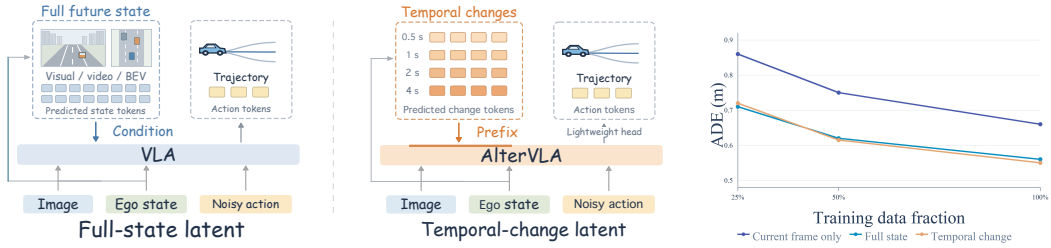
# ALTERVLA: A VISION-LANGUAGE-ACTION MODEL WITH LATENT TEMPORAL CHANGE PREDICTION FOR AUTONOMOUS DRIVING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Anticipating scene evolution can improve autonomous driving, but predicting complete future representations may devote modeling capacity to visual content already available in current observations. We introduce AlterVLA, a vision-language-action framework that uses compact, multi-horizon change tokens to focus foresight on scene evolution and provide complementary information for trajectory planning. Learning these representations in the pretrained model's own visual space avoids reliance on an external visual teacher. Predicted changes serve as direct planning conditions within a shared backbone, enabling continuous trajectory generation through a lightweight head without requiring future observations at inference. We further fine-tune AlterVLA with reinforcement learning to optimize safety, progress, and comfort beyond trajectory imitation. Accordingly, a four-stage training pipeline is developed to bridge future-supervised foresight and deployable planning. The planning interface is first learned with oracle changes and then preserved when predicted changes are introduced, before the policy is refined through reinforcement learning. Using a single front-view camera, AlterVLA achieves 88.53 PDMS on NAVSIM v1 and 87.82 EPDMS on NAVSIM v2 under supervised fine-tuning. With reinforcement fine-tuning, it achieves 90.94 PDMS on NAVSIM v1. Ablations show that temporal-change conditioning outperforms full-state prediction and the no-change baseline in PDMS. https://anonymous.4open.science/r/Alter_vla-369D/




(a) Foresight comparison.

(b) ADE across training data scales.

Figure 1: **Full-state versus temporal-change foresight.** (a) Full-state foresight uses predicted future visual, video, or BEV representations as planning conditions. AlterVLA instead predicts multi-horizon temporal change tokens and incorporates them into the VLA prefix to guide trajectory generation through a lightweight head. (b) Illustrative ADE comparison across training data fractions, where lower ADE indicates closer agreement with ground-truth trajectories. The temporal-change curve lies below the current-only baseline at all shown scales and below full-state prediction at 50% and 100% of the training data.

## 1 INTRODUCTION

End-to-end autonomous driving has progressed from specialized perception and planning networks toward policies built on pretrained vision-language models. Vision-language-action (VLA) models



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

extend this development by connecting visual understanding with trajectory generation (Zhou et al., 2026b; Huang et al., 2026). Recent driving world and action models further incorporate future prediction as auxiliary supervision or an explicit planning condition (Li et al., 2025c; Xie et al., 2026; Hong et al., 2026). These advances highlight the importance of representing future scene evolution and integrating it into continuous trajectory generation.

Future observations contain both shared scene content and temporal changes. When the planner already receives current visual representations, predicting complete future states may repeatedly model information that is already available. Visual changes caused by ego motion and surrounding agents provide complementary cues about scene evolution. Recent residual and visual-update modeling approaches suggest that these changes can be represented compactly (Zhang et al., 2026; Wang et al., 2026). This motivates learning temporal changes in the native visual feature space of a driving VLA and using them to guide action generation.

To this end, AlterVLA learns temporal change tokens from differences between current and future Qwen visual features 1. The same frozen visual encoder provides current perception and future supervision, avoiding an additional visual teacher or video-generation model. Changes are modeled at 0.5, 1, 2, and 4 seconds, preserving information about multiple future horizons. A staged procedure first trains the planner with oracle change codes extracted from future observations. A predictor then learns to reproduce these codes from current visual and ego information, preserving the planning interface without requiring future images at deployment.

