ChainFlow-VLA: Causal Flow Planning with Vision-Language Models

Xiyang Wang[1*] **Xinlin Wang**[1*] **Tingguang Zhou**[1*] **Gong Chen**[12*]
Xingtai Gui[13] **Zhi Xu**[1] **Xiaolei Wu**[1]
Feiyang Tan[1] **Hangning Zhou**[1‡] **Mu Yang**[1]

[1]Afari Intelligent Drive [2]Tianjin University [3]University of Macau

Abstract

Current end-to-end autonomous driving systems are fundamentally limited by a mismatch between temporal causal reasoning and global trajectory consistency. Autoregressive (AR) models capture interaction-aware temporal dependencies via causal factorization, but their step-wise decoding leads to error accumulation and suboptimal global structure. In contrast, diffusion models optimize trajectories globally but lack explicit causal constraints, making them unreliable in interactive and safety-critical scenarios. This dichotomy reveals a deeper issue: existing methods treat causal modeling and global optimization as separate paradigms, without a principled way to unify them within a single trajectory distribution. To address this, we propose ChainFlow-VLA, which unifies causal generation and global refinement within a unified probabilistic framework. We formulate planning as a mixture over AR-induced modes and learn Vision-Language Model (VLM)-conditioned residual distributions over these modes. An autoregressive generator (**Chain**) produces a discrete set of causal trajectory modes, followed by a diffusion-based refiner (**Flow**) that leverages VLM hidden states as semantic priors to perform mode-conditioned correction in residual space while preserving causal structure. This straightforward conditioning seamlessly injects high-level scene understanding into fine-grained trajectory adjustments. Experiments demonstrate that ChainFlow-VLA achieves robust planning in ambiguous and long-tail scenarios, achieving a state-of-the-art score of **94.80** on the NAVSIM v1 leaderboard, matching human-level performance (94.8). Code: https://github.com/AFARI-Research/ChainFlow-VLA.

1 Introduction

End-to-end autonomous driving has emerged as a promising paradigm Hu et al. [2023], Jiang et al. [2023] for unified perception and planning by directly learning a mapping from sensor inputs to future trajectories. While these models can generate smooth and executable trajectories in routine scenarios Chen et al. [2024b], Sun et al. [2025], real-world driving still presents complex interactions, long-tail events, and distribution shifts Chen et al. [2024a]. Addressing such challenges requires not only geometric and motion cues, but also higher-level reasoning Li et al. [2024], Gui et al. [2026] over scene semantics, agent intent, and implicit traffic rules. Stronger semantic understanding and reasoning are therefore essential for robust end-to-end driving Sima et al. [2023], Hwang et al. [2024].

*The authors contributed equally and are listed in no particular order
†Project Leader
‡Corresponding author: zhouhangning@qianli-drive.com

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

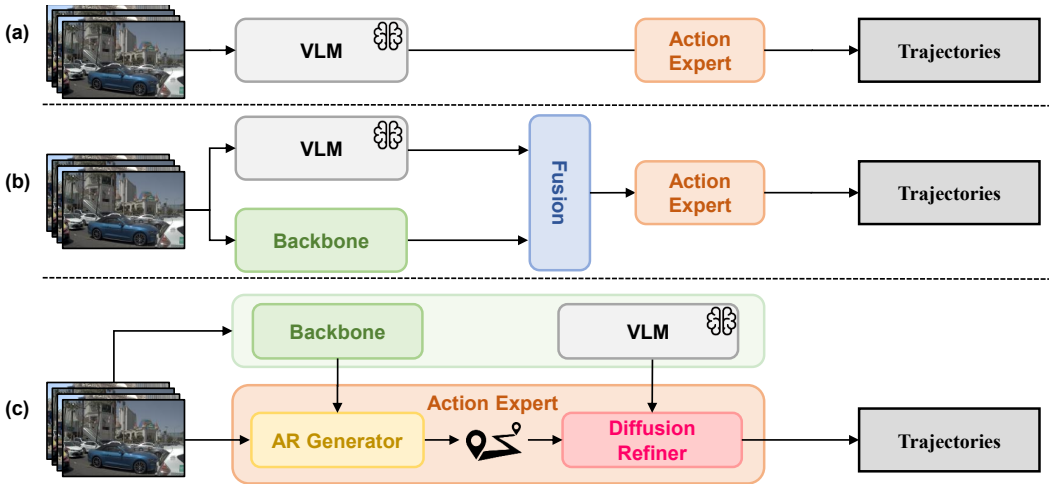


Figure 1: Comparison of different paradigms for integrating VLM into end-to-end autonomous driving. (a) VLM-guided pipeline that predicts high-level guidance to steer an end-to-end model, which introduces an information bottleneck and limits fine-grained trajectory refinement. (b) Feature-level fusion that combines VLM and perception backbones via a fusion module followed by an action expert, but lacks a principled mechanism to enforce consistency between local dynamics and global trajectory structure. (c) Ours (ChainFlow-VLA) formulates trajectory prediction as a unified causal–flow process, where an AR generator produces temporally consistent proposals that are refined by a diffusion model in the residual space. Fine-tuned VLM representations are injected as semantic flow conditioning to guide global trajectory refinement, enabling tight coupling between causal reasoning, global optimization, and high-level semantics.

Recent studies have attempted to incorporate VLM to enhance the semantic understanding and reasoning capabilities of end-to-end autonomous driving systems. As illustrated in Figure 1(a), one category of methods Fu et al. [2025], Zhou et al. [2026a] utilizes a VLM to predict high-level features, which are subsequently processed by a downstream action expert model to generate the final trajectory. Although intuitive, this paradigm compresses rich scene semantics into discrete signals, limiting fine-grained trajectory optimization. Another category of methods Xie et al. [2026], Li et al. [2025c, 2026], shown in Figure 1(b), attempts to fuse VLM features with existing end-to-end driving representations to obtain more robust features, which are then decoded into trajectories via a unified action expert model. However, treating semantic reasoning and physical trajectory generation as loosely coupled components makes it difficult for semantic information to exert a direct impact during the planning stage, where error correction is most critical.

Our analysis reveals that existing methods conflate two fundamental yet insufficiently addressed questions. First, most action experts generate trajectories from high-dimensional features using a single autoregressive or diffusion paradigm Zhang et al. [2026]. Although some works Yang et al. [2025], Li et al. [2025b] combine these approaches, they struggle to maintain consistency between local dynamics and global trajectory structure (see Section 3). Second, existing methods typically integrate VLM at early feature fusion stages Jiang et al. [2025], Fu et al. [2025], assuming semantic information should be injected as early as possible. However, strong end-to-end models already exhibit robust trajectory generation capabilities Chen et al. [2024a]. The main challenge lies not in generating trajectories from scratch, but in refining them in long-tail scenarios Hallgarten et al. [2024] to satisfy semantic constraints. We therefore argue that VLM should not serve as a direct trajectory generator, but rather as a provider of semantic constraints at critical stages of refinement.

