suffix are captured in one CUDA graph. Table 10 reports the steady-state batch-one measurements from the profiling script in `fast_inference.md`.

The eager profile launches about 3.7k small kernels and performs roughly 200 pageable host-to-device copies per sample. Graph replay removes this CPU dispatch overhead and reduces the LLM stage from 91.83 to 33.19 ms (2.77×), while leaving the visual stage unchanged. The action suffix contributes only eight tokens on top of a roughly 2,257-token context. Optional post-hoc CoT is therefore deliberately disabled in the speed-critical configuration: a measured 166-token explanation takes about 4.5 s at batch one, whereas the action-only path is below 100 ms.

Scorer Module. Let C_{vis} be the visual encoder cost, $C_{\mathrm{pre}}(L)$ the prefix cost for L context tokens, and $C_{\mathrm{act}}(T)$ the eight-token action suffix. Our single-trajectory path is

$$T_{\mathrm{ours}} = C_{\mathrm{vis}} + C_{\mathrm{pre}}(L) + C_{\mathrm{act}}(T), \qquad T = 8, \tag{22}$$

and is independent of the number of candidates. A VLA+scorer system that generates N trajectories and scores them costs approximately

$$T_{\mathrm{scorer}}(N) = N\big(C_{\mathrm{pre}}(L) + C_{\mathrm{act}}(T)\big) + C_{\mathrm{vis}} + NC_{\mathrm{score}}. \tag{23}$$

Thus, N=32–64 multiplies the expensive VLM pass even if the scorer itself is lightweight. As a result, this VLA+scorer design is difficult to run in real time on vehicle hardware. We therefore generate one trajectory in one network evaluation; the deployed path takes 73.6 ms on an H800 (13.6 Hz) and 83 ms on the real vehicle (12 Hz), while retaining SOTA public-benchmark performance.

E Multi-Seed Evaluation on Training and Test Seeds

We evaluate the stability of AUGUstus with respect to both training seeds and test seeds. For the training-seed study, we train the model with three independent random seeds while fixing the test seed to 1. For the test-seed study, we train the model with seed 1 and evaluate it with three different test seeds. All other settings are kept identical. Table 11 reports the mean performance of SFT and the final GRPO stage (Stage II).

F Qualitative Results on Public, Private Data, and Driving Test

F.1 Public benchmark examples

The public examples compare the same scene across ReCogDrive (Xiong et al., 2026), AutoDrive-P^3 Ye et al. (2026b), and AUGUstus. The six cases in Fig. 14 provides representative qualitative comparisons.

F.2 Internal 1,000 km data

The internal set contains long-tail interactions and camera conditions not present in NAVSIM. Figures 15 shows six representative cases from the 1,000 km evaluation; the category-level aggregate remains the one reported in Table 4.

F.3 Regression and flow-matching baselines

Figure 16 keeps the direct VLM+regression and VLM+flow-matching comparison in one image. Both baselines use the same scene input; the comparison isolates the action parameterization and the shared CoT training rather than a sensor change.

