AutoVLA-D: Learn a Trajectory-Geometric Condition from Reasoning for Autonomous Driving VLA

Yuguang Yang[1,3*] **Zhewen Tan**[8 *] **Canyu Chen**[2,3 *]
Linlin Yang[5] Cheng Chi[7] Chunyang Liu[4] Kehua Sheng[4] Jinyu Yang [9]
Baochang Zhang[6†] Bo Zhang[4 ‡] Yan Wang [3 ‡] Xianbin Cao [1 ‡]
[1]School of Electronic and Information Engineering, Beihang University
[2]National College for Excellent Engineers, Beihang University
[3]Institute for AI Industry Research (AIR), Tsinghua University
[4]Didi
[5]State Key Laboratory of Media Convergence and Communication, Communication University of China
[6] Hangzhou Innovation Research Institute, Beihang University
[7]School of Information, Renmin University of China
[8]School of Computer Science and Engineering, Beihang University
[9]Harbin Institute of Technology, Shenzhen

Abstract

Autoregressive VLA planning usually predicts continuous trajectories as the final output, but driving instructions often describe actions rather than ego states, which weakens direct conditioning on the final continuous trajectory. We address this gap with *TrajCond*, which introduces a discrete trajectory-geometric condition for VLA planning. Specifically, we represent TrajCond as a 1D-Brohan differential action trajectory and learn it with a two-stage training framework before final continuous trajectory prediction. With the simultaneous autoregressive generation of discrete and numeric trajectories, we name our method **AutoVLA-D** (Double). Experiments on NAVSIM navtest, nuScenes, and NeuroNCAP show that discrete trajectories are weaker as direct outputs but more effective as TrajCond, consistently improving final continuous trajectory prediction across closed-loop and open-loop driving benchmarks. Code is available at `https://github.com/ShivaTeam/AutoVLA-D`.

1 Introduction

Autoregressive Vision-Language-Action (VLA) planning has become a practical Multimodal Large Language Model (MLLM)-native paradigm for autonomous driving. Given camera observations and driving instructions, a VLA planner follows the next-token interface of modern MLLMs: it conditions on visual-language inputs, produces intermediate driving reasoning, and then decodes the future ego trajectory [22]. Recent methods have demonstrated strong closed-loop planning capability with this input–reasoning–trajectory generation process [30; 2; 16; 17; 11].

Despite this progress, existing VLA planners can also be distinguished by the trajectory form used for the final ego plan. One line of work models the future ego states, including the relative x-axis position, y-axis position, and heading in the current ego frame, over the next few seconds as numeric text waypoint tokens [16; 17; 2], which we call *continuous trajectory* in this paper; while

*Equal contribution:{guangbuaa, chencanyu, tanzhewen}@buaa.edu.cn
†Project Lead
‡Corresponding author: `xbcao@buaa.edu.cn`, `zhangbo@didiglobal.com`, `wangyan@air.tsinghua.edu.cn`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

another line, *i.e.*, *discrete trajectory*, represents the step-wise ego motion, or state transition, as discrete action tokens [25; 11; 30]. These two trajectory forms have complementary strengths and limitations. Action tokens introduce reconstruction error when discretizing continuous trajectories, and our experiments over multiple discretization strategies, *e.g.*, K-Means, K-Disks [30; 25; 20], and Brohan-style discretization [31], show that directly regressing discrete trajectories is consistently weaker than directly regressing continuous trajectories. However, driving instructions often describe actions such as yielding, keeping, or turning, and such local motion changes are naturally aligned with discrete trajectory tokens because each action token describes a local state transition. This creates a formulation gap when instructions are mapped directly to the final continuous trajectory.

To address this, we propose **AutoVLA-D** (Double), a two-stage VLA planner that uses a discrete trajectory as a *Trajectory-geometric Condition* (TrajCond) before regressing the final continuous trajectory. This approach features three core innovations:

- **Discrete trajectory as trajectory-geometric condition.** TrajCond first predicts a discrete trajectory and then uses its geometric-semantic as a condition for final continuous prediction. This keeps the final output in the stronger continuous space while using the discrete condition to transmit action-oriented information.
- **Action tokens with 1D-Brohan discretization.** We construct TrajCond from differential ego motions and discretize the x, y, and yaw differentials with separate Brohan-style codebooks. Compared with shared-codebook clustering tokenizers, such as K-Means and K-Disks [30; 25; 20], this design gives the strongest tested TrajCond result and the lowest tested reconstruction error.
- **Two-stage alignment training.** After constructing Brohan TrajCond by discretizing the ground-truth trajectory, the newly constructed action vocabulary has no VLA-task-related initialization. Stage 1 therefore warms up TrajCond's semantics by aligning observations and instructions with the action-token sequence that encodes TrajCond. Stage 2 uses the decoded TrajCond and the available reasoning context to regress the final continuous trajectory. This separates condition learning from final trajectory prediction.

We evaluate AutoVLA-D on NAVSIM navtest, nuScenes, and NeuroNCAP. AutoVLA-D achieves state-of-the-art results across these settings, including 92.0 PDMS and 86.8 EPDMS on NAVSIM navtest. Beyond benchmark comparisons, our experiments are organized to verify how and why trajectory forms matter for driving VLAs. In TrajCond Ablation and Progressive Analysis, we examine the staged performance gains of AutoVLA-D and validate the contribution of its training and reasoning components. In TrajCond Form Analysis, we compare continuous trajectories with different action-token forms, showing that discrete trajectories are weaker as final autoregressive outputs but become more effective when used as TrajCond for final continuous prediction. In TrajCond Semantic Analysis, we qualitatively and quantitatively verify whether TrajCond absorbs action-oriented planning semantics in addition to providing geometric guidance. Together, these experiments suggest that in autoregressive VLAs, discrete trajectories used as TrajCond can encode instruction semantics while preserving trajectory geometry, thereby reducing the formulation gap between instructions and continuous trajectories and improving VLA planning performance.

2 Related Work

Autonomous Driving VLA. Autonomous driving planning first developed through end-to-end systems that map sensor observations to future trajectories, including ST-P3 [6], VAD [9], UniAD [7], DiffusionDrive [13], SparseDrive [24], and BridgeAD [28]. With the emergence of vision-language models (VLMs), recent VLA methods exploit visual-semantic knowledge for driving understanding, reasoning, and planning. They follow two patterns: planner-style methods, including ReCogDrive [12], ImagiDrive-S [10], DriveVLA-W0 [11], and FLARE [26], use the VLM as a feature extractor or semantic reasoner with an external planner or action head; autoregressive methods, including EMMA [8], OpenEMMA [27], OpenDriveVLA [29], Impromptu VLA [3], AutoVLA [30], AdaThinkDrive [16], Curious-VLA [2], and ELF-VLA [17], predict waypoint or action-token sequences through the next-token interface. Unlike autoregressive VLAs that directly decode the final trajectory after reasoning, AutoVLA-D inserts a learned trajectory-geometric condition before final continuous trajectory generation.

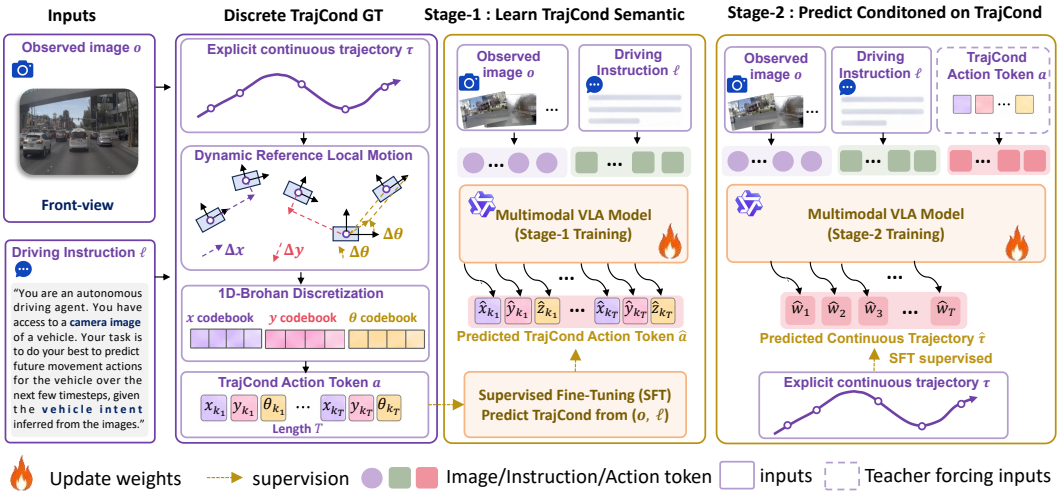


Figure 1: Pipeline of AutoVLA-D.

Trajectory Form in Autoregressive VLA. Autoregressive VLA planners represent the future ego trajectory over three degrees of freedom: x-axis position, y-axis position, and heading. Existing trajectory forms can be grouped into two types. Continuous-trajectory methods, including Impromptu VLA [3], AdaThinkDrive [16], Curious-VLA [2], and ELF-VLA [17], serialize each future ego state as numeric text waypoint tokens; among them, Curious-VLA [2] further improves performance with step-wise trajectory normalization. Discrete-trajectory methods instead serialize each local ego-state transition as an action token and recover the future trajectory by temporal integration. AutoVLA [30] builds K-Means/K-Disks-style action codebooks whose tokens represent 3-DoF local motions [25; 20], while DriveVLA-W0 [11] uses frequency-based FAST tokenization [19]. AutoVLA-D differs by using discrete trajectories as trajectory-geometric conditions rather than final outputs, and adopts a 1D-Brohan form where each action token represents a 1-DoF local motion.

3 Method

3.1 Preliminary & Formulation

Task formulation. Given observed images $\mathbf{o}$ and a linguistic user instruction ℓ, a conventional VLA planner directly predicts the final future trajectory:

$$\tau = f_{\text{VLA}}(\mathbf{o}, \ell), \tag{1}$$

In this paper, the final prediction target τ is always a continuous trajectory, represented for autoregressive decoding as the normalized text-waypoint sequence $\{\tilde{\tau}_t\}_{t=1}^{T}$. This direct formulation therefore asks the model to map language instructions to the final continuous trajectory autoregressively.