Based on these insights, we propose ChainFlow-VLA (Figure 1(c)), which models trajectory generation as a unified causal generation–global refinement process rather than loosely coupled modules. An autoregressive model first generates trajectory modes, capturing temporal causal structure. Conditioned on these priors, a diffusion refiner guided by VLM representations performs residual refinement. This reformulation shifts trajectory modeling from absolute generation to semantic correction, focusing on how prior trajectories should be adjusted under environmental context. It mitigates error accumulation and local optima in autoregressive decoding while preserving global consistency.

Despite its simplicity, this design aligns closely with the structure of end-to-end driving. On the NAVSIM v1 benchmark Dauner et al. [2024], ChainFlow-VLA achieves a score of 94.80, surpassing prior methods and reaching human-level performance. These results highlight the importance of unifying causal modeling, global optimization, and semantic reasoning for robust autonomous driving.

We summarize our contributions as follows:

- We propose ChainFlow-VLA, a unified framework that casts trajectory generation as a probabilistic mixture over AR-induced modes, decomposed into a causal autoregressive Chain and a residual diffusion Flow, unifying temporal reasoning and global geometric consistency within a single formulation.
- We reformulate VLM guidance as mode-conditioned semantic control over residual refinement, where VLM representations are injected to modulate local trajectory corrections rather than global trajectory generation.
- Experiments on NAVSIM v1, NAVSIM v2, and HUGSIM evaluate ChainFlow-VLA across non-reactive simulation, pseudo-simulation, and closed-loop settings. On NAVSIM v1, it achieves state-of-the-art performance and reaches human-level benchmark scores.

2 Related Work

End-to-end Autonomous Driving.

End-to-end autonomous driving learns a direct mapping from sensor inputs to future trajectories or control commands Chen et al. [2024a]. Discriminative planners, such as UniAD Hu et al. [2023] and VAD Jiang et al. [2023], integrate perception and planning efficiently, but their deterministic regression paradigm limits behavioral diversity. Autoregressive planners Jia et al. [2024] capture temporal causality through sequential prediction, yet may suffer from error accumulation and weak global optimization. Diffusion-based methods, such as DiffusionDrive Liao et al. [2025], improve multimodal generation via iterative denoising, but can struggle with stable and physically consistent long-horizon planning. These limitations motivate our Chain–Flow design, which combines causal AR proposal generation with diffusion-based global refinement.

Vision-Language-Action Models for Driving.

Vision-language models have been introduced into autonomous driving for their semantic understanding and reasoning ability Sima et al. [2023], Hwang et al. [2024]. Direct VLA planners Zhou et al. [2026a], Wang et al. [2025], Zhou et al. [2025] map visual-language representations to driving actions or trajectories, but continuous trajectory generation remains challenging for VLMs due to limited fine-grained spatial precision. Other methods use VLMs as high-level reasoners or feature providers Zeng et al. [2025], Li et al. [2025c], Fu et al. [2025], Huang et al. [2026], where semantic information is often injected before the final planning refinement stage. In contrast, ChainFlow-VLA uses VLM hidden states to guide residual diffusion over AR proposals, allowing semantic reasoning to directly modulate trajectory correction.

3 Preliminaries

Task Definition. We consider end-to-end trajectory planning as modeling a multi-modal conditional distribution:

$$P(Y \mid \mathcal{O}), \tag{1}$$

where $\mathcal{O}$ denotes observations from multiple modalities and $Y = \{y_t\}_{t=1}^{T}$ is the future trajectory.

From Global Distribution to Conditional Decomposition. The trajectory distribution $P(Y \mid \mathcal{O})$ is inherently multi-modal, making direct modeling challenging due to its highly entangled structure.

Autoregressive and diffusion-based models provide complementary inductive biases. AR models provide a causal factorization of the trajectory distribution:

$$P(Y_{\mathrm{AR}} \mid \mathcal{O}) = \prod_{t} P(y_t \mid y_{<t}, \mathcal{O}), \tag{2}$$

whereas diffusion models capture global structure via iterative denoising.

We bridge these two paradigms through a conditional decomposition. An AR model produces a set of trajectory proposals $\{Y_{\mathrm{AR}}^{(k)}\}_{k=1}^{K}$, where each $Y_{\mathrm{AR}}^{(k)}$ denotes the k-th trajectory mode. Conditioned on each proposal, the problem reduces to modeling a local conditional distribution:

$$P(Y \mid Y_{\mathrm{AR}}^{(k)}, \mathcal{O}), \tag{3}$$

We parameterize this conditional distribution using a representation h_{VLM} extracted from a vision-language model, which encodes semantic context from observations $\mathcal{O}$:

$$P(Y \mid Y_{\mathrm{AR}}^{(k)}, \mathcal{O}) \;\approx\; P(Y \mid Y_{\mathrm{AR}}^{(k)}, h_{\mathrm{VLM}}), \tag{4}$$

where h_{VLM} modulates the local conditional distribution for each trajectory mode.

This yields an implicit mixture formulation, inspired by the law of total probability:

$$P(Y \mid \mathcal{O}) \approx \sum_{k=1}^{K} P(Y \mid Y_{\mathrm{AR}}^{(k)}, h_{\mathrm{VLM}}) \cdot P(Y_{\mathrm{AR}}^{(k)} \mid \mathcal{O}), \tag{5}$$

where each component corresponds to a local distribution centered around a trajectory mode, which is later instantiated in residual space for efficient learning.

4 Methods

4.1 Overview

Building on the above formulation, the ChainFlow-VLA framework (Figure 2) realizes planning as a two-stage sequential refinement process that instantiates the factorization in Eq. (5). Initially, the Autoregressive Trajectory Generation module processes driving features to produce a set of K trajectory proposals, ensuring causal consistency and physical feasibility. These proposals serve as initial modes for the subsequent VLM-Guided Residual Diffusion stage. In this stage, a Diffusion Transformer (DiT) models the residual distribution conditioned on VLM hidden states, providing fine-grained semantic guidance to refine the AR-induced modes into final trajectories.

4.2 Chain: Autoregressive Trajectory Generation

As illustrated in Figure 2, the autoregressive module takes BEV-style driving features and learnable trajectory queries as inputs, and iteratively generates future states through a recurrent decoding process.

We follow the autoregressive factorization introduced in section 3, where trajectory generation is modeled as a sequential conditional process:

$$p(y_t \mid y_{<t}, \mathcal{O}), \tag{6}$$

which introduces a strong causal inductive bias, ensuring temporally consistent and physically plausible rollouts.

In practice, each conditional term is implicitly parameterized by a deterministic predictor.