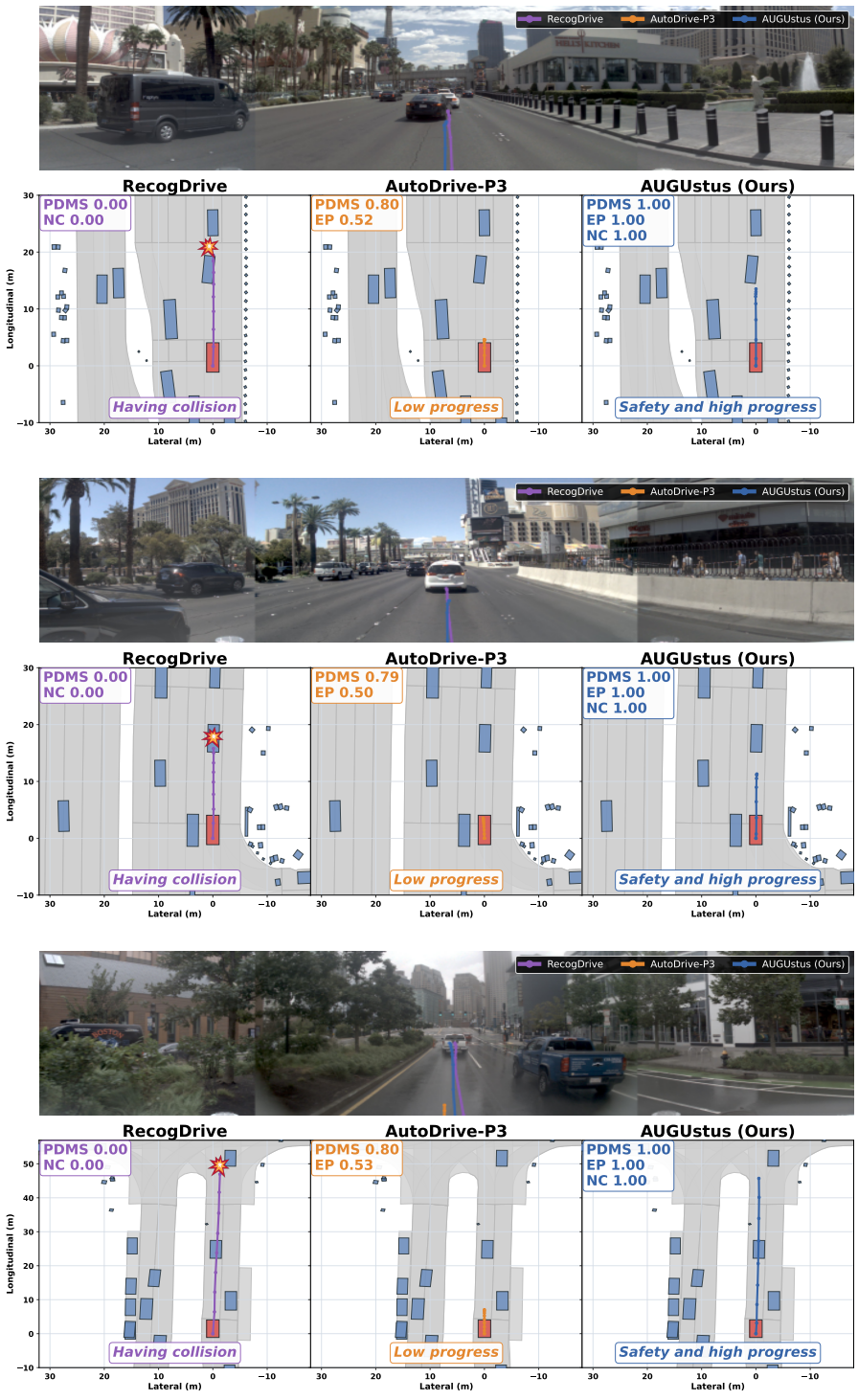

Figure 14: **Public-set qualitative comparison.** Each panel contrasts ReCogDrive, AutoDrive-P^3, and AUGUstus on the same public scene. The single AUGUstus trajectory is both collision-free and route compliant in these examples.