As shown in Fig. 1, **AutoVLA-D**, introduces an explicit intermediate *trajectory-geometric condition*. Let $\mathbf{a} = \{a_t\}_{t=1}^{T}$ denote the action-token sequence that encodes this condition. Unlike the final output trajectory, TrajCond is discrete. Instead of mapping ℓ directly to the final continuous trajectory τ, we train the same VLA model through two supervised stages:

$$\mathbf{a} = f_{\text{VLA}}^{(1)}(\mathbf{o}, \ell), \qquad \tau = f_{\text{VLA}}^{(2)}(\mathbf{o}, \ell, \mathbf{a}), \quad f_{\text{VLA}}^{(2)} \leftarrow f_{\text{VLA}}^{(1)}. \tag{2}$$

where superscripts indicate the training stage rather than separate architectures, and $f_{\text{VLA}}^{(2)} \leftarrow f_{\text{VLA}}^{(1)}$ means that the Stage-2 model is initialized from the Stage-1 model and then further trained. Both stages use the MLLM's autoregressive token decoder: Stage 1 predicts TrajCond action tokens from $(\mathbf{o}, \ell)$, and Stage 2 predicts final continuous waypoint tokens from $(\mathbf{o}, \ell, \mathbf{a})$.

Continuous trajectory preliminary. Given observed images $\mathbf{o}$ and a linguistic user instruction ℓ, the model predicts a future ego trajectory $\tau = \{(x_t, y_t, \theta_t)\}_{t=1}^{T}$. Following Curious-VLA [2], the final continuous trajectory is serialized as normalized text waypoints. Concretely, letting $\tau_t = (x_t, y_t, \theta_t)$ denote the trajectory state at step t, we use the step-wise normalized representation

$$\tilde{\tau}_t = \frac{\tau_t - \mu_t}{\sigma_t}, \quad \hat{\tau}_t = \hat{\tilde{\tau}}_t \sigma_t + \mu_t, \tag{3}$$

where (μ_t, σ_t) are per-step statistics estimated from the training set. We use *TrajCond* to bridge action-oriented instructions and continuous ego-state outputs.

Context preliminary. For all datasets, ℓ denotes the linguistic user instruction. For NAVSIM, we additionally use Qwen3-VL-235B to annotate Chain-of-Thought (CoT) reasoning from the front-view image, high-level intent, past ego trajectory, and expert future trajectory. Following prior VLA reasoning practice [22], the annotation contains three fields: critical objects and conditions, a natural-language explanation of the expert trajectory, and speed/command meta-behavior labels. The CoT sequence can be viewed as an extension term associated with ℓ in Eq. 2, but it is not provided as input; instead, the model is trained to regress it as additional language supervision. The full annotation prompt and the dataset-specific Stage-1/Stage-2 user prompts are provided in Supplement Sec. C. For nuScenes and NeuroNCAP, no CoT annotation is used. For fair comparison, $\mathbf{o}$ uses a single front-view image on NAVSIM and nuScenes, and left-front-right three view images on NeuroNCAP.

3.2 Trajectory-Geometric Condition via Action Tokens

A useful TrajCond should be predictable with discrete tokens while retaining enough geometry to guide continuous planning. We therefore build TrajCond from trajectory differentials, which preserve local motion changes while admitting a compact discrete vocabulary.

Action tokenizer. The trajectory $\tau = \{(x_t, y_t, \theta_t)\}_{t=1}^{T}$ is expressed in the ego frame at $t = 0$. Unlike previous works that rely on a static coordinate frame [30; 11], to describe the motion from step $t-1$ to t, we use the local frame at step $t-1$ and compute the differential motion with a standard planar rotation transform [18]. Supplement Fig. 6 illustrates this local-frame differential construction:

$$\begin{aligned} \delta_t &= (\Delta x_t, \Delta y_t, \Delta\theta_t), \quad \Delta\theta_t = \theta_t - \theta_{t-1}, \\ \begin{bmatrix} \Delta x_t \\ \Delta y_t \end{bmatrix} &= \mathbf{R}(-\theta_{t-1}) \begin{bmatrix} x_t - x_{t-1} \\ y_t - y_{t-1} \end{bmatrix}, \quad \mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \end{aligned} \tag{4}$$

Let $\Delta\tau = \{\delta_t\}_{t=1}^{T}$ denote the differential trajectory. Following previous action-token practice [30; 25; 20], we extract differential actions from training trajectories and build an action-template set. Existing action-template extraction methods commonly use K-Means or K-Disks to cluster the joint 3-DoF motion vector $(\Delta x, \Delta y, \Delta\theta)$ into one shared codebook of size Q. In our analysis, this is the *3D shared-codebook* form because each time step is represented by one action token that encodes a 3-DoF local motion.

In contrast, we find in Sec. 4.4 that clustering-based K-Means/K-Disks tokenizers produce a clear train-test reconstruction-error gap. Our final setting therefore follows the linear per-dimension discretization strategy used in RT-2 [31]. For each action dimension $i \in \{1, 2, 3\}$ corresponding to x, y, and θ, we compute the 1st and 99th quantiles on the training set and uniformly divide that interval into Q_i bins. The total action-token codebook size is $Q = \sum_{i=1}^{3} Q_i$. This yields three codebooks $\mathbf{x}$, $\mathbf{y}$, and $\boldsymbol{\theta}$, one per motion dimension, namely the *1D-Brohan* form where each individual action token represents a 1-DoF local motion and the three tokens at a time step jointly describe a 3-DoF local transition.

Trajectory discretization. Following [20; 30; 25], we iteratively find the token with minimum corner distance to the next state. In our final 1D-Brohan setting, a candidate tuple $\mathbf{k} = (k_1, k_2, k_3)$ selects one bin center from each action codebook:

$$\bar{\delta}(\mathbf{k}) = (x_{k_1}, y_{k_2}, \theta_{k_3}), \quad k_i \in \{1, \ldots, Q_i\}. \tag{5}$$

Starting from the current ego pose $\hat{\tau}_0 = (0, 0, 0)$, we render each candidate from the previously tokenized pose $\hat{\tau}_{t-1}$:

$$\begin{bmatrix} \bar{x}_t(\mathbf{k}) \\ \bar{y}_t(\mathbf{k}) \end{bmatrix} = \begin{bmatrix} \hat{x}_{t-1} \\ \hat{y}_{t-1} \end{bmatrix} + \mathbf{R}(\hat{\theta}_{t-1}) \begin{bmatrix} x_{k_1} \\ y_{k_2} \end{bmatrix}, \quad \bar{\theta}_t(\mathbf{k}) = \hat{\theta}_{t-1} + \theta_{k_3}. \tag{6}$$

The best candidate tuple for step t is selected by comparing the rendered candidate state with the ground-truth next state:

$$\mathbf{k}_t^* = \arg\min_{\mathbf{k}} d_{\text{corner}}\big((\bar{x}_t(\mathbf{k}), \bar{y}_t(\mathbf{k}), \bar{\theta}_t(\mathbf{k})), (x_t, y_t, \theta_t)\big), \tag{7}$$

where d_{corner} is the mean distance between the ordered box corners. The action-token tuple is $a_t = \mathbf{k}_t^*$. Iterating this process yields the full discrete action sequence $\mathbf{a} = \{a_t\}_{t=1}^{T}$. Since the three

1D codebooks correspond to different DoFs, exhaustive tuple search would require Q_i^3 candidates at each step. We therefore first take the four tokens with the smallest absolute per-dimension residuals for each DoF, enumerate the resulting $4 \times 4 \times 4$ candidate tuples, and choose the tuple with the smallest corner error. This design converts a continuous trajectory into a sequence of local action components; Supplement Sec. D provides the visualization and pseudocode for the full process.

3.3 AutoVLA-D Training & Inference

Here we detail the training and inference process described in the task formulation of Sec. 3.1.

Stage-1 Inference & Training. The newly constructed action vocabulary has no VLA-task-related initialization, so Stage 1 warms up TrajCond's semantics by taking multimodal vision and language context $(\mathbf{o}, \ell)$ as input and outputting the action-token sequence $\mathbf{a}$. We train this stage with Supervised Fine-Tuning (SFT) using the autoregressive token loss:

$$\mathcal{L}_{\text{stage1}} = -\sum_{t=1}^{T} \log P(a_t \mid \mathbf{o}, \ell, a_{<t}). \tag{8}$$

Stage 1 is not used to produce the final trajectory directly; it learns TrajCond as the trajectory-geometric condition that guides Stage 2.

Stage-2 Inference & Training. This stage model learns to use TrajCond $\mathbf{a}$ together with visual and linguistic instruction context $(\mathbf{o}, \ell)$ to regress the target continuous trajectory. Let $\mathbf{w} = \{w_t\}_{t=1}^{T}$ denote the token sequence corresponding to the target continuous trajectory. Stage 2 takes the same multimodal context $(\mathbf{o}, \ell)$ as input and follows the output order $\mathbf{a} \rightarrow \mathbf{w}$. We train this stage with SFT, and the objective combines the action-token loss and the final waypoint-token loss:

$$\mathcal{L}_{\text{stage2}} = -\sum_{t=1}^{T} \log P(a_t \mid \mathbf{o}, \ell, a_{<t}) - \sum_{t=1}^{T} \log P(w_t \mid \mathbf{o}, \ell, \mathbf{a}, w_{<t}). \tag{9}$$

Stage 2 optimizes the full ordered output sequence that contains both TrajCond action tokens and final trajectory tokens. As shown in the dashed box in Fig. 1, we adopt the teacher-forcing strategy, where action tokens are supervised by ground truth trajectory during training, while inference is performed in a fully autoregressive manner that TrajCond firstly predicts TrajCond action trajectory, then predicts the final continuous trajectory.

Reasoning & RFT. For NAVSIM, we additionally use CoT reasoning, as described in the context preliminary of Sec. 3.1. Let $\mathbf{r}$ denote the CoT reasoning tokens; the output order becomes $\mathbf{r} \rightarrow \mathbf{a}$ in Stage 1 and $\mathbf{a} \rightarrow \mathbf{r} \rightarrow \mathbf{w}$ in Stage 2. Following Curious-VLA [2], we add a third training stage with the output format $\mathbf{a} \rightarrow \mathbf{r} \rightarrow \mathbf{w}$, and use GRPO [23] to perform Reinforcement Fine-Tuning (RFT). For fair comparison, nuScenes and NeuroNCAP use only Stage-1/2 SFT training without CoT reasoning.

As supported by the trajectory-representation analysis in Fig. 3, directly predicting discrete trajectories is weaker than predicting the final continuous trajectory, but using the discrete trajectory as TrajCond strengthens the final planner. The two-stage design therefore preserves the strength of the continuous output space while exploiting the alignment advantage of TrajCond.

4 Experiments

In Sec. 4.2, we report performance on NAVSIM navtest, nuScenes, and NeuroNCAP. In Sec. 4.3, we conduct TrajCond ablation and progressive analysis. In Sec. 4.4, we compare different TrajCond forms and validate the compatibility of TrajCond with different discrete trajectory tokenizers. In Sec. 4.5, we conduct semantic analyses of TrajCond.