To capture multi-modality, we maintain a set of K parallel trajectory hypotheses. Each trajectory $Y_{\mathrm{AR}}^{(k)} = \{y_t^{(k)}\}_{t=1}^{T}$ represents a distinct kinematic mode, yielding a discrete approximation of the global trajectory distribution.

At each step t, the model predicts control variables $(a_t^{(k)}, \omega_t^{(k)})$ conditioned on the previous state and scene context:

$$(a_t^{(k)}, \omega_t^{(k)}) = H_\theta(y_{<t}^{(k)}, \mathcal{O}), \tag{7}$$

where H_θ denotes a learnable predictor that parameterizes the conditional prediction of control variables. The next state is obtained through a kinematic transition:

$$y_t^{(k)} = \mathrm{Bicycle}(y_{t-1}^{(k)}, a_t^{(k)}, \omega_t^{(k)}), \tag{8}$$

where $\mathrm{Bicycle}(\cdot)$ denotes a standard bicycle kinematic model, which enforces physical feasibility and stabilizes long-horizon prediction.

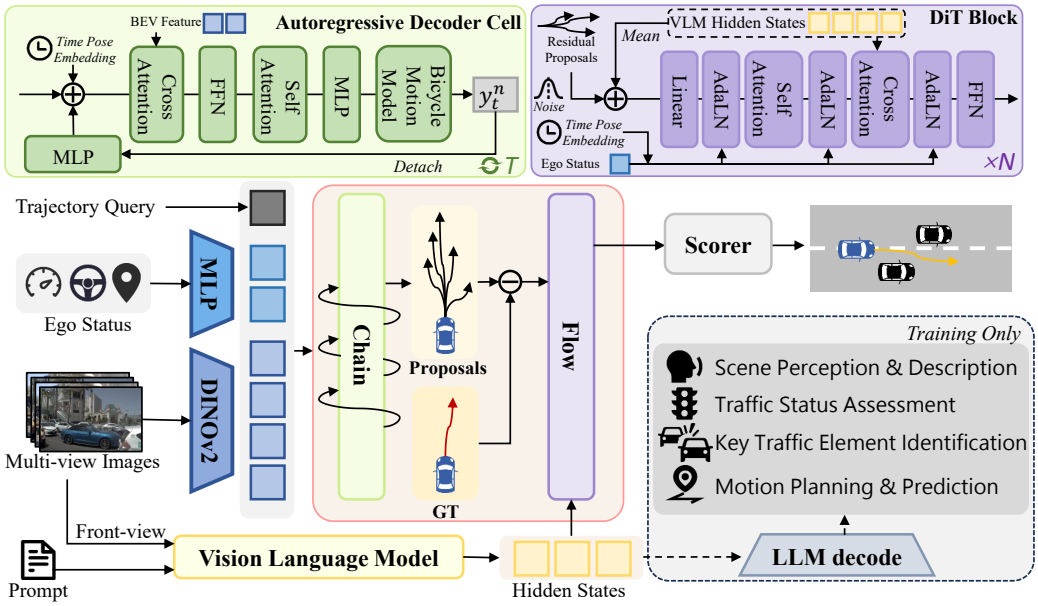


Figure 2: ChainFlow-VLA framework. The model first performs Autoregressive Trajectory Generation (Chain) to produce K causal proposals, which are then refined via VLM-Guided Residual Diffusion (Flow). By learning the residuals between AR proposals and ground-truth trajectories, the model unifies causal rollout with VLM-based semantic guidance, formulating planning as a mixture of VLM-conditioned residual distributions over AR-induced modes.

Scene observations are encoded into latent tokens and queried at each step to provide environment-aware context, while the autoregressive hidden state propagates motion intent over time.

After T steps, the model produces a set of trajectory proposals:

$$Y_{\mathrm{AR}} = \{Y_{\mathrm{AR}}^{(k)}\}_{k=1}^{K}. \tag{9}$$

From a modeling perspective, this stage performs a causal discretization of the global trajectory distribution, providing structured initialization for subsequent flow-based refinement.

4.3 Flow: VLM-Guided Residual Diffusion

Following Eq. (4), the Flow module instantiates the local conditional term $P(Y \mid Y_{\mathrm{AR}}^{(k)}, h_{\mathrm{VLM}})$ in Eq. (5). Rather than modeling the full trajectory distribution in the global space, we refine each AR proposal in a local residual space Zheng et al. [2025]. This turns trajectory generation into proposal-centered correction guided by VLM.

AR-Conditioned Residual Modes. We leverage the trajectory proposals generated by the preceding AR module as mode-specific proposals for residual refinement. The Flow module does not re-estimate the AR proposal distribution $P(Y_{\mathrm{AR}}^{(k)} \mid \mathcal{O})$ in Eq. (5). Instead, for AR-conditioned modes, it learns a residual distribution that corrects the proposal toward the expert trajectory. The refined trajectory is represented as:

$$Y = Y_{\mathrm{AR}}^{(k)} + \Delta Y_k, \tag{10}$$

where ΔY_k denotes the correction relative to the k-th AR proposal. Accordingly, the local conditional distribution is instantiated in residual space:

$$P\left(Y \mid Y_{\mathrm{AR}}^{(k)}, h_{\mathrm{VLM}}\right) = P\left(\Delta Y_k \mid Y_{\mathrm{AR}}^{(k)}, h_{\mathrm{VLM}}\right). \tag{11}$$

This converts each local component in Eq. (5) into a proposal-conditioned residual refinement problem.

VLM-Guided Conditional Diffusion. Residual refinement requires determining how each AR proposal should be corrected under the current scene. Such correction depends not only on geometric

deviation, but also on semantic understanding, such as route intention, traffic context, and trajectory-level feasibility. Following ReCogDrive Li et al. [2025c], we adopt a driving-oriented VLM supervised fine-tuned on environment-understanding and trajectory-QA tasks. Without further optimizing the VLM under the diffusion objective, we directly use its hidden states h_{VLM} as semantic conditions for the residual diffusion model. This enables the Flow module to transfer the VLM's general driving knowledge into proposal correction without task-specific VLM adaptation.

Formally, given the expert trajectory Y^* and the k-th AR proposal, the residual target is

$$\Delta Y_k^* = Y^* - Y_{\mathrm{AR}}^{(k)}. \tag{12}$$

We then construct noisy residual samples by

$$\mathbf{z}_t^{(k)} = \sqrt{\bar{\alpha}_t}\Delta Y_k^* + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \tag{13}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The diffusion model predicts the injected noise conditioned on the timestep t, ego state c_{ego}, VLM hidden states h_{VLM}, and the AR proposal $Y_{\mathrm{AR}}^{(k)}$:

$$\hat{\boldsymbol{\epsilon}}^{(k)} = \boldsymbol{\epsilon}_\theta\left(\mathbf{z}_t^{(k)}, t, c_{\mathrm{ego}}, h_{\mathrm{VLM}}, Y_{\mathrm{AR}}^{(k)}\right). \tag{14}$$

Architecture and Inference. As shown in the DiT block of Figure 2, our residual refiner follows the general architecture of DiT Peebles and Xie [2023]. Noisy residual tokens are processed by stacked transformer blocks, where conditions are injected through adaptive LayerNorm. In addition, full VLM hidden states are incorporated via cross-attention, allowing high-level semantic information to guide the residual denoising process.