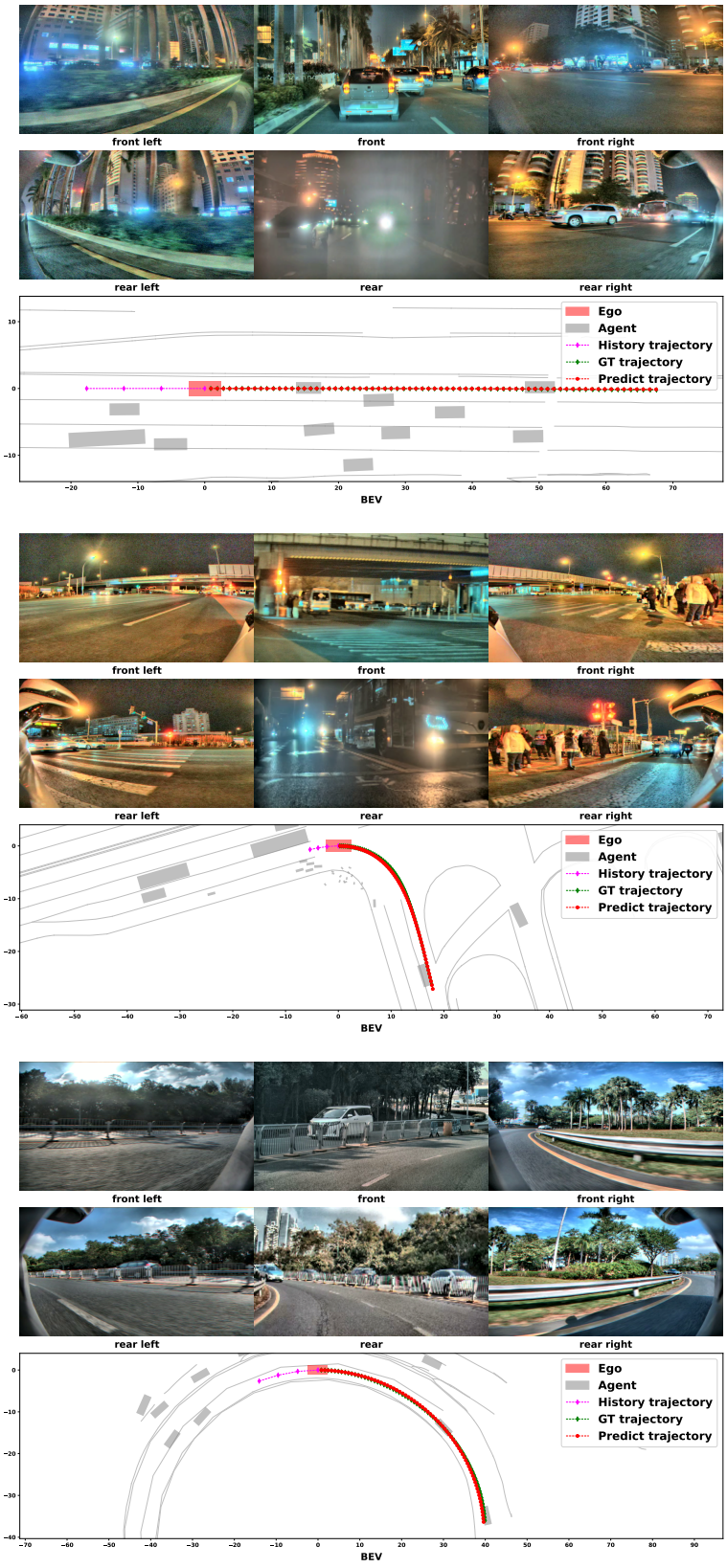

Figure 15: **Internal 1,000 km qualitative examples.** Representative long-tail interactions and camera conditions from the private evaluation set.

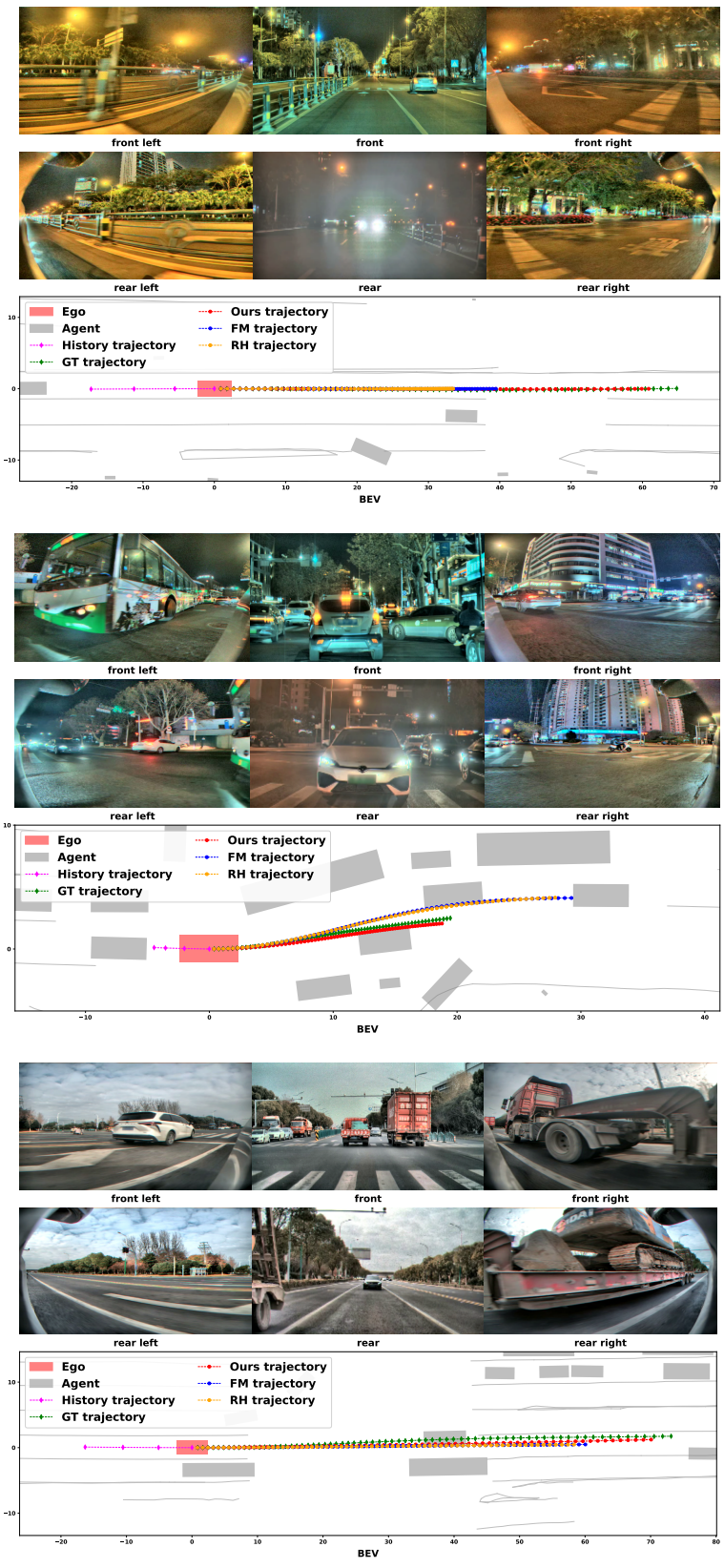

Figure 16: **Internal qualitative comparison of VLM+regression, VLM+flow matching, and AUGUstus.**

F.4 REAL-VEHICLE VISUALIZATION

The real-vehicle examples use the same action-only deployment path as the latency measurement. The camera mosaic and bird's-eye trajectory in Fig. 17 show that the one-step output remains smooth in nighttime and multi-camera conditions.