4.1 Experimental Setup

Datasets and benchmark settings. We report results on **NAVSIM navtest** [4; 2], a large-scale non-reactive closed-loop simulation benchmark, **nuScenes** [1], an open-loop real-world driving benchmark, and **NeuroNCAP** [14], a photorealistic closed-loop simulator built on nuScenes. For

Table 1: NAVSIM navtest PDMS results under the public metric setting. † denotes results reproduced by us. * denotes concurrent work.

Method	Base Model	NC↑	DAC↑	EP↑	TTC↑	C↑	PDMS↑
ImagiDrive-S	InternVL2.5-4B	98.6	96.2	80.5	94.5	**100.0**	87.4
ReCogDrive	InternVL3-8B	98.2	97.8	83.5	95.2	**100.0**	89.6
ReCogDrive†	Qwen3-VL-4B	98.2	97.7	86.0	97.7	**100.0**	90.7
DriveVLA-W0	Emu3-8B	98.7	**99.1**	83.3	95.3	99.3	90.2
AutoVLA	Qwen2.5-VL-3B	98.4	95.6	81.9	**98.0**	99.9	89.1
AdaThinkDrive	InternVL3-8B	98.4	97.8	84.4	95.2	**100.0**	90.3
Curious-VLA	Qwen2.5-VL-3B	98.4	96.9	88.5	97.9	98.1	90.3
Curious-VLA†	Qwen3-VL-4B	98.5	96.7	87.3	**98.0**	98.3	89.7
FLARE*	Qwen3-VL-4B	98.5	98.4	86.0	96.0	**100.0**	91.4
ours	Qwen3-VL-4B	**98.8**	97.7	**89.8**	97.7	99.8	**92.0**

Table 2: NAVSIM navtest EPDMS results following the official EPDMS protocol. AutoVLA-D uses the same 1-camera setting as the PDMS experiment and reports the completed EPDMS evaluation. † denotes results reproduced by us. * denotes concurrent work. EPDMS denotes the bug-fixed EPDMS metric released by the official repository of NAVSIM v2.

Method	Base Model	NC↑	DAC↑	DDC↑	TLC↑	EP↑	TTC↑	LK↑	C↑	EC↑	EPDMS↑
ReCogDrive	InternVL3-8B	98.3	95.2	**99.5**	99.8	87.1	97.5	96.6	98.3	86.5	83.6
DiffusionDrive	–	98.2	95.9	99.4	99.8	87.5	97.3	96.8	98.3	**87.7**	84.5
Curious-VLA	Qwen2.5-VL-3B	98.4	96.9	99.2	99.8	88.5	97.9	**96.9**	98.1	81.5	85.3
Curious-VLA†	Qwen3-VL-4B	98.5	96.7	99.37	**99.9**	87.3	98.0	96.5	98.3	83.0	84.9
DriveVLA-W0	Emu3-8B	98.5	**99.1**	98.0	99.7	86.4	98.1	93.2	97.9	58.9	86.1
FLARE*	Qwen3-VL-4B	**98.8**	96.6	**99.5**	**99.9**	87.9	**98.2**	95.8	**98.4**	87.5	86.3
ours	Qwen3-VL-4B	**98.8**	97.7	98.8	99.8	**91.2**	**98.2**	**96.9**	98.2	80.2	**86.8**

fair comparison, we follow ReCogDrive [12] to report NAVSIM navtest, follow Impromptu VLA [3] to report nuScenes, and follow ImagiDrive [10] to report NeuroNCAP. More benchmark details are provided in Supplement Sec. B.

Training setup. For NAVSIM and the representation ablations, we use Qwen3-VL-4B unless otherwise specified. For nuScenes and NeuroNCAP, we report both Qwen2.5-VL and Qwen3-VL variants. Training consists of SFT followed by optional RFT. We implement SFT with LLaMA-Factory and RFT with EasyR1. We optimize with AdamW [15], weight decay 0.05, warmup ratio 0.10, cosine learning-rate scheduler, and global batch size 32. Stage 1 and Stage 2 each run for 3 epochs with learning rate 4×10^{-5}. For RFT, we perform GRPO training for 160 steps with an actor batch size of 512, broadly consistent with Curious-VLA (Chen et al., 2026). The model-context setup for NeuroNCAP, NAVSIM, and nuScenes is provided in context preliminary of Sec. 3.1.

Trajectory formulation. Unless otherwise specified, *continuous trajectory* denotes the default final trajectory form expressed as normalized text waypoints, *TrajCond* denotes discrete trajectory encoded by 1D-Brohan Q=768 codebooks, where each DoF's codebook has Q_i=256 bins. We choose Q=768 after comparing different codebook-size settings in Sec. 4.4.

4.2 Benchmark Comparisons

NAVSIM navtest PDMS results. Table 1 reports the NAVSIM navtest comparison. The compared methods include VLA-planner methods ImagiDrive-S (Li et al., 2025), ReCogDrive (Li et al., 2025), and DriveVLA-W0 (Li et al., 2025); and VLA-token methods AutoVLA (Zhou et al., 2025), AdaThinkDrive (Luo et al., 2025), Curious-VLA (Chen et al., 2026), and FLARE (Xie et al., 2026). Additional baselines are provided in Supplement Table 7. Our NAVSIM setting uses front-view input, reason data, and optional GRPO [23]. Experiment results demonstrated that AutoVLA-D reaches 92.0 PDMS and exceeds the reported scores of prior VLA planners.

NAVSIM navtest EPDMS results. Table 2 reports the same benchmark under the stricter EPDMS metric set. The compared methods include ReCogDrive (Li et al., 2025), DiffusionDrive (Liao et al., 2025), DriveVLA-W0 (Li et al., 2025), Curious-VLA (Chen et al., 2026), and FLARE (Xie et al.,

Table 3: Open-loop comparison on nuScenes. We report L2 error and collision rate (Col.). Gray rows denote own ablation w/o TrajCond (TC).

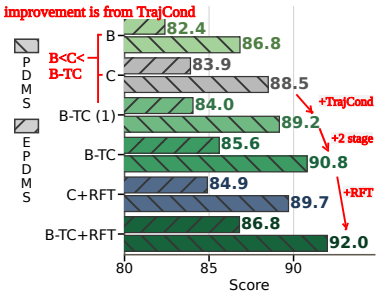
Method	Base Model	ST-P3 metrics		UniAD metrics	
		L2 ↓	Col. ↓	L2 ↓	Col. ↓
ST-P3	–	2.11	0.71	–	–
VAD	–	0.37	0.14	–	–
UniAD	–	0.69	0.12	1.03	0.31
EMMA	Gemini	0.32	–	–	–
OpenEMMA	Qwen2-VL-7B	2.81	–	–	–
OpenDriveVLA	Qwen2.5-VL-3B	0.33	0.10	0.67	0.30
AutoVLA	Qwen2.5-VL-3B	0.48	0.13	0.86	0.35
Impromptu VLA	Qwen2.5-VL-3B	0.33	0.13	0.67	0.38
Ours	Qwen2.5-VL-3B	0.30	0.10	0.60	0.35
Ours w/o TC	Qwen3-VL-4B	0.30	0.11	0.60	0.34
Ours	Qwen3-VL-4B	**0.30**	**0.09**	**0.60**	**0.29**

Table 4: Closed-loop comparison on NeuroNCAP. Gray rows denote own ablation w/o TrajCond (TC).

Method	Base Model	NeuroNCAP Score ↑				Collision rate ↓			
		Avg.	Stat.	Frontal	Side	Avg.	Stat.	Frontal	Side
UniAD	–	0.73	0.84	0.10	1.26	0.89	0.88	0.98	0.80
VAD	–	0.66	0.47	0.04	1.45	0.93	0.96	1.00	0.82
SparseDrive	–	0.92	–	–	–	0.94	–	–	–
BridgeAD-S	–	1.52	–	–	–	0.76	–	–	–
BridgeAD-B	–	1.60	–	–	–	0.73	–	–	–
ImagiDrive-S	InternVL2.5-4B	3.49	**4.15**	2.45	**3.88**	0.45	0.34	0.74	**0.27**
Impromptu VLA	Qwen2.5-VL-3B	2.15	1.77	2.31	2.10	0.66	0.70	0.59	0.65
Ours w/o TC	Qwen2.5-VL-3B	2.76	2.95	2.98	2.35	0.49	0.43	0.48	0.56
Ours	Qwen2.5-VL-3B	**3.63**	4.02	3.43	3.44	**0.36**	**0.26**	0.44	0.36
Ours w/o TC	Qwen3-VL-4B	1.66	1.81	1.06	1.66	0.68	0.69	0.82	0.68
Ours	Qwen3-VL-4B	3.23	3.17	**3.68**	2.86	0.39	0.41	**0.32**	0.44

Table 5: Progressive performance improvement on NAVSIM with Qwen3-VL-4B. We abbreviate Reason, Continuous, Brohan, and epochs as R, C, B, and ep., respectively. The right panel reports the corresponding EPDMS/PDMS scores.

Method	**Stage 1**	**Stage 2**	**RFT**
SFT-only Results w/ Reason (R)			
Continuous (C)	R → C (6 ep.)	–	–
Continuous TrajCond	R → C (3 ep.)	C → R → C (3 ep.)	–
Brohan (B)	R → B (6 ep.)	–	–
Brohan TrajCond (1)	–	B → R → C (6 ep.)	–
Brohan TrajCond	R → B (3 ep.)	B → R → C (3 ep.)	–
SFT + RFT (GRPO) Results w/ Reason (R)			
Continuous + RFT	R → C (6 ep.)	–	✓
Brohan TrajCond + RFT	R → B (3 ep.)	B → R → C (3 ep.)	✓



2026); additional E2E baselines are provided in Supplement Table 8. Under this setting, AutoVLA-D reaches 86.8 EPDMS and improves over the reported Curious-VLA (Chen et al., 2026) score of 85.3.

nuScenes open-loop comparison. Table 3 reports the nuScenes open-loop comparison against ST-P3 (Hu et al., 2022), VAD (Jiang et al., 2023), UniAD (Hu et al., 2023), EMMA (Hwang et al., 2024), OpenEMMA (Xing et al., 2025), OpenDriveVLA (Zhou et al., 2025), AutoVLA (Zhou et al., 2025), and Impromptu VLA (Chi et al., 2025). The Qwen2.5-VL row is the primary setting, while the gray Qwen3-VL rows isolate backbone and TrajCond ablations. Notably, the L2 error on nuScenes has become nearly saturated at approximately 0.30 m across existing methods, while collision rate provides a more discriminative measure.