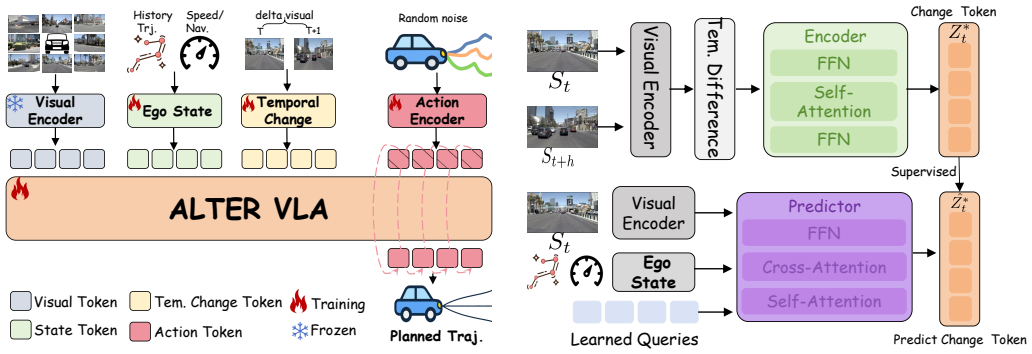
Predicted changes serve as direct planning conditions, extending their role beyond auxiliary training targets. AlterVLA incorporates change tokens into the Qwen prefix alongside current visual and ego-state tokens. The shared backbone processes this prefix together with noisy action tokens, allowing future-change information to condition each denoising iteration. A lightweight head predicts clean trajectories, which DPM-Solver++ uses for iterative sampling. This design connects native visual change modeling with continuous action generation. Experiments on NAVSIM demonstrate strong planning performance before reinforcement learning and support the benefit of multi-horizon change conditioning.

The main contributions are summarized as follows:

- We propose **AlterVLA**, which introduces multi-horizon temporal change tokens to represent scene evolution in the native visual space of a driving VLM. Predicted from current observations and ego information, these tokens provide explicit planning foresight without an additional visual teacher.
- We develop **horizon-aware change conditioning** that incorporates predicted change tokens into the foundation model prefix with their temporal identities. The shared backbone processes change and noisy action tokens together, directly conditioning trajectory denoising without a separate DiT planner.
- AlterVLA achieves **88.53 PDMS** on NAVSIM v1 and **87.82 EPDMS** on NAVSIM v2 under SFT, leading the listed SFT-based VLM methods on v1. Reinforcement fine-tuning increases v1 PDMS to **90.94**. Ablations show a **2.26-point PDMS gain** from multi-horizon temporal change conditioning.

## 2 Related Work

**Vision-Language-Action Models for Autonomous Driving.** The application of vision-language models to autonomous driving has gradually progressed from scene understanding to direct action generation. Early methods use large language models to generate scene descriptions, driving explanations, and high-level decisions, improving the semantic understanding and interpretability of driving systems (Xu et al., 2024; Sima et al., 2024; Jiang et al., 2026b). Subsequent approaches connect visual-language reasoning with low-level control through separate perception, reasoning, and planning modules (Fu et al., 2025; Yang et al., 2026; Xiong et al., 2026). More recent Vision-Language-Action models directly map visual observations, navigation instructions, and vehicle states to executable trajectories. AutoVLA represents trajectories as discrete action tokens for autoregressive generation (Zhou et al., 2026b), while diffusion and flow matching methods model continuous trajectory distributions (Jiang et al., 2025; Xiong et al., 2026). Recent unified architectures further




(a) Overall framework of AlterVLA.

(b) Temporal change representation and prediction.

Figure 2: Overview of AlterVLA. (a) Predicted change tokens enter the VLA prefix alongside current visual and ego-state tokens to condition trajectory denoising. (b) Compact change codes are learned from current-to-future visual feature differences and predicted from current observations and ego information.

process visual, language, ego state, and continuous action tokens within a shared backbone, enabling semantic understanding and trajectory generation through a common representation (Huang et al., 2026; Ma et al., 2026).

**Latent Foresight in World and Action Models.** World models provide future information for decision making by predicting how observations or latent states evolve over time. Early approaches mainly generate future images, videos, occupancy states, or discrete visual tokens for simulation and data synthesis (Hu et al., 2023a; Chen et al., 2025; Zhang et al., 2024). More recent methods perform future prediction in latent space and use the predicted representations to improve planning. LAW, FLARE, DriveVLA-W0, and LVDrive predict future latent or visual representations as auxiliary supervision for learning scene dynamics (Li et al., 2025b; Xie et al., 2026; **?**; Mei et al., 2026). DriveFuture, DriveWorld-VLA, DriveLaW, and related approaches further use predicted future states as direct conditions for trajectory generation (Hong et al., 2026; Liu et al., 2026; Xia et al., 2026). Another line of research represents temporal evolution through changes rather than complete future states. RLA-WM learns residual latent actions from differences between current and future DINO features for world modeling and policy learning (Zhang et al., 2026), while DeltaV models intermediate visual reasoning through compact visual updates instead of repeatedly generating complete visual states (Wang et al., 2026).