At inference time, we sample a residual $\hat{\Delta Y}_k$ for each AR proposal using the DDIM process and reconstruct the refined trajectory as

$$\hat{Y}_k = Y_{\mathrm{AR}}^{(k)} + \hat{\Delta Y}_k. \tag{15}$$

Overall, the Flow module implements each local term in Eq. (5) as VLM-guided residual refinement around an AR proposal.

4.4 Scorer

We employ a scoring head Guo et al. [2025] to evaluate each candidate trajectory, producing a set of utility scores. The scorer acts as a proxy utility function, defining a decision rule over the learned trajectory distribution. The final trajectory is selected by aggregating these scores and choosing the highest-scoring candidate.

4.5 Training Objectives and Target Assignment

ChainFlow-VLA is trained in two stages, both leveraging trajectory and scorer supervision, with Stage II further introducing diffusion-based refinement.

Stage I. We train the AR module using WTA-based supervision, following Kirby et al. [2026]:

$$\mathcal{L}_{\mathrm{stage1}} = \mathcal{L}_{\mathrm{traj}} + \lambda_1 \mathcal{L}_{\mathrm{scorer}}, \tag{16}$$

where the trajectory loss selects the closest mode to the expert trajectory.

Stage II. We jointly fine-tune the Stage I planning modules together with the diffusion refiner and scorer, while keeping the VLM frozen:

$$\mathcal{L}_{\mathrm{stage2}} = \lambda_2 \mathcal{L}_{\mathrm{diff}} + \lambda_3 \mathcal{L}_{\mathrm{traj}} + \lambda_4 \mathcal{L}_{\mathrm{scorer}}. \tag{17}$$

We adopt an asymmetric WTA assignment in Stage II. For diffusion supervision, the expert trajectory is matched to the closest AR proposal:

$$k^* = \arg\min_k \left\| Y_{\mathrm{AR}}^{(k)} - Y^* \right\|_2. \tag{18}$$

The diffusion objective is then computed within this selected AR-conditioned mode:

$$\mathcal{L}_{\mathrm{diff}} = \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \right\|_2^2. \tag{19}$$

Table 1: Comparison of results on NAVSIM v1. All metrics are higher-is-better. Best results are highlighted in bold. The 94.8 PDMS result of ChainFlow-VLA on trainval uses 12 denoising steps.

Methods	Venue	PDMS↑	NC↑	DAC↑	EP↑	TTC↑	Comf.↑
End-to-End Methods							
UniAD Hu et al. [2023]	CVPR'23	83.4	97.8	91.9	78.8	92.9	**100.0**
VADv2 Chen et al. [2024b]	arXiv'24	80.9	97.2	89.1	76.0	91.6	**100.0**
iPad Guo et al. [2025]	arXiv'25	91.7	98.6	98.3	88.0	94.9	**100.0**
DiffusionDrive Liao et al. [2025]	CVPR'25	88.1	98.2	96.2	82.2	94.7	**100.0**
Hydra-MDP++ Li et al. [2025a]	arXiv'25	91.0	98.6	98.6	85.7	95.1	**100.0**
Centaur Sima et al. [2025]	arXiv'25	92.6	99.5	98.9	85.9	98.0	**100.0**
TrajDiff$_{(train)}$ Gui et al. [2025]	arXiv'25	86.4	98.0	95.0	80.8	93.7	**100.0**
TrajDiff$_{(trainval)}$	arXiv'25	88.5	98.1	97.0	82.7	94.3	**100.0**
DriveSuprim Yao et al. [2026]	AAAI'26	93.5	98.6	98.6	91.3	95.5	**100.0**
RAP-DINO† Feng et al. [2025]	ICLR'26	93.8	99.1	98.9	90.3	96.7	**100.0**
DrivoR$_{(train)}$ Kirby et al. [2026]	CVPR'26	93.1	98.9	98.3	89.1	96.2	**100.0**
DrivoR$_{(trainval)}$	CVPR'26	93.7	99.0	98.9	90.0	96.7	**100.0**
VLA-based Methods							
UniVLA Wang et al. [2025]	RSS'25	81.7	96.9	91.1	76.8	91.7	96.7
AutoVLA Zhou et al. [2025]	NeurIPS'25	89.1	98.4	95.6	81.9	98.0	99.9
FSDrive Zeng et al. [2025]	NeurIPS'25	85.1	98.2	93.8	80.1	93.3	99.9
DriveVLA-W0 Li et al. [2025b]	ICLR'26	90.2	98.7	99.1	83.3	95.3	99.3
ReCogDrive Li et al. [2025c]	ICLR'26	90.8	97.9	97.3	87.3	94.9	**100.0**
SGDrive Li et al. [2026]	CVPR'26	91.1	98.6	97.8	85.8	96.2	**100.0**
SpanVLA Zhou et al. [2026b]	arXiv'26	90.3	99.1	97.1	86.3	95.2	**100.0**
LatentVLA Xie et al. [2026]	arXiv'26	92.4	98.9	98.2	88.2	96.0	**100.0**
Baselines & Oracle							
Human Driver Dauner et al. [2024]	NeurIPS'24	94.8	100.0	100.0	87.5	100.0	99.9
ChainFlow-VLA$_{(train)}$	–	93.6	98.8	98.6	90.8	96.1	**100.0**
ChainFlow-VLA$_{(trainval)}$	–	**94.8**	**99.2**	**99.0**	**91.9**	**97.2**	99.9

† *Note*: RAP-DINO is pre-trained on a private dataset that is 10× larger than the default *navtrain* set.

This design separates mode selection from residual refinement, enabling the diffusion objective to focus on local correction around AR proposals. Meanwhile, trajectory supervision $\mathcal{L}_{\text{traj}}$ is applied to the refined outputs. This output-level supervision provides a direct optimization signal in trajectory space, accelerating convergence and stabilizing refinement training.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate our method on NAVSIM v1 Dauner et al. [2024], a large-scale benchmark for vision-based autonomous driving that combines real-world driving data with a non-reactive simulation protocol for scalable evaluation. To provide a comprehensive assessment, we compare two training configurations: one trained on the navtrain split and the other on the combined trainval split. This setup allows us to analyze the impact of training data scale on planning performance. We additionally evaluate on NAVSIM v2 Cao et al. [2025] and HUGSIM Zhou et al. [2024], using navtrain alone or navtrain augmented with 65K SimScale samples Tian et al. [2026]. Appendix A describes the metrics and the 436-scenario HUGSIM evaluation setup.