NeuroNCAP closed-loop comparison. Table 4 reports the NeuroNCAP closed-loop comparison using NeuroNCAP Score and collision rate. The compared methods include UniAD (Hu et al., 2023), VAD (Jiang et al., 2023), BridgeAD (Zhang et al., 2025), ImagiDrive-S (Li et al., 2025), and Impromptu VLA (Chi et al., 2025), together with the additional E2E rows shown in the table. The Qwen2.5-VL AutoVLA-D row is the primary setting, and the gray rows show that removing TrajCond or switching backbone weakens the result.

4.3 TrajCond Ablation & Progressive Analysis

Progressive analysis on NAVSIM. Table 5 traces the improvement path from continuous trajectory regression to the full AutoVLA-D recipe. A one-stage Brohan TrajCond already improves over the direct continuous baseline (`C < B-TC(1)`), and separating the process into two stages improves further (`B-TC(1) < B-TC`) because TrajCond should first absorb action-oriented semantics before being used as the condition for final continuous trajectory prediction. This progression also clarifies that Brohan is weaker as the direct output target (`B<C`) but stronger as an intermediate condition (`B<C<B-TC`), suggesting that the gain of AutoVLA-D does not come from distilling discrete trajectories, but from using discrete trajectories as a condition.

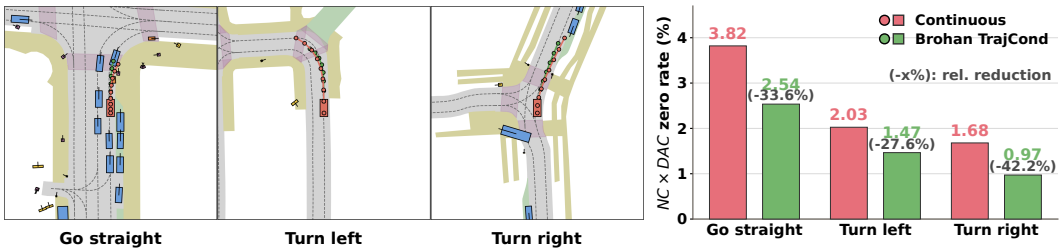


Figure 2: Collision examples and failed $NC \times DAC$ product rates on NAVSIM navtest. The left three panels show representative collision-related scenes for the three high-level intents, and the right panel reports the failed $NC \times DAC$ product rate for Continuous and Brohan TrajCond. Full NC/DAC/Product counts are provided in Supplement Table 14.

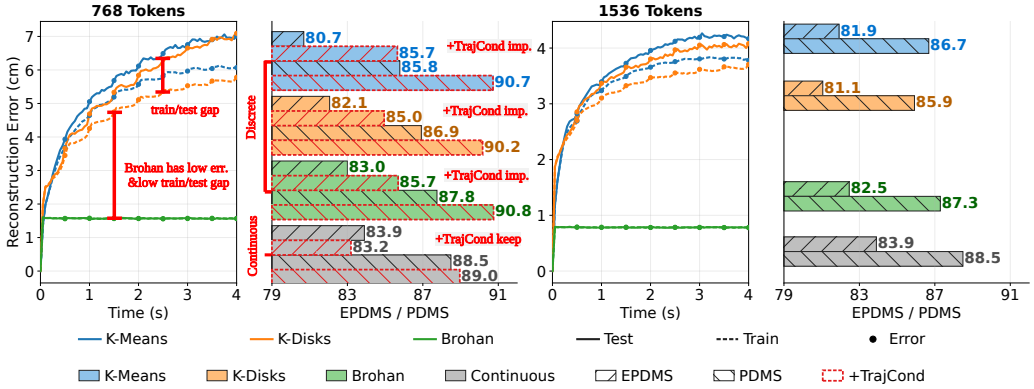


Figure 3: TrajCond form analysis for action tokens. Each codebook-size group pairs the reconstruction-error curve with the corresponding EPDMS/PDMS bars. The left axis reports trajectory reconstruction error; the right axis reports EPDMS/PDMS. Due to space limits, we move codebook-size=384 to Supplement Fig. 8.

TrajCond ablation on nuScenes/NeuroNCAP. The completed comparisons further validate AutoVLA-D beyond NAVSIM. In the SFT-only nuScenes setting, TrajCond improves open-loop planning over the w/o-TC variant under both ST-P3 and UniAD metrics. On NeuroNCAP, TrajCond also improves closed-loop safety and reduces collisions. This supports that TrajCond is not a NAVSIM-only artifact.

Attribution analysis. The closed-loop results indicate that TrajCond improves collision-related behavior most clearly. Our technical route is also close to Curious-VLA [2] except for TrajCond, and compared with this baseline, the main gain comes from the reduction of collision rate. We therefore further analyze NAVSIM collision-related metrics in Fig. 2. The figure focuses on No Collisions (NC) and Drivable Area Compliance (DAC), and shows representative cases where the high-level meta-behavior is already correct, but TrajCond produces more accurate local control and avoids collision-prone motion. The ablation rows in Tables 3 and 4 show a consistent trend on nuScenes and NeuroNCAP, suggesting that TrajCond improves safety-critical ego control rather than only route progress.

4.4 TrajCond Form Analysis

Discretization error and generalization. Following Trajeglish [20], we measure the reconstruction error introduced by discretizing continuous trajectories into action tokens. Fig. 3/ 8 (in supplement) shows a trend that $Q = 384 \rightarrow 768 \rightarrow 1536$, larger codebooks reduce reconstruction error and that 1D-Brohan consistently gives the smallest reconstruction error with a smaller train-test gap than clustering-based codebooks; Supplement Table 11 quantifies the same shift.

TrajCond as final prediction. When used as direct autoregressive output targets, all discrete representations remain weaker than continuous text waypoints (`grey bar` *v.s.* `blue/yellow/green bars` in Fig. 3), even when Brohan gives the best discrete result . The K-Means/K-Disks comparison also shows that planning performance is not determined by oracle reconstruction error alone.

Table 6: TrajCond semantic validation on NAVSIM with Qwen3-VL-4B. Left: reasoning-dependence ablation. Except for the second row, which uses a standalone 6-epoch run, all left-table settings use the same 3+3 stage allocation. Right: perplexity analysis under pause-token replacement. In both tables, B, R, and C denote Brohan, Reason, and Continuous; Pause is written in full. Absolute perplexity values are provided in Supplement Table 13.

Method	Stage 1	Stage 2	RFT	EPDMS/PDMS
C-TC	R → C	C → R → C	–	83.2 / 89.0
B-TC	R → B	B → R → C	–	84.9 / 90.6
B-TC w/o TrajCond	R → C (6 ep.)	–	–	83.9 / 88.5
B-TC w/o R	R → B	B → Pause → C	–	85.3 / 90.5
B-TC w/o R	R → B	B → Pause → C	✓	84.1 / 89.1
B-TC	R → B	B → R → C	✓	**86.8 / 92.0**

Setting	Cond.	Target	ΔPPL
nuScenes	B→Pause	C	+0.0094
NAVSIM	B→Pause	C	+0.0011
	R→Pause	C	+0.0012
	R→Pause	B	+0.0002

Ablation on TrajCond form. We further evaluate the codebook-size 768 representations as trajectory-format semantic conditions. In Fig. 3, all three discrete tokenizers become much stronger when used as TrajCond, whereas using continuous trajectories as the condition gives only a limited gain. This confirms that action tokens are most useful as an intermediate interface rather than a replacement for final continuous prediction. Since 1D-Brohan gives both the lowest reconstruction error and the best TrajCond score, we use it as the default representation.

4.5 TrajCond Semantic Analysis

Reasoning Dependence and Perplexity Analysis. Pause tokens follow Goyal et al. [5] and serve only as a structure-preserving placeholder, *i.e.*, be used to replace the meaningful content with `[PAUSE]` proxy token. **(I) The left half of Table** 6 tests whether TrajCond truly learns semantics rather than acting only as a geometric auxiliary signal. The Stage-1 comparison already supports this point: removing TrajCond and reverting to continuous regression degrades performance (`B-TC>B-TC w/o TrajCond`). We then keep the same two-stage SFT recipe but remove the Stage-2 textual reason and replace it with pause tokens, so the second-stage prediction must rely on TrajCond instead of re-reading explicit reasoning text. The resulting gain (`B-TC<B-TC w/o R`) suggests that once reasoning semantics have been absorbed into TrajCond, the trajectory-structured condition can provide cleaner guidance than injecting reason text again in Stage 2. **(II) The right half of** Table 6 provides a complementary decoding view. On nuScenes, replacing action tokens with pause tokens increases the perplexity of the final continuous trajectory. On NAVSIM Stage 2, replacing either action tokens or reasoning with pause tokens also increases perplexity, with the reasoning replacement causing the largest degradation. On NAVSIM Stage 1, replacing reasoning with pause tokens similarly increases the perplexity of TrajCond prediction itself. Together with the reasoning-dependence ablation, these results suggest that both TrajCond and reasoning reduce the uncertainty of trajectory decoding, while TrajCond serves as a semantically meaningful trajectory-geometric condition rather than a purely geometric auxiliary signal. **(III)** `C-TC<B-TC` proves that the gains is not from longer output/auxiliary loss but from the semantic nature of Brohan action tokens. **(IV)** The RFT result further clarifies the role of TrajCond beyond the two SFT stages. Starting from the w/o-R variant, GRPO reduces performance, whereas the full B-TC + RFT pipeline remains strongest. This suggests that TrajCond is a strong complement to reasoning, but not a full replacement for textual reasoning during RFT, likely because RFT is harder to improve without reliable CoT reasoning support [21].

T-SNE Visualization. We further inspect the hidden states of the discrete trajectory used as TrajCond on NAVSIM navtest. We extract the motion direction/speed behavior from the CoT reasoning meta behavior. In Fig. 4, the left panel shows grouping by motion direction, while the right panel shows grouping by speed behavior. The t-SNE structure reveals tendency boundaries between different motion types. Interestingly, samples within the same action category still exhibit substantial spread rather than collapsing into a single compact center, suggesting that TrajCond captures rich semantic variation beyond mere action labels.