## 3 Methodology

AlterVLA uses predicted temporal changes as explicit conditions for continuous trajectory planning (Figure 2). Its design connects a compact representation of multi-horizon change, a predictor that estimates this representation from current inputs, and a shared backbone that uses the predicted changes during action denoising.

Given a current front-view image $I_t$, ego history, vehicle dynamics, and a navigation command, the policy predicts eight waypoints spanning four seconds. Each waypoint contains an ego-frame position and heading. A frozen visual encoder extracts features $\mathbf{S}_t = V(I_t)$. Future observations provide supervision during training and are unavailable at deployment.

### 3.1 Multi-Horizon Latent Change Representation

The current visual representation already provides the planner with scene context. We construct additional foresight around how this context changes over time. Current and future observations are encoded by the same frozen visual encoder, providing a common feature space for learning temporal changes without an additional visual teacher.

For each future horizon $h \in \mathcal{H} = \{0.5, 1, 2, 4\}$ seconds, a shared normalization module $\mathcal{N}$ defines a current-relative feature difference. A change encoder compresses this difference into an oracle code:

$$\Delta_{t,h} = \mathcal{N}(\mathbf{S}_{t+h}) - \mathcal{N}(\mathbf{S}_t), \qquad \mathbf{Z}^*_{t,h} = E_\phi(\Delta_{t,h}). \tag{1}$$

All horizons share the same current reference and encoder. The resulting codes describe changes at different temporal distances without recursively predicting intermediate states.

Following residual latent encoding (Zhang et al., 2026), the encoder jointly processes feature differences and learnable tokens through self-attention. The learned-token outputs form the compact code. A decoder $D_\psi$ reconstructs future visual features from the current features and this code. The representation objective $\mathcal{L}_{\text{repr}}$ combines mean absolute and mean squared reconstruction errors, averaged over future horizons.

Providing current features to the decoder encourages the code to retain information needed to explain the future given the present. Full future features supervise representation learning, while the compact change codes form the interface to planning. Encoder and decoder configurations are provided in the appendix.

### 3.2 Predicting Change from Current Observations

At deployment, the observations used to construct oracle change codes are unavailable. The predictor must supply compatible codes using information available at the current decision step. It reads current visual features $\mathbf{S}_t$, compressed visual tokens $\mathbf{V}_t$, and context tokens $\mathbf{C}_t$ encoding ego history, vehicle dynamics, and navigation. Horizon-specific query banks predict the corresponding future change codes.

Training aligns predictions with both the oracle codes and the future features they represent:

$$\begin{aligned} \widehat{\mathbf{Z}}_t &= P_\eta(\mathbf{S}_t, \mathbf{V}_t, \mathbf{C}_t), \\ \mathcal{L}_{\text{pred}} &= \text{SmoothL1}\left(\widehat{\mathbf{Z}}_t, \text{sg}(\mathbf{Z}^*_t)\right) \\ &\quad + \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \text{MAE}\left(D_\psi(\mathbf{S}_t, \widehat{\mathbf{Z}}_{t,h}), \mathbf{S}_{t+h}\right), \end{aligned} \tag{2}$$

where sg denotes stop-gradient and $\mathbf{Z}^*_t$ collects the oracle codes across horizons.

Latent matching aligns predictions with the code space used by the planner. Reconstruction additionally constrains what the predicted codes imply about future visual features. The decoder remains frozen, but gradients through its predicted-code input update the predictor. At inference, predicted codes replace oracle codes at the same planning interface, and the reconstruction decoder is discarded.

### 3.3 Horizon-Aware Change-Conditioned Planning

Changes at different future horizons provide distinct temporal context for the same planning decision. We preserve this distinction when transferring change codes into the language backbone. Each code is projected into the backbone hidden space and augmented with a learned horizon embedding $\mathbf{e}_h$:

$$\begin{aligned} \mathbf{T}_{t,h} &= \text{LN}\left(\Pi(\widehat{\mathbf{Z}}_{t,h}) + \mathbf{e}_h\right), \\ \mathbf{P}_t &= \left[\mathbf{V}_t; \mathbf{C}_t; \{\mathbf{T}_{t,h}\}_{h \in \mathcal{H}}\right], \end{aligned} \tag{3}$$

where $\Pi$ is a learned projection and $[\cdot;\cdot]$ denotes token concatenation. The prefix combines current scene context with anticipated changes and their temporal identities.