Implementation Details. We train our model on 8 NVIDIA A800 GPUs through a two-stage pipeline. In Stage 1, we fine-tune the image encoder with LoRA and train the Chain module for 25 epochs. In Stage 2, we jointly fine-tune the Stage 1 modules, Flow module, and scorer for 40 epochs; the VLM remains fully frozen. Following ReCogDrive Li et al. [2025c], we use the 2B VLM fine-tuned from InternVL as the driving-oriented VLM. Under this configuration, the two training stages take approximately two days in total. Throughout both stages, we employ the AdamW optimizer Loshchilov and Hutter [2017] with a per-GPU batch size of 6. The base learning rate is set to

2×10^{-4} and scaled according to $\sqrt{B/64}$, using a linear warmup for the first 10% of steps followed by a cosine decay schedule. The loss weights $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are set to 1, 10, 20, and 4, respectively. During inference, we use 4 denoising steps by default; the best NAVSIM v1 result of 94.80 (reported as 94.8 in the main comparison) uses 12 steps.

5.2 Main Results

As shown in Table 1, ChainFlow-VLA achieves a new state-of-the-art PDMS of 94.8 using 12 denoising steps, significantly outperforming prior end-to-end (93.8) and VLA-based (92.4) models. **Remarkably, our approach reaches human-level performance, matching expert trajectory scores on the benchmark**. Compared to feature-fusion methods like LatentVLA, which merge VLM and perception features, our architecture uses VLM semantics specifically for flow-based residual refinement. This suggests that simple high-level fusion is insufficient. These results validate our causal-to-global paradigm as an effective bridge between semantic reasoning and geometric precision.

Table 2: NAVSIM v2 results; all metrics are higher-is-better. Stage 1 and Stage 2 denote the initial and follow-up simulations, respectively; EPDMS is the overall score. Training data are indicated in parentheses; Ours denotes ChainFlow-VLA.

Method	Stage 1									Stage 2									EPDMS
	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	
DrivoR *(navtrain)*	98.8	95.1	98.9	100	72.6	98.7	94.0	97.6	73.3	90.2	88.4	91.9	98.6	70.0	88.0	50.1	98.5	76.2	48.3
Ours *(navtrain)*	99.1	97.1	99.4	100	74.0	99.1	91.7	97.6	52.8	92.7	92.5	95.5	99.5	70.7	90.4	58.3	97.6	42.3	**50.4**
DrivoR *(navtrain + 65K SimScale)*	98.9	97.3	99.2	99.6	77.7	99.1	95.3	97.6	68.4	92.3	92.2	97.0	99.0	72.1	90.3	56.3	97.1	38.8	52.3
Ours *(navtrain + 65K SimScale)*	98.9	97.6	99.4	99.6	78.5	98.7	94.7	97.6	71.6	92.7	91.0	96.2	99.2	76.1	89.5	54.6	97.0	40.6	**55.2**

Table 3: Closed-loop HUGSIM results on 436 scenarios. RC and HD-Score are multiplied by 100 and rounded to one decimal place; higher is better. E/M/H/X denote Easy/Medium/Hard/Extreme. Bold indicates the better result within each training-data setting.

Method	Route Completion (RC)↑					HD-Score↑				
	E	M	H	X	Avg.	E	M	H	X	Avg.
DrivoR *(navtrain)*	84.6	40.5	29.0	**29.8**	43.5	75.3	25.1	13.7	**15.0**	29.4
ChainFlow-VLA *(navtrain)*	**88.0**	**52.1**	**34.5**	28.9	**49.3**	**79.4**	**36.4**	**20.1**	14.8	**35.6**
DrivoR *(navtrain + 65K SimScale)*	78.8	53.7	28.0	27.1	46.4	73.7	50.0	**18.7**	14.1	39.0
ChainFlow-VLA *(navtrain + 65K SimScale)*	**86.5**	**55.1**	**29.0**	**27.8**	**48.7**	**80.5**	**50.2**	18.2	**15.2**	**40.5**

Tables 2 and 3 extend the evaluation beyond NAVSIM v1. ChainFlow-VLA improves EPDMS over DrivoR by 2.1 and 2.9 points, respectively, without and with SimScale augmentation. On HUGSIM, the corresponding average HD-Scores increase from 29.4 to 35.6 and from 39.0 to 40.5.

5.3 Qualitative Results

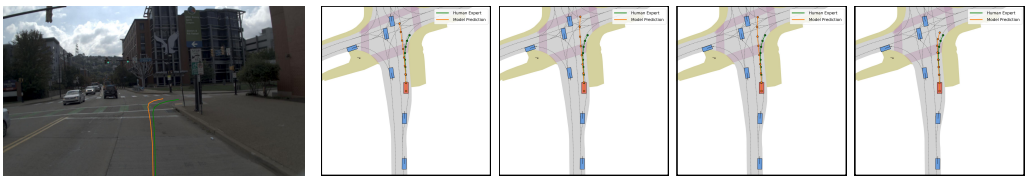


Figure 3: Visualization of trajectory predictions on representative NAVSIM scenarios. The predicted trajectory is shown in orange, while the human expert trajectory is shown in green.

We compare ChainFlow-VLA with DrivoR and the three stages of the ablation pipeline on NAVSIM test scenarios (Figure 3). Stage 1 produces safer, more consistent global trajectories, while Stage 2 benefits from VLM features that provide richer semantic cues than BEV features and improve road-centerline tracking. These qualitative observations complement the component analysis in Table 4. Additional comparisons, including challenging interaction cases, are provided in Appendices D and E.

5.4 Ablation Study

We conduct ablations to evaluate the contribution of each component in ChainFlow-VLA, including the AR trajectory generator, the residual DiT refiner, VLM guidance, and several key design choices.

Table 4: Main component ablation. ID 0 is the DrivoR baseline. Without VLM guidance, the DiT refiner uses the default DrivoR scene tokens as conditioning. ID 3 reports the best 12-step result; the default 4-step full model obtains 94.72 PDMS.