5 Conclusion & Limitation

Conclusion. In this paper, we introduce a discrete trajectory as TrajCond and propose a two-stage framework for using it in VLA planning. Our analyses lead to three conclusions. First, direct

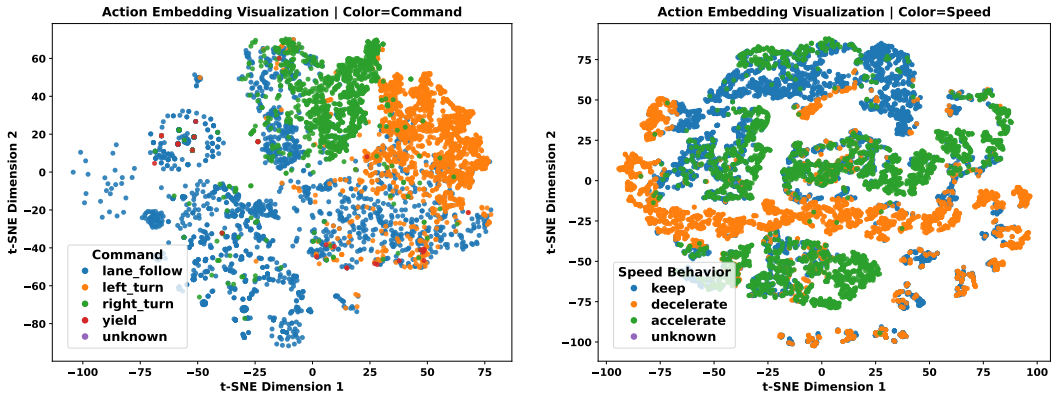


Figure 4: t-SNE visualization of hidden states from the discrete trajectory used as TrajCond. Left: points are colored by instruction motion direction. Right: points are colored by speed behavior.

regression on discrete trajectories is weaker than direct regression on continuous trajectories. Second, when used as TrajCond, discrete trajectories are more effective than continuous-trajectory conditions and consistently improve final continuous trajectory prediction. Third, this gain comes from TrajCond absorbs useful planning semantics rather than auxiliary losses/ output length.

Limitation. AutoVLA-D adds decoding/training cost through the intermediate trajectory condition, and some low-frequency action tokens may be insufficiently learned.

Acknowledge

This work is funded by DiDi and Wuxi Research Institute of Applied Technologies, Tsinghua University under Grant 20242001120.

References

[1] 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 *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 11618–11628. IEEE, 2020.

[2] Canyu Chen, Yuguang Yang, Zhewen Tan, Yizhi Wang, Ruiyi Zhan, Haiyan Liu, Xuanyao Mao, Jason Bao, Xinyue Tang, Linlin Yang, et al. Devil is in narrow policy: Unleashing exploration in driving vla models. *arXiv preprint arXiv:2603.06049*, 2026.

[3] Haohan Chi, Huan-ang Gao, Ziming Liu, Jianing Liu, Chenyu Liu, Jinwei Li, Kaisen Yang, Yangcheng Yu, Zeda Wang, Wenyi Li, et al. Impromptu vla: Open weights and open data for driving vision-language-action models. *arXiv preprint arXiv:2505.23757*, 2025.

[4] 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.

[5] Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations (ICLR)*, 2024.

[6] 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*, pages 533–549. Springer, 2022.

[7] 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*, pages 17853–17862, 2023.

[8] Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, et al. Emma: End-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2410.23262*, 2024.

[9] 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*, pages 8340–8350, 2023.

[10] 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.

[11] 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. *arXiv preprint arXiv:2510.12796*, 2025.

[12] Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, et al. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. *arXiv preprint arXiv:2506.08052*, 2025.

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

[14] William Ljungbergh, Adam Tonderski, Joakim Johnander, Holger Caesar, Kalle Åström, Michael Felsberg, and Christoffer Petersson. Neuroncap: Photorealistic closed-loop safety testing for autonomous driving. In *European Conference on Computer Vision*, pages 161–177. Springer, 2024.

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[16] 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.

[17] Yuechen Luo, Qimao Chen, Fang Li, Shaoqing Xu, Jaxin Liu, Ziying Song, Zhi-Xin Yang, and Fuxi Wen. Unleashing vla potentials in autonomous driving via explicit learning from failures. *arXiv preprint arXiv:2603.01063*, 2026.

[18] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[19] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[20] Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajeglish: Traffic modeling as next-token prediction. In *The Twelfth International Conference on Learning Representations*, 2024.

[21] Salman Rahman, Jingyan Shen, Anna Mordvina, Hamid Palangi, Saadia Gabriel, and Pavel Izmailov. When can llms learn to reason with weak supervision? *arXiv preprint arXiv:2604.18574*, 2026.

[22] Luke Rowe, Rodrigue de Schaetzen, Roger Girgis, Christopher Pal, and Liam Paull. Poutine: Vision-language-trajectory pre-training and reinforcement learning post-training enable robust end-to-end autonomous driving. *arXiv preprint arXiv:2506.11234*, 2025.

[23] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[24] Wenchao Sun, Xuewu Lin, Yining Shi, Chuang Zhang, Haoran Wu, and Sifa Zheng. Sparsedrive: End-to-end autonomous driving via sparse scene representation. *ICRA*, 2025.

[25] Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time motion generation via next-token prediction. *Advances in Neural Information Processing Systems*, 37:114048–114071, 2024.

[26] Chengen Xie, Chonghao Sima, Tianyu Li, Bin Sun, Junjie Wu, Zhihui Hao, and Hongyang Li. Flare: Learning future-aware latent representations from vision-language models for autonomous driving. *arXiv preprint arXiv:2601.05611v2*, 2026.

[27] Shuo Xing, Chengyuan Qian, Yuping Wang, Hongyuan Hua, Kexin Tian, Yang Zhou, and Zhengzhong Tu. Openemma: Open-source multimodal model for end-to-end autonomous driving. In *Proceedings of the Winter Conference on Applications of Computer Vision*, pages 1001–1009, 2025.

[28] Bozhou Zhang, Nan Song, Xin Jin, and Li Zhang. Bridging past and future: End-to-end autonomous driving with historical prediction and planning. *arXiv preprint arXiv:2503.14182*, 2025.

[29] Xingcheng Zhou, Xuyuan Han, Feng Yang, Yunpu Ma, and Alois C Knoll. Opendrivevla: Towards end-to-end autonomous driving with large vision language action model. *arXiv preprint arXiv:2503.23463*, 2025.

[30] Zewei Zhou, Tianhui Cai, Seth Z 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. *arXiv preprint arXiv:2506.13757*, 2025.

[31] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

A Computation Resources

All experiments—including two-stage model training, reinforcement fine-tuning, and comprehensive benchmark evaluations—were conducted on a high-performance internal computing cluster. Specifically, the compute node utilized is equipped with 8 NVIDIA H100 GPUs, 2TB of system memory (RAM), and 192 CPU cores. Under this hardware configuration, the complete suite of experiments required approximately one week of continuous execution time.

B Benchmark Details

We provide brief benchmark details for the three evaluation settings used in the main paper.

NAVSIM navtest. Following Curious-VLA [2], we use NAVSIM navtest as the primary large-scale non-reactive closed-loop simulation benchmark. It focuses on planning evaluation under challenging driving scenarios and is reported here under the same protocol family as ReCogDrive [12], using PDMS and EPDMS on the official navtest split.

nuScenes. Following Curious-VLA [2] and Impromptu VLA [3], we use nuScenes as the real-world open-loop benchmark for auxiliary generalization evaluation. nuScenes contains 1000 driving scenes of 20 seconds each and is evaluated here under the standard 3-second planning metrics, including L2 distance error and collision rate.

NeuroNCAP. Following ImagiDrive [10], we use NeuroNCAP as the safety-critical closed-loop benchmark. NeuroNCAP is a photorealistic simulator built on nuScenes and is designed to stress-test driving policies under interactive closed-loop rollout. We report NeuroNCAP Score together with collision rate under the same evaluation setting used in prior VLA comparison work.

C System Prompt

For NAVSIM, we use Qwen3-VL-235B to generate pseudo-labeled CoT reasoning. The prompt is adapted from DriveTeach-VLA by keeping the reasoning subtasks and removing the future-trajectory-prediction task.

You are an expert driver. Suppose you are driving. Let's complete the following tasks step by step.
Input: front-view image, high-level intent $\in$ {GO STRAIGHT, TURN RIGHT, TURN LEFT}, 1.5s past ego trajectory, and 4s expert future trajectory. Each trajectory point is $(x, y, \text{heading})$.
Task 1: Critical objects. For each class, answer yes/no on whether it can influence the ego future path: nearby vehicle, conflicting pedestrian, cyclist, construction, traffic element, weather condition, road hazard, emergency vehicle, animal, special vehicle, conflicting vehicle, door-opening vehicle.
Task 2: Explanation. Give a concise explanation of the expert 4s trajectory. Mention only yes-marked classes and explain how they affect the trajectory.
Task 3: Meta-behavior. Select one speed label from {keep, accelerate, decelerate, other} and one command label from {straight, yield, left turn, right turn, lane follow, lane change left, lane change right, reverse, other}.
Output: strict JSON with keys `critical_objects`, `explanation`, and `meta_behaviour`; no extra keys, markdown, or commentary.

Figure 5: Prompt for Qwen3-VL-235B to pseudo-label NAVSIM CoT reasoning data.

NAVSIM Stage-1 user prompt. The Stage-1 NAVSIM training prompt asks the model to predict TrajCond action tokens from the image, intent, and past trajectory:

```
You are an expert driver. Suppose you are driving. Let's complete the following tasks step by step.
Input:
- 1 frame of front-view image collected from the ego-vehicle at the present timestep
Picture 1: <image> the front view of the ego-vehicle
- Current high-level intent (string): go straight
- 1.5-second past trajectory(3 steps at 2 Hz):    - t-3: (0.0, 0.0, 0.0)    - t-2: (0.0, 0.0, 0.0)    - t-1: (0.0, 0.0,
↪  0.0)    - t-0: (0.0, 0.0, 0.0)
Each trajectory point format: (x:float, y:float, heading:float)
Task 1: Critical Objects and Conditions Detection
Decide whether at least one critical instance of each class could influence the ego-vehicle's future path (no
↪  omissions). A vehicle can be a car, bus, truck, motorcyclist, scooter, etc. traffic_element includes traffic signs
↪  and traffic lights. road_hazard may include hazardous road conditions, road debris, obstacles, etc. A
↪  conflicting_vehicle is a vehicle that may potentially conflict with the ego's future path. Output "yes" or "no" for
↪  every class (no omissions).
   Object classes to audit:
```

```
      - nearby_vehicle
      - conflicting_pedestrian
      - cyclist
      - construction
      - traffic_element
      - weather_condition
      - road_hazard
      - emergency_vehicle
      - animal
      - special_vehicle
      - conflicting_vehicle
      - door_opening_vehicle

Task 2: Natural Language Explanation
(thinking output should be wrapped in <thinking>...</thinking>)
Compose a 100-word concise natural-language description of the optimal future 4-second trajectory for the ego vehicle
↪  that the expert driver (you) plans and explain why the expert driver plans to execute this trajectory.
   - Mention only the classes you marked "yes" in the previous task.
   - Describe how each of those critical objects or conditions influences the optimal trajectory.
   - Do not invent objects or conditions not present in the input.

Task 3: Meta-Behaviour Selection
Assign exactly one category from each list. Choose the label that best summarises the overall behaviour of the optimal
↪  future trajectory:
   - speed ? { keep, accelerate, decelerate }
   - command ? { straight, yield, left_turn, right_turn, lane_follow, lane_change_left, lane_change_right, reverse }
   Choose the label that best summarises the overall behaviour of the optimal future trajectory.
   - If none fits, use 'other', but do this sparingly.

Task 4: Future Trajectory Prediction
(answer output should be wrapped in <answer>...</answer>)
Given the input, critical objects/conditions, natural language explanation, and meta-behaviour, predict the optimal
↪  4-second future trajectory (8 steps at 2 Hz) of the ego vehicle. Predict 8 groups of three future trajectory action
↪  tokens in [AT, ...] format.
Output format (strict JSON, no extra keys, no markdown codeblock chars(```), no commentary):
{
  "critical_objects": {
    "nearby_vehicle": "yes | no",
    "conflicting_pedestrian": "yes | no",
    "cyclist": "yes | no",
    "construction": "yes | no",
    "traffic_element": "yes | no",
    "weather_condition": "yes | no",
    "road_hazard": "yes | no",
    "emergency_vehicle": "yes | no",
    "animal": "yes | no",
    "special_vehicle": "yes | no",
    "conflicting_vehicle": "yes | no",
    "door_opening_vehicle": "yes | no"
  },
  "explanation": "100-word description that references only the classes marked 'yes'",
  "meta_behaviour": {
    "speed": "keep | accelerate | decelerate | other",
    "command": "straight | yield | left_turn | right_turn | lane_follow | lane_change_left | lane_change_right | reverse
    ↪  | other"
  },
  "future_trajectory": "<answer>[AT, ...]</answer>"
}
```