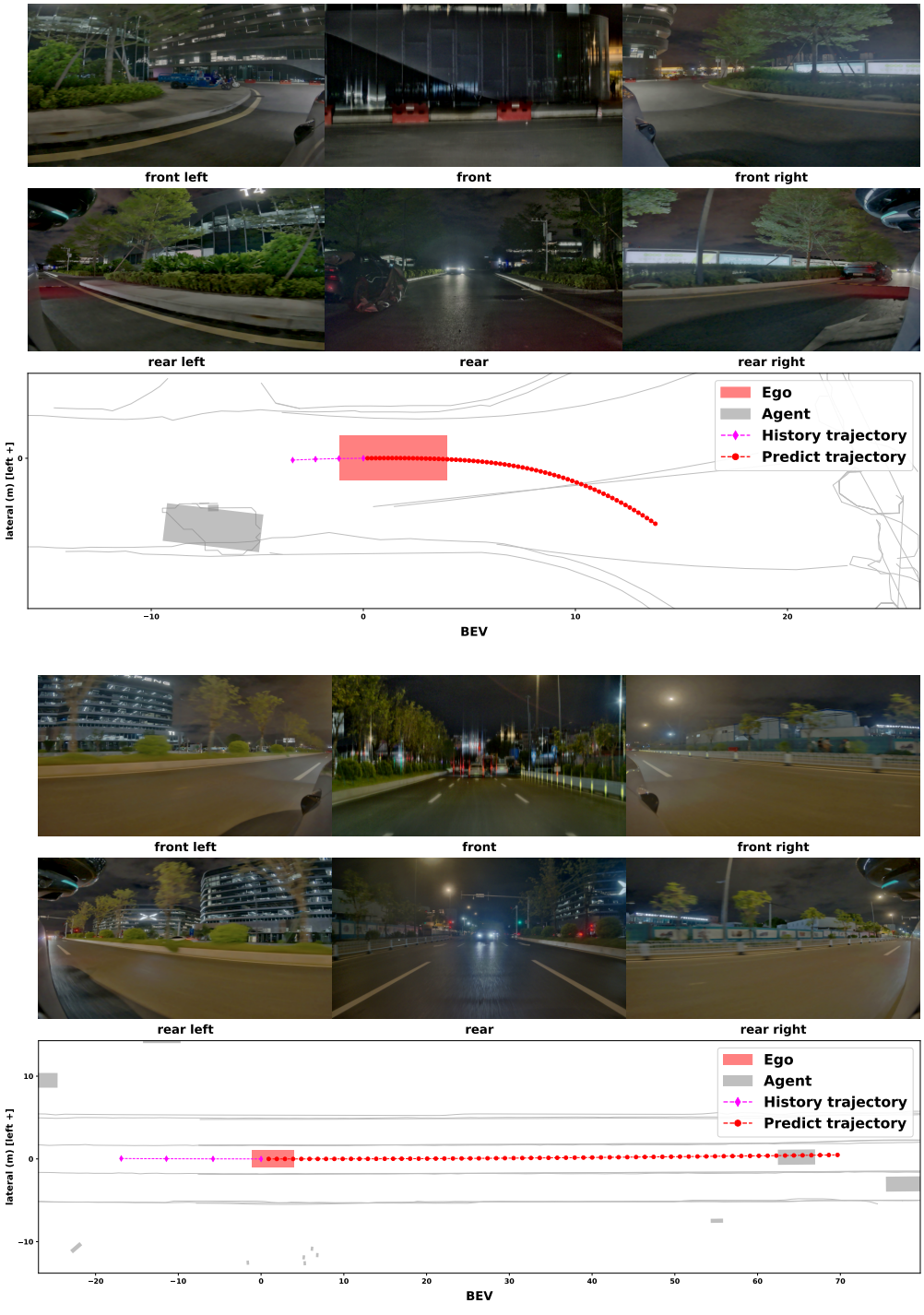

Figure 17: **Real-vehicle qualitative examples.** The deployed action-only path runs at 12 Hz on the vehicle. The BEV trajectory is generated directly by the shared model; CoT decoding is omitted from this loop.