ID	**AR Gen.** (Chain)	**DiT Refiner** (Flow)	**VLM Guidance** (VLA)	**Metrics** PDMS↑	NC↑	DAC↑	EP↑	TTC↑	Comf.↑
0	✗	✗	✗	93.7	99.0	98.9	90.0	96.7	**100.0**
1	✓	✗	✗	94.0 (↑ 0.3)	99.1	98.9	90.8	96.7	99.9
2	✓	✓	✗	94.1 (↑ 0.4)	99.1	98.9	91.0	96.7	99.9
3	✓	✓	✓	**94.8 (↑ 1.1)**	**99.2**	**99.0**	**91.9**	**97.2**	99.9

Component analysis. Table 4 summarizes the component configurations. Without VLM guidance, the DiT refiner uses the default DrivoR scene tokens as conditioning. Appendix D provides corresponding qualitative comparisons. The AR generator improves DrivoR from 93.7 to 94.0 PDMS, and the residual DiT refiner further improves the score to 94.1. With VLM hidden-state guidance, the full model achieves 94.72 PDMS at 4 steps and 94.8 at 12 steps. The largest gain comes from EP, increasing from 90.0 to 91.9, indicating significantly improved efficiency while maintaining strong safety performance through enhanced environment understanding.

Table 5: Ablation on DiT design choices under the trainval split.

Modeling Target	PDMS↑
Trajectory space	92.89
Residual space	**94.72**

(a) DiT trajectory formulation

DiT Blocks	PDMS↑
8	94.64
12	**94.72**

(b) Number of DiT blocks

VLM Guidance Source	PDMS↑
Action QA	94.11
Env. & Traj. QA	**94.72**

(c) VLM guidance source

DiT design choices. Table 5 validates several refiner designs under the default 4-step denoising setting. Residual-space modeling outperforms direct trajectory-space prediction, confirming the benefit of refining AR proposals instead of generating trajectories from scratch. Increasing the DiT depth from 8 to 12 blocks brings a modest gain. For VLM guidance, environment- and trajectory-level QA SFT provides more useful hidden states than action-only QA, suggesting that scene and trajectory reasoning better supports residual refinement.

Table 6: Ablation on denoising steps under the trainval split.

N_{step}	2	4	8	12	16
PDMS↑	94.68	94.72	94.74	**94.80**	94.67

Denoising steps. Table 6 studies the number of denoising steps at inference. Increasing N_{step} from 2 to 12 improves PDMS from 94.68 to **94.80**, where the 12-step result already reached the same PDMS obtained by evaluating the human trajectory. However, we use $N_{\text{step}} = 4$ as the default setting to balance performance and inference efficiency.

Table 7: Generalization of ChainFlow across different end-to-end planning paradigms.

Method	Modes	PDMS	NC	DAC	EP	TTC	Comf.
DiffusionDrive	20	88.1	98.2	96.2	82.2	94.7	**100.0**
+ ChainFlow	6	**88.9** (↑ 0.8)	**98.3**	**96.8**	**83.0**	**95.0**	99.9
iPad	64	91.7	98.6	98.3	88.0	94.9	**100.0**
+ ChainFlow	64	**92.7** (↑ 1.0)	**99.0**	**98.7**	**88.4**	**96.1**	99.9

Generalization of ChainFlow. Table 7 evaluates the generalization of ChainFlow across different end-to-end planning paradigms. We integrate ChainFlow into multiple backbones without VLM features, and conduct all experiments on the navtrain set for fair comparison. On a diffusion-based planner (DiffusionDrive Liao et al. [2025]), replacing clustering-based anchors with our ChainFlow module improves performance from 88.1 to 88.9 using only 6 modes (vs. 20 originally). A similar trend is observed on a score-based planner (iPad Guo et al. [2025]), where ChainFlow improves performance from 91.7 to 92.7. These consistent improvements across heterogeneous backbones demonstrate that ChainFlow serves as a general and effective action expert.

Inference-cost measurements, additional VLM feature controls, and qualitative comparisons are provided in Appendices B, C, and E.

6 Conclusion

We introduced ChainFlow-VLA, a unified vision-language-action framework that casts trajectory planning as a Chain-to-Flow formulation. By decomposing planning into a causal autoregressive Chain and a residual diffusion Flow, our approach unifies temporal reasoning and global geometric consistency within a single probabilistic framework. A central finding of this work is that vision-language models are more effective as semantic conditioners for trajectory refinement rather than direct generators. By leveraging VLM hidden states to guide residual diffusion, we transform planning from global trajectory synthesis into mode-conditioned semantic correction, significantly improving robustness in long-tail scenarios. Extensive experiments on NAVSIM v1 demonstrate that ChainFlow-VLA achieves state-of-the-art performance and reaches human-level driving quality. We hope this work provides a step toward more principled integration of causal reasoning, generative refinement, and semantic understanding in autonomous driving.

Limitations. Although the current VLM guidance improves residual refinement, it is still based on a general driving-oriented VLM trained with environment-understanding and trajectory-QA supervision. Since the Flow module essentially performs trajectory refinement rather than action generation, a score-oriented or judge-oriented VLM with stronger trajectory evaluation ability may be better aligned with this task. Designing such refinement-aware VLM guidance is an important direction for future work.

Acknowledgments and Disclosure of Funding

Funding. Chongqing Afari Intelligent Drive Co., Ltd. provided computational resources for this work.

Competing interests. The authors declare no additional competing interests beyond their disclosed affiliations.

References

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

Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10164–10183, 2024a. doi: 10.1109/TPAMI.2024.3435937.

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

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

Lan Feng, Yang Gao, Eloi Zablocki, Quanyi Li, Wuyang Li, Sichao Liu, Matthieu Cord, and Alexandre Alahi. Rap: 3d rasterization augmented end-to-end planning. *arXiv preprint arXiv:2510.04333*, 2025.

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. Orion: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24823–24834, 2025.

Xingtai Gui, Jianbo Zhao, Wencheng Han, Jikai Wang, Jiahao Gong, Feiyang Tan, Cheng-zhong Xu, and Jianbing Shen. Trajdiff: End-to-end autonomous driving without perception annotation. *arXiv preprint arXiv:2512.00723*, 2025.

Xingtai Gui, Meijie Zhang, Tianyi Yan, Wencheng Han, Jiahao Gong, Feiyang Tan, Cheng-zhong Xu, and Jianbing Shen. Bridging scene generation and planning: Driving with world model via unifying vision and motion representation. *arXiv preprint arXiv:2603.14948*, 2026.

Ke Guo, Haochen Liu, Xiaojun Wu, Jia Pan, and Chen Lv. ipad: Iterative proposal-centric end-to-end autonomous driving, 2025. URL `https://arxiv.org/abs/2505.15111`.

Marcel Hallgarten, Julian Zapata, Martin Stoll, Katrin Renz, and Andreas Zell. Can vehicle motion planning generalize to realistic long-tail scenarios? In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5388–5395. IEEE, 2024.

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.

Minqing Huang, Yujiao Xiang, Zihan Liang, Jiajie Huang, Jingqi Wang, Zhi Xu, Feiyang Tan, Hangning Zhou, Mu Yang, and Gong Che. Coworld-vla: Thinking in a multi-expert world model for autonomous driving. *arXiv preprint arXiv:2605.10426*, 2026.