NAVSIM Stage-2 user prompt. The Stage-2 NAVSIM training prompt asks the model to use the discrete action-token plan, environment reasoning, and driving logic before producing the final continuous trajectory:

```
You are an expert driver. Suppose you are driving. Let's complete the following tasks step by step.
Input:
- 1 frame of front-view image collected from the ego-vehicle at the present timestep
Picture 1: <image> the front view of the ego-vehicle
- Current high-level intent (string): go straight
- 1.5-second past trajectory(3 steps at 2 Hz):    - t-3: (0.0, 0.0, 0.0)    - t-2: (0.0, 0.0, 0.0)    - t-1: (0.0, 0.0,
↪  0.0)    - t-0: (0.0, 0.0, 0.0)
Each trajectory point format: (x:float, y:float, heading:float)
Task 1: Discrete Action Trajectory Planning
(answer output should be wrapped in <answer>...</answer>)
Based on the road geometry and intent, plan a naive "Discrete Action Trajectory" (8 steps at 2 Hz). Assume no dynamic
↪  obstacles (e.g., follow lane center, maintain speed).
Predict 8 future movement actions in [AT, ...] format.

Task 2: Critical Objects and Conditions Detection
Identify critical objects/conditions that might force a deviation from the Discrete Action Trajectory. Output "yes" or
↪  "no" for every class.
   Object classes to audit:
     - nearby_vehicle
     - conflicting_pedestrian
     - cyclist
     - construction
     - traffic_element
     - weather_condition
     - road_hazard
     - emergency_vehicle
     - animal
     - special_vehicle
```

```
      - conflicting_vehicle
      - door_opening_vehicle

Task 3: Meta-Behaviour Selection
Assign exactly one category from each list to describe the *Final* behavior:
   - speed ? { keep, accelerate, decelerate }
   - command ? { straight, yield, left_turn, right_turn, lane_follow, lane_change_left, lane_change_right, reverse }


Task 4: Natural Language Explanation
(thinking output should be wrapped in <thinking>...</thinking>)
Compose a 100-word concise description explaining the driving process.
   - Contrast the Discrete Action Plan with the environment.
   - Explain why the Discrete Action Plan must be adjusted to the Final Plan based on the visual evidence.

Task 5: Text Nuemeric Trajectory Planning
(answer output should be wrapped in <answer>...</answer>)
Based on the reasoning in Task 4, plan the optimal "Text Nuemeric Trajectory" (8 steps at 2 Hz) that accounts for all
↪  safety factors.
Predict 8 trajectory points in [PT, ...] format.

Output format (strict JSON, no extra keys, no markdown codeblock chars(```), no commentary):
{
  "discrete_action_trajectory": "<answer>[AT, ...]</answer>",
  "critical_objects": {
    "nearby_vehicle": "yes | no",
    "conflicting_pedestrian": "yes | no",
    "cyclist": "yes | no",
    "construction": "yes | no",
    "traffic_element": "yes | no",
    "weather_condition": "yes | no",
    "road_hazard": "yes | no",
    "emergency_vehicle": "yes | no",
    "animal": "yes | no",
    "special_vehicle": "yes | no",
    "conflicting_vehicle": "yes | no",
    "door_opening_vehicle": "yes | no"
  },
  "meta_behaviour": {
    "speed": "keep | accelerate | decelerate | other",
    "command": "straight | yield | left_turn | right_turn | lane_follow | lane_change_left | lane_change_right | reverse
    ↪  | other"
  },
  "explanation": "100-word description of the driving logic",
  "future_trajectory": "<answer>[PT, ...]</answer>"
}
```

nuScenes and NeuroNCAP user prompt. For nuScenes and NeuroNCAP, Stage 1 and Stage 2 share the same user prompt template and differ only in the expected output target. The first line in the file is the prompt, followed by example Stage-1 and Stage-2 outputs:

```
You are an autonomous driving agent. You have access to a front view camera image of a vehicle <image>. Your task is to
↪  do your best to predict future movement actions for the vehicle over the next 6 timesteps, given the vehicle's
↪  intent inferred from the images. Provided are the previous ego vehicle status recorded over the last 3.0 seconds (at
↪  0.5-second intervals). This includes the x and y coordinates of the ego vehicle. Positive x means forward direction
↪  while positive y means leftwards. The data is presented in the format [x, y]:. (t-3.0s) [-19.78, -0.68],
↪  Acceleration: X +0.72, Y -0.43 m/s^2, Velocity: +5.72 m/s, Steering angle: -0.21 (positive: left turn, negative:
↪  right turn), (t-2.5s) [-16.86, -0.34], Acceleration: X +0.71, Y -1.03 m/s^2, Velocity: +5.93 m/s, Steering angle:
↪  -0.62 (positive: left turn, negative: right turn), (t-2.0s) [-13.73, -0.12], Acceleration: X +0.83, Y -1.07 m/s^2,
↪  Velocity: +6.19 m/s, Steering angle: -0.53 (positive: left turn, negative: right turn), (t-1.5s) [-10.50, -0.02],
↪  Acceleration: X +1.03, Y -0.69 m/s^2, Velocity: +6.52 m/s, Steering angle: -0.20 (positive: left turn, negative:
↪  right turn), (t-1.0s) [-7.05, +0.01], Acceleration: X +1.08, Y -0.41 m/s^2, Velocity: +6.87 m/s, Steering angle:
↪  -0.10 (positive: left turn, negative: right turn), (t-0.5s) [-3.42, 0.00], Acceleration: X +1.00, Y -0.38 m/s^2,
↪  Velocity: +7.26 m/s, Steering angle: -0.04 (positive: left turn, negative: right turn), (t-0.0s) [0.00, 0.00],
↪  Acceleration: X +0.95, Y -0.43 m/s^2, Velocity: +7.56 m/s, Steering angle: -0.02 (positive: left turn, negative:
↪  right turn)



------------stage1 output------------


<PLANNING>Predicted future movement details for the next 3.0 seconds (sampled at 0.5-second intervals), including
↪  movement actions in x, y directions and heading rotation. For each timestep, only output three action tokens in
↪  brackets separated by comma. The action output is formatted as [<action_x>,<action_y>,<action_h>]:
↪  [<action_110>,<action_382>,<action_636>], [<action_110>,<action_383>,<action_638>],
↪  [<action_116>,<action_387>,<action_637>], [<action_120>,<action_388>,<action_638>],
↪  [<action_131>,<action_384>,<action_637>], [<action_124>,<action_383>,<action_637>]</PLANNING>


------------stage2 output------------


<PLANNING>Predicted future movement details for the next 3.0 seconds (sampled at 0.5-second intervals), including
↪  movement actions in x, y directions and heading rotation. For each timestep, only output three action tokens in
↪  brackets separated by comma. The action output is formatted as [<action_x>,<action_y>,<action_h>]:
↪  [<action_110>,<action_382>,<action_636>], [<action_110>,<action_383>,<action_638>],
↪  [<action_116>,<action_387>,<action_637>], [<action_120>,<action_388>,<action_638>],
↪  [<action_131>,<action_384>,<action_637>], [<action_124>,<action_383>,<action_637>]
```

```
Now I'm going to accumulate the 6 action groups and convert them into text waypoints for each time step. Let's conduct
↪  comprehensive safety assessment, identify risks like collision, overturn, boundary crossing or leaving drivablle
↪  area, mark high-risk objects, and propose optimized trajectories to avoid dangers while retaining the original
↪  motion purpose. Predicted future movement details for the next 3 seconds (sampled at 0.5-second intervals),
↪  including BEV location in x and y directions (in meters). Larger x means forward direction while larger y means
↪  leftwards. The output is formatted as [x, y]: [+3.97, -0.03], [+8.00, -0.07], [+12.23, -0.08], [+16.65, -0.09],
↪  [+21.53, -0.10], [+26.11, -0.12]</PLANNING>
```

D Trajectory Differential Illustration

Fig. 6 illustrates the coordinate transform used to compute differential motions. Although the trajectory waypoints are expressed in the ego frame at $t = 0$, the transition from step $t-1$ to t is represented in the local frame at step $t-1$. Fig. 7 visualizes the iterative trajectory-tokenization process, and Alg. 1 gives the complete 1D-Brohan codebook construction and action-token assignment procedure used by AutoVLA-D.