Building on shared-backbone action modeling (Huang et al., 2026), the Qwen3.5-2B language backbone processes this prefix together with noisy action tokens. A lightweight trajectory head reads the resulting action hidden states:

$$\widehat{\mathbf{A}}_0 = g_\omega\left(F_\theta\left([\mathbf{P}_t; E_{\text{act}}(\mathbf{A}_u, u)]\right)_{\text{act}}\right), \tag{4}$$

where $\mathbf{A}_u$ is a noisy normalized trajectory at diffusion time $u$. The head predicts its clean counterpart $\widehat{\mathbf{A}}_0$.

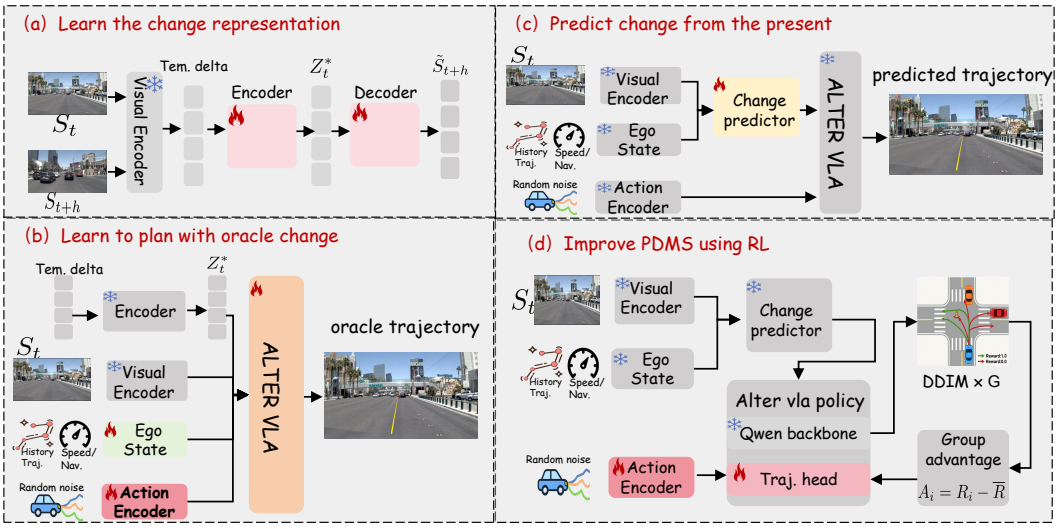



Figure 3: Four-stage training pipeline of AlterVLA. (a) Learn temporal change representations through future-feature reconstruction. (b) Train the planner with oracle change codes. (c) Train the predictor to supply compatible codes from current inputs while keeping the planner fixed. (d) Refine the trajectory head with Diffusion Dr.GRPO under predicted-change conditioning. The visual encoder remains frozen across all stages.

The predicted-change prefix remains fixed throughout sampling, making future-change information available at every denoising step. Temporal change therefore serves as a direct condition for action generation in addition to its role as a prediction target. This conditioning takes place within the shared backbone without a separate DiT planner.

Trajectory learning uses variance-preserving diffusion with clean-trajectory prediction. The action objective $\mathcal{L}_{\text{act}}$ supervises normalized waypoints and denormalized planar positions. At inference, DPM-Solver++ iteratively updates the trajectory using the clean estimates. Normalization, diffusion, and sampling details are provided in the appendix.

### 3.4 Reinforcement Learning with Diffusion Dr.GRPO

Reinforcement fine-tuning refines trajectory generation under the predicted-change conditions used at deployment. For each scene, the frozen predictor supplies a common prefix, and stochastic DDIM sampling generates $G$ candidate trajectories. Each candidate receives a PDMS-based reward $r_i$ that accounts for driving safety, compliance, progress, and comfort.

Diffusion Dr.GRPO centers rewards within each group without dividing by their standard deviation:

$$a_i = r_i - \frac{1}{G}\sum_{j=1}^{G} r_j. \tag{5}$$

The trajectory head is optimized with a reward-weighted denoising likelihood surrogate and a reference-policy regularizer:

$$\mathcal{L}_{\text{RL}} = -\mathbb{E}_{\text{scene}}\left[\frac{1}{G}\sum_{i=1}^{G} \text{sg}(a_i)\,\ell_{\omega,i}\right] + \beta\mathcal{L}_{\text{BC}}, \tag{6}$$

where $\ell_{\omega,i}$ aggregates discounted transition log-density surrogates over stochastic denoising steps. The reference term is computed from frozen reference-policy rollouts and regularizes changes to the pretrained behavior.