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.

Xiaosong Jia, Shaoshuai Shi, Zijun Chen, Li Jiang, Wenlong Liao, Tao He, and Junchi Yan. Amp: Autoregressive motion prediction revisited with next token prediction for autonomous driving. *arXiv preprint arXiv:2403.13331*, 2024.

Anqing Jiang, Yu Gao, Zhigang Sun, Yiru Wang, Jijun Wang, Jinghao Chai, Qian Cao, Yuweng Heng, Hao Jiang, Yunda Dong, et al. Diffvla: Vision-language guided diffusion planning for autonomous driving. *arXiv preprint arXiv:2505.19381*, 2025.

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.

Ellington Kirby, Alexandre Boulch, Yihong Xu, Yuan Yin, Gilles Puy, Éloi Zablocki, Andrei Bursuc, Spyros Gidaris, Renaud Marlet, Florent Bartoccioni, et al. Driving on registers. *arXiv preprint arXiv:2601.05083*, 2026.

Jingyu Li, Junjie Wu, Dongnan Hu, Xiangkai Huang, Bin Sun, Zhihui Hao, Xianpeng Lang, Xiatian Zhu, and Li Zhang. Sgdrive: Scene-to-goal hierarchical world cognition for autonomous driving. *arXiv preprint arXiv:2601.05640*, 2026.

Kailin Li, Zhenxin Li, Shiyi Lan, Yuan Xie, Zhizhong Zhang, Jiayi Liu, Zuxuan Wu, Zhiding Yu, and Jose M Alvarez. Hydra-mdp++: Advancing end-to-end driving via expert-guided hydra-distillation. *arXiv preprint arXiv:2503.12820*, 2025a.

Yiming Li, Sihang Li, Xinhao Liu, Moonjun Gong, Kenan Li, Nuo Chen, Zijun Wang, Zhiheng Li, Tao Jiang, Fisher Yu, et al. Sscbench: A large-scale 3d semantic scene completion benchmark for autonomous driving. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13333–13340. IEEE, 2024.

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*, 2025b.

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*, 2025c.

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.

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

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. *arXiv preprint arXiv:2312.14150*, 2023.

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

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

Haochen Tian, Tianyu Li, Haochen Liu, Jiazhi Yang, Yihang Qiu, Guang Li, Junli Wang, Yinfeng Gao, Zhang Zhang, Liang Wang, Hangjun Ye, Long Chen, and Hongyang Li. SimScale: Learning to drive via real-world simulation at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. *arXiv preprint arXiv:2506.19850*, 2025.

Chengen Xie, Bin Sun, Tianyu Li, Junjie Wu, Zhihui Hao, XianPeng Lang, and Hongyang Li. Latentvla: Efficient vision-language models for autonomous driving via latent action prediction. *arXiv preprint arXiv:2601.05611*, 2026.

Zhenjie Yang, Yilin Chai, Xiaosong Jia, Qifeng Li, Yuqian Shao, Xuekai Zhu, Haisheng Su, and Junchi Yan. Drivemoe: Mixture-of-experts for vision-language-action model in end-to-end autonomous driving. *arXiv preprint arXiv:2505.16278*, 2025.

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

Shuang Zeng, Xinyuan Chang, Mengwei Xie, Xinran Liu, Yifan Bai, Zheng Pan, Mu Xu, Xing Wei, and Ning Guo. Futuresightdrive: Thinking visually with spatio-temporal cot for autonomous driving. *arXiv preprint arXiv:2505.17685*, 2025.

Yiwei Zhang, Xuesong Chen, Jin Gao, Hanshi Wang, Fudong Ge, Weiming Hu, Shaoshuai Shi, and Zhipeng Zhang. Onedrive: Unified multi-paradigm driving with vision-language-action models. *arXiv preprint arXiv:2604.17915*, 2026.

Zhiyu Zheng, Shaoyu Chen, Haoran Yin, Xinbang Zhang, Jialv Zou, Xinggang Wang, Qian Zhang, and Lefei Zhang. Resad: Normalized residual trajectory modeling for end-to-end autonomous driving. *arXiv preprint arXiv:2510.08562*, 2025.

Hongyu Zhou, Longzhong Lin, Jiabao Wang, Yichong Lu, Dongfeng Bai, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. HUGSIM: A real-time, photo-realistic and closed-loop simulator for autonomous driving. *arXiv preprint arXiv:2412.01718*, 2024.

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

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.

Zewei Zhou, Ruining Yang, Yiluan Guo, Sherry X Chen, Tao Feng, Kateryna Pistunova, Yishan Shen, Lili Su, Jiaqi Ma, et al. Spanvla: Efficient action bridging and learning from negative-recovery samples for vision-language-action model. *arXiv preprint arXiv:2604.19710*, 2026b.

A Benchmarks and Evaluation Metrics

NAVSIM v1. NAVSIM v1 Dauner et al. [2024] evaluates planned trajectories through a four-second non-reactive simulation. We report the PDM Score (PDMS) and its component metrics: no at-fault collisions (NC), drivable area compliance (DAC), ego progress (EP), time-to-collision within bound (TTC), and comfort (Comf.). NC and DAC act as multiplicative penalties, while EP, TTC, and comfort contribute to the weighted score. NAVSIM scores in our tables are reported on a 0–100 scale.

NAVSIM v2. NAVSIM v2 Cao et al. [2025] uses two-stage pseudo-simulation and the Extended PDM Score (EPDMS). In addition to NC, DAC, EP, and TTC, it evaluates driving direction compliance (DDC), traffic light compliance (TLC), lane keeping (LK), history comfort (HC), and extended comfort (EC). HC accounts for consistency with the vehicle's motion history; EC assesses consistency between plans at consecutive frames. Follow-up scenes are weighted according to their proximity to the first-stage endpoint, and the stage scores are aggregated into EPDMS. The Stage 1 and Stage 2 component scores in Table 2 refer to the initial and follow-up simulations, respectively.

HUGSIM. HUGSIM Zhou et al. [2024] provides photorealistic closed-loop evaluation. Route Completion (RC) measures progress along the reference route, while HD-Score combines route completion with trajectory-quality criteria including collision avoidance, drivable-area compliance, time to collision, and comfort. We report both metrics on a 0–100 scale (multiplied by 100 and rounded to one decimal place) for the Easy, Medium, Hard, and Extreme subsets and for the full evaluation set. In our reproduction, we apply bug fixes following the official instructions and use the updated downloaded data, comprising 436 scenarios. Both methods are evaluated under this same setup; results from earlier publications using different scenario sets may not be directly comparable.