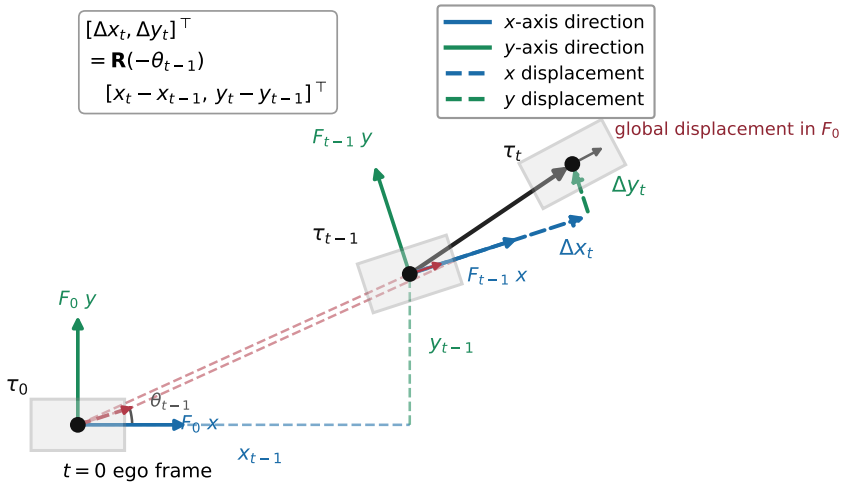


Figure 6: Illustration of local-frame differential motion. The displacement between τ_{t-1} and τ_t is first observed in the $t = 0$ ego frame, then rotated by $\mathbf{R}(-\theta_{t-1})$ into the local frame at step $t-1$ to obtain $(\Delta x_t, \Delta y_t)$.

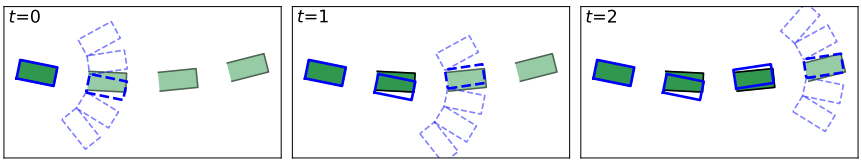


Figure 7: Trajectory tokenization illustration **Copied From Trajeglish** [20]. An example trajectory is shown in green. The raw representation of the tokenized trajectory is shown as boxes with blue outlines. States that have yet to be tokenized are light green. Token templates are optimized to minimize the error between the tokenized trajectories and the raw trajectories.

Algorithm 1 1D-Brohan codebook construction and action-token assignment

Require: Training trajectories $\mathcal{D}$, target trajectory $\tau = \{(x_t, y_t, \theta_t)\}_{t=1}^{T}$, and bin counts $\{Q_i\}_{i=1}^{3}$
Ensure: Codebooks $\mathbf{x}$, $\mathbf{y}$, $\boldsymbol{\theta}$ and action-token sequence $\mathbf{a} = \{a_t\}_{t=1}^{T}$

Compute local-frame differential motions $\delta_t = (\Delta x_t, \Delta y_t, \Delta\theta_t)$ for all trajectories in $\mathcal{D}$ using Eq. 4:
$\Delta x_t = (x_t - x_{t-1})\cos\theta_{t-1} + (y_t - y_{t-1})\sin\theta_{t-1}$.
$\Delta y_t = -(x_t - x_{t-1})\sin\theta_{t-1} + (y_t - y_{t-1})\cos\theta_{t-1}$, $\Delta\theta_t = \theta_t - \theta_{t-1}$.
We adopt the 1D-Brohan codebook in Alg. 1. For each DoF, take the 1st and 99th quantiles of the training differentials as the discretization range.
$r_i = \left[\mathrm{Quantile}_{1\%}(\{\delta_t^i\}), \mathrm{Quantile}_{99\%}(\{\delta_t^i\})\right]$, $i \in \{1, 2, 3\}$.
Uniformly divide the three ranges into Q_1, Q_2, Q_3 bins to obtain codebooks $\mathbf{x}$, $\mathbf{y}$, and $\boldsymbol{\theta}$.
Initialize $\hat{\tau}_0 = (0, 0, 0)$.
for $t = 1$ to T **do**
    Select the four nearest tokens in each DoF by absolute residual from the local-frame differential motion.
    Enumerate the resulting candidate set $\mathcal{K}_t$ of $4 \times 4 \times 4$ tuples.
    for each $\mathbf{k} = (k_1, k_2, k_3) \in \mathcal{K}_t$ **do**
        Render $\bar{\tau}_t(\mathbf{k}) = (\bar{x}_t(\mathbf{k}), \bar{y}_t(\mathbf{k}), \bar{\theta}_t(\mathbf{k}))$ from $\hat{\tau}_{t-1}$ using Eq. 6:
        $\begin{bmatrix} \bar{x}_t(\mathbf{k}) \\ \bar{y}_t(\mathbf{k}) \end{bmatrix} = \begin{bmatrix} \hat{x}_{t-1} \\ \hat{y}_{t-1} \end{bmatrix} + \mathbf{R}(\hat{\theta}_{t-1}) \begin{bmatrix} x_{k_1} \\ y_{k_2} \end{bmatrix}$.
        $\bar{\theta}_t(\mathbf{k}) = \hat{\theta}_{t-1} + \theta_{k_3}$.
    end for
    $\mathbf{k}_t^* \leftarrow \arg\min_{\mathbf{k} \in \mathcal{K}_t} d_{\mathrm{corner}}(\bar{\tau}_t(\mathbf{k}), \tau_t)$.
    $a_t \leftarrow \mathbf{k}_t^*$ and $\hat{\tau}_t \leftarrow \bar{\tau}_t(\mathbf{k}_t^*)$.
end for
return $\mathbf{a}$

Table 7: Additional baselines on NAVSIM navtest under PDMS. These rows are moved from the main table to keep the main comparison focused on stronger VLA methods.

Method	Base Model	NC↑	DAC↑	EP↑	TTC↑	C↑	PDMS↑
Ego-MLP	–	93.1	78.3	63.2	84.0	100.0	66.4
UniAD	–	97.7	92.8	79.2	92.8	100.0	84.0
DiffusionDrive	–	98.2	96.2	82.2	94.7	100.0	88.1
WoTE	–	98.5	96.8	81.9	94.4	99.9	88.3
Qwen2.5-VL	Qwen2.5-VL-7B	97.8	92.1	78.3	92.8	100.0	83.3
InternVL3	InternVL3-8B	97.0	92.4	78.9	91.8	100.0	83.3

Table 8: Additional E2E baselines on NAVSIM navtest under EPDMS. These rows are moved from the main table to keep the main comparison focused on VLA methods.

Method	NC↑	DAC↑	DDC↑	TLC↑	EP↑	TTC↑	LK↑	C↑	EC↑	EPDMS↑
Ego-MLP	93.1	77.9	92.7	99.6	86.0	91.5	89.4	98.3	85.4	64.0
VADv2	97.3	91.7	98.2	99.9	77.6	92.7	66.0	100.0	97.4	76.6
TransFuser	96.9	89.9	97.8	99.7	87.1	95.4	92.7	98.3	87.2	76.7
HydraMDP++	97.2	97.5	99.4	99.6	83.1	96.5	94.4	98.2	70.9	81.4
DriveSuprim	97.5	96.5	99.4	99.6	88.4	96.6	95.5	98.3	77.0	83.1
ARTEMIS	98.3	95.1	98.6	99.8	81.5	97.4	96.5	98.3	–	83.1

E Progressive Analysis Data

Table 9 gives the numeric values plotted in the right panel of Table 5. The stage descriptions follow the same abbreviations as the main paper: Reason, Continuous, Brohan, and epochs are denoted by R, C, B, and ep., respectively.

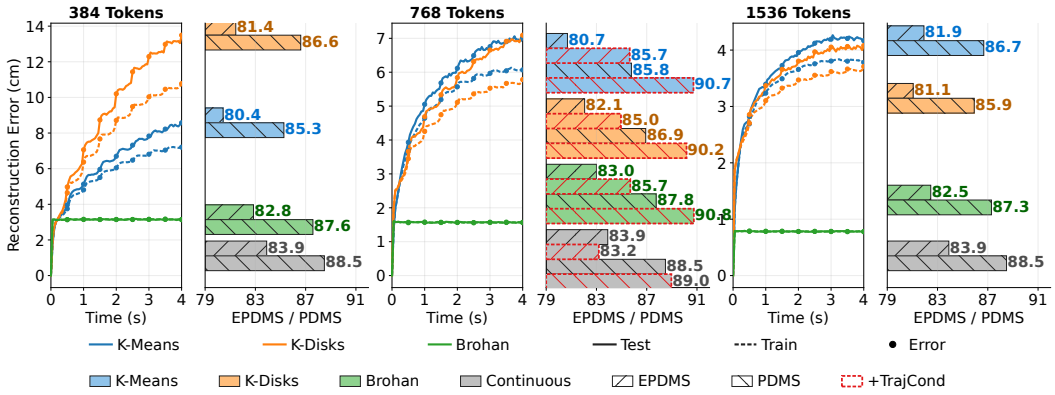


Figure 8: Complete trajectory-representation analysis across codebook sizes 384, 768, and 1536. The figure reports reconstruction error for each discretization method together with the corresponding PDMS/EPDMS performance bars.

Table 9: Detailed progressive-analysis data behind Table 5.

Method	**Stage 1**	**Stage 2**	**RFT**	**EPDMS↑**	**PDMS↑**
Brohan (B)	R → B (6 ep.)	–	–	82.4	86.8
Continuous (C)	R → C (6 ep.)	–	–	83.9	88.5
Brohan TrajCond (1)	–	B → R → C (6 ep.)	–	84.0	89.2
Brohan TrajCond	R → B (3 ep.)	B → R → C (3 ep.)	–	85.6	90.8
Continuous + RFT	R → C (6 ep.)	–	✓	84.9	89.7
Brohan TrajCond + RFT	R → B (3 ep.)	B → R → C (3 ep.)	✓	86.8	92.0

F Complete Trajectory-Representation Analysis

Fig. 8 extends the main-paper trajectory-representation analysis in Fig. 3 by adding the smaller codebook-size 384 setting. The reconstruction-error curves show the same qualitative behavior as the main figure: larger codebooks reduce discretization error, 1D-Brohan remains the most stable discrete representation, and clustering-based K-Means and K-Disks show larger train-test gaps than rule-based Brohan. Continuous text waypoints are plotted as the zero-discretization-error reference because they do not require tokenization before trajectory regression.

Table 10 reports the reconstruction-error values behind the curves in Fig. 8. Values are sampled at 1, 2, 3, and 4 seconds from the 80-step evaluation trajectory and are reported in centimeters.

Table 10: Trajectory reconstruction error values behind Fig. 8. Lower is better.