Only the trajectory head is updated. The predictor and remaining model parameters stay frozen, preserving the learned change interface during policy refinement. Reward composition, stochastic sampling, and likelihood-surrogate details are provided in Appendix A.2.

### 3.5 FOUR-STAGE TRAINING PIPELINE

The pipeline in Figure 3 separates learning to represent temporal change, learning to use it for planning, and learning to predict it from current observations. Oracle codes first establish the planner's conditioning interface. The predictor subsequently learns to supply compatible codes while that interface remains fixed. This ordering allows the planner to learn the use of future change before encountering prediction errors. The visual encoder remains frozen throughout.

**Stage I: Establishing the Change Representation.** The change encoder, reconstruction decoder, and shared normalization are optimized with $\mathcal{L}_{\text{repr}}$. Future-feature reconstruction supervises compact codes of current-relative changes at multiple horizons. The representation modules are then frozen, fixing the code space used by later stages.

**Stage II: Learning to Plan with Oracle Change.** Oracle codes from the frozen encoder replace predicted codes in the prefix. The language backbone and planning modules are trained with $\mathcal{L}_{\text{act}}$ to combine these codes with current visual and ego context. This stage establishes how temporal change is used for action generation independently of the accuracy of the change predictor.

Table 1: Performance comparison on NAVSIM v1 navtest. Bold indicates the best performance under SFT and RFT settings.

| Method | Sensors | NC↑ | DAC↑ | TTC↑ | Comf.↑ | EP↑ | PDMS↑ |
|---|---|---|---|---|---|---|---|
| Constant Velocity | – | 68.0 | 57.8 | 50.0 | 100.0 | 19.4 | 20.60 |
| Ego Status MLP | – | 93.0 | 77.3 | 83.6 | 100.0 | 62.8 | 65.60 |
| Hydra-MDP-$V_{8192}$ Li et al. (2024) | 3×Cam+L | 97.9 | 91.7 | 92.9 | 100.0 | 77.6 | 83.00 |
| UniAD Hu et al. (2023b) | 6×Cam | 97.8 | 91.9 | 92.9 | 100.0 | 78.8 | 83.40 |
| LTF Chitta et al. (2022) | 3×Cam | 97.4 | 92.8 | 92.4 | 100.0 | 79.0 | 83.80 |
| TransFuser Chitta et al. (2022) | 3×Cam+L | 97.7 | 92.8 | 92.8 | 100.0 | 79.2 | 84.00 |
| PARA-Drive Weng et al. (2024) | 6×Cam | 97.9 | 92.4 | 93.0 | 99.8 | 79.3 | 84.00 |
| Epona Zhang et al. (2025) | 3×Cam | 97.9 | 95.1 | 93.8 | 99.9 | 80.4 | 86.20 |
| Hydra-MDP-$V_{8192}$-W-EP Li et al. (2024) | 3×Cam+L | 98.3 | 96.0 | 94.6 | 100.0 | 78.7 | 86.50 |
| ARTEMIS Feng et al. (2025) | 3×Cam+L | 98.3 | 95.1 | 94.3 | 100.0 | 81.4 | 87.00 |
| DiffusionDrive Liao et al. (2025) | 3×Cam+L | 98.2 | 96.2 | 94.7 | 100.0 | 82.2 | 88.10 |
| WoTE Li et al. (2025d) | 3×Cam+L | 98.5 | 96.8 | 94.9 | 99.9 | 81.9 | 88.30 |
| **VLMs-based Methods (SFT)** | | | | | | | |
| AutoVLA-3B$^{\dagger}$ Zhou et al. (2026b) | 3×Cam | 96.9 | 92.4 | 88.1 | 99.1 | 75.8 | 80.50 |
| QwenVL3-4B$^{\star}$ | 1×Cam | 96.0 | 91.0 | 90.8 | 99.7 | 76.3 | 80.80 |
| ReCogDrive-2B Xiong et al. (2026) | 3×Cam | 97.6 | 93.1 | 92.7 | **100.0** | 79.1 | 84.10 |
| ReCogDrive-2B$^{\dagger}$ Xiong et al. (2026) | 3×Cam | 98.1 | 94.7 | 94.2 | **100.0** | 80.9 | 86.50 |
| ReCogDrive-8B$^{\dagger}$ Xiong et al. (2026) | 3×Cam | 98.3 | 95.1 | 94.3 | **100.0** | 81.1 | 86.80 |
| FLARE-4B Xie et al. (2026) | 1×Cam | 98.2 | 95.0 | 93.9 | **100.0** | 81.8 | 86.90 |
| Qwen-Drive-1.0-SFT Zhou et al. (2026a) | – | 98.4 | **96.6** | 94.7 | **100.0** | **82.4** | 88.20 |
| AlterVLA (ours) | 1×Cam | **99.7** | 96.1 | **97.5** | 95.7 | 82.0 | **88.53** |
| **VLMs-based Methods (RFT)** | | | | | | | |
| AutoVLA-3B Zhou et al. (2026b) | 3×Cam | 98.4 | 95.6 | **98.0** | 99.9 | 81.9 | 89.10 |
| DriveVLA-W0 Li et al. (2026) | 1×Cam | 98.7 | **99.1** | 95.3 | 99.3 | 83.3 | 90.20 |
| ReCogDrive-8B Xiong et al. (2026) | 3×Cam | 97.8 | 97.7 | 94.9 | **100.0** | 86.3 | 90.50 |
| ReCogDrive-2B Xiong et al. (2026) | 3×Cam | 97.9 | 97.3 | 94.9 | **100.0** | 87.3 | 90.80 |
| AutoVLA-3B$^{\ddagger}$ (Zhou et al., 2026b) | 3×Cam | 99.1 | 97.1 | 97.1 | **100.0** | 87.6 | 92.10 |
| DriveVLA-W0$^{\ddagger}$ Li et al. (2026) | 3×Cam | 99.3 | 97.4 | 97.0 | 99.9 | **88.3** | **93.00** |
| FLARE-4B Xie et al. (2026) | 1×Cam | 98.5 | 98.4 | 96.0 | **100.0** | 86.0 | 91.40 |
| Qwen-Drive-1.0-RL Zhou et al. (2026a) | – | 98.6 | 98.2 | 95.9 | **100.0** | 84.8 | 90.70 |
| AlterVLA (ours) | 1×Cam | **99.7** | 98.3 | 97.8 | **100.0** | 84.9 | 90.94 |