Asset licenses. The NAVSIM repository and DrivoR code are distributed under Apache-2.0, except where otherwise specified. NAVSIM's underlying datasets, including nuPlan and OpenScene, retain their original distribution licenses. The SimScale code is Apache-2.0, while its released dataset is CC BY-NC-SA 4.0. The released ReCogDrive 2B VLM weights are labeled Apache-2.0. DINOv2 code and model weights are Apache-2.0, and HUGSIM code is MIT-licensed. These repository and model licenses do not replace the terms applicable to underlying third-party datasets. We credit the original sources and use these assets under their respective terms.

B Inference Cost and Model Size

Table 8: Inference-time measurement settings.

Setting	Value
GPU	NVIDIA A800-SXM4-80GB
Batch size	1, single-frame inference
Perception input	Four cameras (front/left/right/back), each 1148×672
VLM input	Front view; eight dynamic patches and one thumbnail, each 448×448
Timing protocol	`torch.cuda.Event`; 10 warm-up runs and 50 measured runs
VLM attention	FlashAttention 2.8.3

Table 9: Inference latency (ms). Component timings and end-to-end totals are measured separately, so totals need not equal the sum of component timings. Dual-GPU execution uses an additional GPU.

Component	ChainFlow-VLA	ReCogDrive	DrivoR
VLM (InternVL-2B)	178.9	177.8	–
Backbone (DINOv2)	100.5	–	98.8
Action decoder	17.7	–	3.3
DiT refiner	120.3 (4 steps)	106.6 (5 steps)	–
Scorer	3.0	–	3.3
Sequential total	408.2	288.5	101.3
Sequential rate (FPS)	2.45	3.5	9.9
Dual-GPU parallel total	302.0	–	–
Dual-GPU rate (FPS)	3.3	–	–

Table 10: Parameter counts in millions (M), including frozen parameters. The total row excludes the VLM.

Component	ChainFlow-VLA	ReCogDrive	DrivoR
VLM (InternVL-2B, frozen)	2089	2089	–
Backbone (DINOv2-ViT-S/14 + LoRA)	23.73	–	23.73
Action decoder	4.03	–	10.97
DiT	21.37	34.33	–
Scorer	6.09	–	6.09
Total (excluding VLM)	55.22	34.33	40.79

Under dual-GPU execution, the perception backbone and AR decoder run in parallel with the VLM. The measured total is 302.0 ms, compared with 408.2 ms for sequential execution. The 4-step refiner processes 64 proposals in 120.3 ms. These latency measurements use the default 4-step configuration, rather than the 12-step configuration used for the best NAVSIM v1 score.

C VLM Feature Controls

Conditioning features are the last hidden states from the VLM's first forward pass, following ReCogDrive Li et al. [2025c]. We train with the released driving-fine-tuned ReCogDrive VLM and replace features only at inference. The original setting uses this VLM unchanged. The shuffled-command control randomizes driving commands within its official prompt template. The zero-input control uses hidden states produced by this VLM from all-zero inputs, rather than directly zeroing the hidden states. The generic control uses InternVL-3 without driving-specific fine-tuning.

Table 11: Inference-time VLM feature controls on NAVSIM v1. ΔPDMS is relative to the original-feature score.

Feature source	PDMS↑	ΔPDMS
Original ReCogDrive VLM	94.74	–
Shuffled driving commands	93.73	-1.01
Zero-input VLM	94.28	-0.46
Generic InternVL-3	94.42	-0.32

All three replacements reduce PDMS, with the largest drop under command shuffling. These controls support the utility of the original conditioning features, without isolating a specific semantic reasoning mechanism.

D Qualitative Component Analysis

Figure 4 complements the component analysis in Table 4. Without VLM guidance, the DiT refiner uses the default DrivoR scene tokens as conditioning. We compare this backbone-conditioned refinement with VLM-conditioned refinement to illustrate the effect of the conditioning source.



Figure 4: Qualitative comparison between BEV-conditioned and VLM-conditioned refinement. Expert trajectories are green; backbone-conditioned (ChainFlow-BEV) and VLM-conditioned (ChainFlow-VLA) trajectories are red and orange, respectively.

E Additional Qualitative Results

E.1 Comparison with Other Planners

We evaluate the qualitative performance of our method against alternative baselines on representative navtest scenarios, as illustrated in Fig. 5. In roundabout and left-turn ramp scenarios (Rows 1–2), while ReCogDrive and DrivoR either deviate from the drivable area or drift into incorrect lanes, our ChainFlow-VLA strictly adheres to navigation routes with collision-free maneuvers. For sharp turns (Row 3), it generates smooth, safe trajectories that closely match the expert trajectory, whereas both baselines fail by running off-road. In the intersection right-turn case (Row 4), our approach successfully bypasses static roadside vehicles to achieve higher ego progress than the expert trajectory without tailgating, while competing methods result in collisions. Furthermore, ChainFlow-VLA demonstrates robust safety by dynamically avoiding a static road barrier on the right side (Row 5)—a scenario where both baselines fail. Together, these results highlight the proposed model's superior scene understanding and robustness across diverse, challenging environments.



Figure 5: Qualitative comparison of trajectory predictions on representative NAVSIM scenarios. GT trajectories are shown in green. Predicted trajectories from ReCogDrive, DrivoR, and ChainFlow-VLA are visualized in red, purple, and orange, respectively.

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: All performance claims are supported by experimental results in Section 5.

   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: The limitations of our work are discussed in the conclusion section.

   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: [Yes]

   Justification: Section 3 presents the derivation of our method.

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: We provide comprehensive implementation details, including network architecture, loss functions, and the training pipeline. The code is publicly available.

   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: [Yes]

   Justification: The code is publicly available at the project repository linked in the abstract.

   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: Section 5 reports training configurations and evaluation settings; the appendices provide additional benchmark, inference-time, and feature-control details.

   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: We report benchmark scores and ablation comparisons without error bars, confidence intervals, or formal statistical significance tests. These results describe observed empirical performance rather than establish statistical significance.

   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: Section 5.1 reports the training configuration and an approximate total training time of two days for both stages. Appendix B reports the inference hardware and timings.

   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: Our work adheres to 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 method and its empirical evaluation and does not discuss both potential positive and negative societal impacts.

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: [No]

    Justification: The paper does not describe additional safeguards for the release of code or models.

    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: Appendix A identifies the asset sources and licenses, distinguishing code and model licenses from dataset distribution terms. The original works are credited, and the assets are used under their respective terms.

    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: [Yes]

    Justification: The new assets are documented in the accompanying project repository, with instructions for their use.

    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: This work does not involve crowdsourcing or research with human participants.

    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: This work does not involve research with human participants and therefore does not require IRB approval.

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: An open-source 2B vision-language model is a component of our method; its role and frozen conditioning features are described in Section 4 and Appendix C. A VLM was also used to assist manuscript writing and revision. No other LLM or VLM use was involved.

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.