Codebook	**Tokenizer**	**Train error (cm)**				**Test error (cm)**			
		1s	**2s**	**3s**	**4s**	**1s**	**2s**	**3s**	**4s**
384	3D K-Means	4.81	6.04	6.81	7.21	5.13	6.70	7.83	8.58
384	3D K-Disks	6.37	8.71	10.03	10.77	7.06	10.20	12.29	13.49
384	1D-Brohan	3.16	3.15	3.15	3.14	3.15	3.14	3.15	3.14
768	3D K-Means	4.64	5.57	5.97	6.07	5.05	6.12	6.75	7.01
768	3D K-Disks	4.25	5.11	5.53	5.78	4.69	5.84	6.65	7.09
768	1D-Brohan	1.57	1.56	1.57	1.57	1.58	1.57	1.57	1.56
1536	3D K-Means	3.24	3.66	3.81	3.78	3.39	3.96	4.19	4.17
1536	3D K-Disks	3.10	3.47	3.62	3.71	3.37	3.80	4.05	4.08
1536	1D-Brohan	0.78	0.78	0.78	0.78	0.79	0.78	0.78	0.78

Table 11: Train/test action-token distribution shift measured by Jensen-Shannon divergence. Lower is better.

Codebook size	3D K-Means	3D K-Disks	1D-Brohan
384	0.0108958	0.0107432	**0.0039045**
768	0.0126990	0.0135494	**0.0044001**
1536	0.0167875	0.0175003	**0.0051211**

To quantify the train/test distribution shift behind this trend, Table 11 reports the Jensen-Shannon (JS) divergence between the training and test action-token distributions for each tokenizer and codebook size. Lower JS divergence indicates a closer match between training and held-out token distributions. Across all codebook sizes, 1D-Brohan has the smallest divergence, while 3D K-Means and 3D K-Disks show larger train/test distribution shifts. This supports the main-paper interpretation that Brohan-style per-dimension discretization produces more stable action-token labels under held-out trajectories.

Table 12 reports the PDMS/EPDMS bar values shown in Fig. 8. Direct output means that the action-token representation is used as the final autoregressive output target. TrajCond means that the representation is used as an intermediate condition before predicting the final continuous trajectory.

Table 12: PDMS/EPDMS score values behind Fig. 8. Higher is better.

Codebook	Representation	Direct output		TrajCond	
		EPDMS↑	PDMS↑	EPDMS↑	PDMS↑
384	3D K-Means	80.43	85.27	–	–
384	3D K-Disks	81.44	86.61	–	–
384	1D-Brohan	82.85	87.58	–	–
384	Continuous	83.90	88.49	–	–
768	3D K-Means	80.69	85.78	85.65	90.72
768	3D K-Disks	82.06	86.92	84.95	90.17
768	1D-Brohan	83.00	87.75	85.69	90.75
768	Continuous	83.90	88.49	83.19	88.96
1536	3D K-Means	81.92	86.68	–	–
1536	3D K-Disks	81.06	85.92	–	–
1536	1D-Brohan	82.46	87.29	–	–
1536	Continuous	83.90	88.49	–	–

The 384-codebook performance bars further explain why the main paper focuses on codebook sizes 768 and 1536. When action tokens are used as direct output targets, 3D K-Means reaches 80.43/85.27 EPDMS/PDMS, 3D K-Disks reaches 81.44/86.61, and 1D-Brohan reaches 82.85/87.58. These results follow the same ordering as the main figure: Brohan is the best discrete direct-output representation, but all discrete direct-output variants remain below the continuous text-waypoint baseline of 83.90/88.49. This supports the main-paper conclusion that direct autoregressive generation of discrete action tokens is limited by both discretization error and token-prediction difficulty.

Together with the 768 and 1536 settings in the main figure, the complete study supports the default design choice used by AutoVLA-D. The best role for action tokens is not to serve as the final output format; instead, they are most useful as an intermediate trajectory-geometric condition that guides the second-stage continuous planner.

G Complete Perplexity Values

Table 13 reports the absolute perplexity values behind the main-paper pause-token replacement analysis in the right half of Table 6. All evaluations use temperature $= 1\mathrm{e}{-3}$ and top-$p = 1$. Following the main paper, we abbreviate Brohan, Reason, and Continuous as B, R, and C, while Pause is written in full.

Table 13: Absolute perplexity values for the pause-token replacement analysis. Lower is better.

Setting	Condition	Target	Mean PPL
nuScenes	original	C	1.0250
nuScenes	B→Pause	C	1.0344
NAVSIM	original	C	1.0057
NAVSIM	B→Pause	C	1.0068
NAVSIM	R→Pause	C	1.0069
NAVSIM	original	B	1.0339
NAVSIM	R→Pause	B	1.0341

H Intent-Grouped NC/DAC Statistics

Table 14 provides the full statistics behind Fig. 2. We compare AutoVLA-D with the Continuous (C) baseline and report the number of samples where the NC-related collision term is zero, the DAC-related drivable-area term is zero, and their $NC \times DAC$ product is zero. Product rates use 12,146 samples as the denominator. Lower counts and rates indicate fewer samples whose multiplicative PDMS terms are zeroed.

Table 14: Intent-grouped NC/DAC zero-count comparison against the Continuous (C) baseline. Product rates use 12,146 samples as the denominator, and reduction is computed as $(\text{Baseline} - \text{Ours})/\text{Baseline}$.

Intent	NC zero count			DAC zero count			$NC \times DAC$ zero				
	Ours	C	Reduction	Ours	C	Reduction	Ours	C	Ours rate	C rate	Reduction
Go straight	160	263	39.16%	153	209	26.79%	308	464	2.54%	3.82%	33.62%
Turn left	29	52	44.23%	151	200	24.50%	178	246	1.47%	2.03%	27.64%
Turn right	35	53	33.96%	85	162	47.53%	118	204	0.97%	1.68%	42.16%

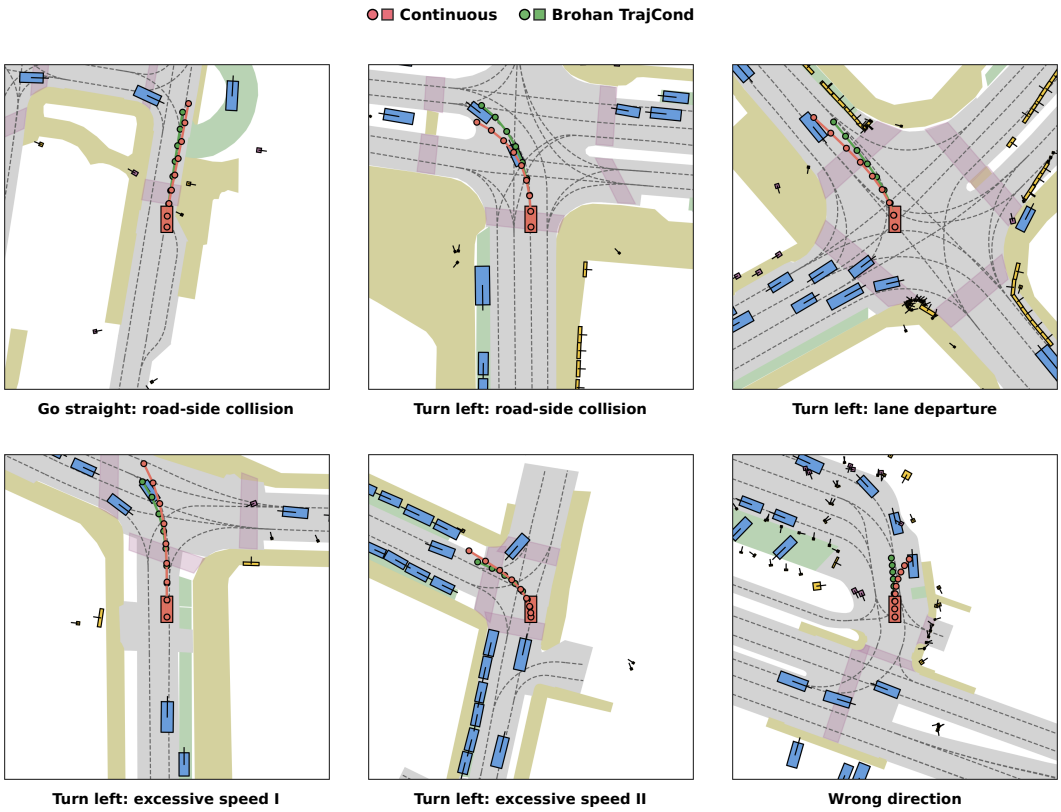


Figure 9: Additional collision-related examples for the NC/DAC analysis. The legend maps pink circular markers and square bars to Continuous, and green circular markers and square bars to Brohan TrajCond. The six panels preserve the original sample-image resolution after white-border cropping and show representative failure modes indicated by their panel labels: road-side collision, lane departure, excessive speed, and wrong direction.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately outline the core contributions, including the AutoVLA-D framework, the 1D-Brohan trajectory representation, and the performance improvements on NAVSIM, nuScenes, and NeuroNCAP, which are all supported in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 (Conclusion & Limitation) explicitly discusses the limitations, noting that TrajCond should not be seen as a full replacement for explicit textual reasoning, especially during reinforcement fine-tuning.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results or mathematical proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4.1 specifies the datasets, optimizer, learning rates, epochs, and hyperparameters, while Appendix C provides the exact system and user prompts required to reproduce the pseudo-labeled training data and model behavior.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The research is based on publicly accessible datasets (NAVSIM, nuScenes, NeuroNCAP) and open-weights models (Qwen-VL). We are very sorry that we are currently unable to organize and make the code public, though we have provided sufficient information for easy reproduction.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed training setups, including the use of AdamW, weight decay, warmup ratio, batch sizes, learning rates, and dataset splits, are provided in Section 4.1 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: For NAVSIM, We only report single-run results without error bars due to insufficient computational respources, as training and evaluating large multimodal VLA models is highly computationally expensive, especially during the GRPO rollout stage. For nuScenes and NeuroNCAP, the reported results are the mean of 10 experimental runs with different random seeds, which mitigates randomness and statistical fluctuations.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The Appendix A provides full details regarding the computational resources used, specifying the exact hardware configuration (an internal cluster with 8 NVIDIA H100 GPUs, 2TB RAM, 192 CPUs) and the approximate total execution time of one week.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research relies on standard, publicly available autonomous driving benchmarks and models, adhering to ethical norms and the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: The paper focuses on the technical development of an autonomous driving VLA planner and does not dedicate a specific section to discussing broader societal impacts. Because our proposed method demonstrably reduces driving collisions and improves safety-critical control, its application inherently yields positive societal impacts by enhancing road safety, and barely poses any negative social consequences.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not introduce or release high-risk assets, datasets, or models requiring specific safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The original assets (NAVSIM, nuScenes, NeuroNCAP, Qwen-VL variants) are properly cited in the references, Section 4, and Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release new datasets or independent software assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The research does not involve human subjects or crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The research does not involve human subjects or crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The use of LLMs (Qwen3-VL-235B) to generate pseudo-labeled CoT reasoning data is explicitly documented in Section 4.1 and the exact prompts used are listed in Appendix C.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.