**Stage III: Replacing Oracle Change with Predictions.** The planner and representation modules remain frozen, and the predictor is optimized with $\mathcal{L}_{\text{pred}}$. Predicted codes occupy the same prefix positions and use the same horizon embeddings as oracle codes. The learned planning interface can therefore operate with current inputs, while latent matching and reconstruction constrain the predictions supplied to it.

**Stage IV: Reinforcing Planning with Predicted Change.** The trajectory head is refined with $\mathcal{L}_{\text{RL}}$ under predicted-change conditioning, following Section 3.4. The predictor and remaining model parameters stay frozen. This final stage optimizes driving behavior under the foresight conditions available at deployment.

At inference, current observations and ego information produce change tokens that directly condition trajectory generation. Future images, the oracle change encoder, and the reconstruction decoder are unnecessary.

Table 2: Reported performance on NAVSIM v2 navtest.

| Method | NC↑ | DAC↑ | EP↑ | TTC↑ | HC↑ | TLC↑ | DDC↑ | LK↑ | EC↑ | EPDMS↑ |
|---|---|---|---|---|---|---|---|---|---|---|
| TransFuser Chitta et al. (2022) | 97.7 | 92.8 | 79.2 | 92.8 | **100.0** | 99.9 | 98.3 | 67.6 | 95.3 | 77.8 |
| Hydra-MDP Li et al. (2024) | 97.5 | 96.3 | 80.1 | 93.0 | **100.0** | 99.9 | 98.3 | 65.5 | 97.4 | 79.8 |
| Hydra-MDP++ Li et al. (2025a) | 97.9 | 96.5 | 79.2 | 93.4 | **100.0** | **100.0** | 98.9 | 67.2 | 97.7 | 80.6 |
| ARTEMIS Feng et al. (2025) | 98.3 | 95.1 | 81.5 | 97.4 | **100.0** | 99.8 | 98.6 | 96.5 | **98.3** | 83.1 |
| ReCogDrive-8B Xiong et al. (2026) | 98.3 | 95.2 | 87.1 | 97.5 | 98.3 | 99.8 | 99.5 | 96.6 | 86.5 | 83.6 |
| DiffusionDrive Liao et al. (2025) | 98.0 | 96.0 | 87.7 | 97.1 | 98.3 | 99.8 | 99.5 | 97.2 | 87.6 | 84.3 |
| VADv2 Jiang et al. (2026a) | 98.0 | 98.3 | 87.1 | 96.8 | 98.3 | 99.8 | 99.4 | 95.2 | 88.1 | 85.8 |
| DriveVLA-W0 Li et al. (2026) | 98.5 | **99.1** | 86.4 | 98.1 | 97.9 | 99.7 | 98.0 | 93.2 | 58.9 | 86.1 |
| AutoDrive-P$^3$ (Detailed)$^\dagger$ Ye et al. (2026) | 99.1 | 97.4 | **88.0** | 98.7 | 98.3 | 99.8 | 99.2 | 96.3 | 85.5 | 86.2 |
| FLARE-4B Xie et al. (2026) | 98.8 | 96.6 | 87.9 | 98.2 | 98.4 | 99.9 | 99.5 | 95.8 | 87.5 | 86.3 |
| DriveWorld-VLA (Liu et al., 2026) | 98.6 | **99.1** | 87.4 | 97.9 | 97.8 | 99.8 | 99.6 | 97.0 | 78.6 | 86.8 |
| AlterVLA (ours) | **99.7** | 96.1 | 86.0 | **99.0** | 93.0 | 99.9 | **99.7** | **98.3** | 78.45 | **87.8** |

# 4 EXPERIMENT

## 4.1 EXPERIMENTAL SETUP

**Implementation Details.** AlterVLA adopts Qwen3.5-2B with a frozen visual encoder. Temporal change learning uses future supervision at 0.5, 1, 2, and 4 seconds. Trajectory generation employs $x_0$ prediction with DPM-Solver++, followed by Diffusion Dr.GRPO fine-tuning using a PDMS-based reward. Optimization details are provided in the appendix.

**Dataset.** Experiments use the NAVSIM benchmark (Dauner et al., 2024), with navtrain for training and navtest for evaluation. Given a current front-view image, ego history, vehicle dynamics, and navigation command, AlterVLA predicts eight waypoints over four seconds. Future images are used only during training.

**Evaluation Metrics.** NAVSIM v1 (Dauner et al., 2024) performance is measured by PDMS and its component scores, with DDC additionally reported. The evaluation weights for progress, TTC, and comfort are 5, 5, and 2, respectively. NAVSIM v2 (Cao et al., 2025) uses EPDMS with extended compliance and comfort metrics.

## 4.2 MAIN RESULTS

Table 1 compares planning performance on NAVSIM v1 under SFT and RFT settings. With a single front-view camera and a Qwen3.5-2B backbone, AlterVLA achieves 88.53 PDMS before RL, the highest among the listed SFT-based VLM methods. It surpasses FLARE-4B and Qwen-Drive-1.0-SFT by 1.63 and 0.33 points, respectively. Its NC and TTC scores of 99.7 and 97.5 also lead

the listed SFT methods, demonstrating strong collision-avoidance performance before reward-based optimization. Reinforcement fine-tuning further improves PDMS to 90.94, a gain of 2.41 points.

Table 2 reports NAVSIM v2 results. AlterVLA achieves 87.82 EPDMS, exceeding the reported scores of DriveWorld-VLA and FLARE-4B by 1.02 and 1.52 points, respectively. NC, TTC, and lane-keeping scores reach 99.68, 99.01, and 98.30, highlighting collision avoidance and lane following as strengths. However, lower drivable-area compliance and extended comfort scores indicate room for improvement. Cross-paper comparisons should account for evaluation differences, including human penalty filtering.

**Effect of Future Representations and Temporal Horizons.** Table 3 examines the contribution of future conditioning, the choice of future representation, and the coverage of prediction horizons. These comparisons assess whether the benefit comes from introducing future information alone or also from how that information is represented and organized. Full-state prediction achieves 88.20 PDMS, compared with 86.27 without temporal change, showing the benefit of future conditioning. Multi-horizon change prediction further increases PDMS to 88.53, with higher NC (99.68 vs. 99.20) and DAC (96.12 vs. 95.45), while matching EP, TTC, and comfort at the reported precision. This result favors temporal-change conditioning over full-state prediction in the evaluated configurations. Single-horizon change prediction at 1 and 2 seconds achieves 87.20 and 87.63 PDMS, respectively. The stronger multi-horizon result supports incorporating both near-term and longer-term scene evolution into planning. Overall, predicted multi-horizon changes improve PDMS by 2.26 points over the no-change baseline. The oracle variant further evaluates performance when change tokens are extracted from ground-truth future observations. It reaches 89.10 PDMS, leaving a 0.57-point gap to predicted conditioning. This comparison shows that the deployable predictor approaches the oracle reference without requiring future observations at inference.

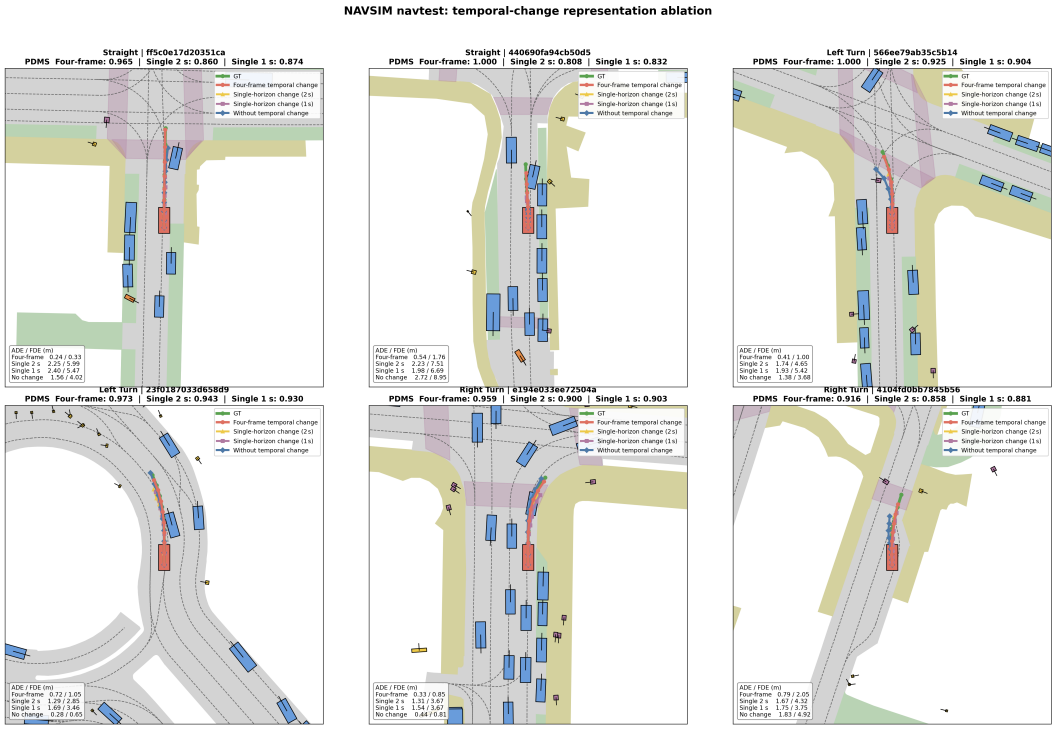



Figure 4: Qualitative comparison on NAVSIM navtest across straight-driving, left-turn, and right-turn scenarios. Green denotes GT; red, yellow, purple, and blue represent four-frame temporal change, single-horizon change at 2 s and 1 s, and planning without temporal change, respectively.

**Qualitative Analysis of Temporal Change.** Figure 4 compares predicted trajectories across straight-driving and turning scenarios. Multi-horizon temporal change achieves lower ADE and FDE than both single-horizon variants in all six displayed cases. The trajectories more closely match

Table 3: Ablation of future representations and temporal horizons. Bold indicates the best result among deployable variants. The oracle reference uses ground-truth future observations.

| Variant | NC↑ | DAC↑ | EP↑ | TTC↑ | Comf.↑ | PDMS↑ |
|---|---|---|---|---|---|---|
| Without temporal change | 98.90 | 95.80 | 80.60 | 94.10 | 95.20 | 86.27 |
| Multi-horizon full-state prediction | 99.20 | 95.45 | **82.03** | **97.51** | **95.74** | 88.20 |
| Single-horizon change (1 s) | 99.40 | 96.10 | 81.30 | 95.40 | 95.10 | 87.20 |
| Single-horizon change (2 s) | 99.30 | 96.10 | 81.20 | 96.70 | 95.30 | 87.63 |
| Multi-horizon change (predicted) | **99.68** | **96.12** | **82.03** | **97.51** | **95.74** | **88.53** |
| *Oracle reference* | | | | | | |
| Multi-horizon change (oracle) | 99.71 | 96.55 | 82.70 | 97.72 | 96.05 | 89.10 |

GT in longitudinal extent during straight driving and path curvature through turns, with smaller endpoint deviations. In the top-middle example, ADE/FDE decrease from 2.72/8.95 m without temporal change to 0.54/1.76 m. Compared with the no-change baseline, multi-horizon conditioning reduces ADE in five of the six cases, although endpoint improvements are not universal. These observations support the benefit of multi-horizon change information for maintaining trajectory fidelity over the planning horizon.

**Effect of Trajectory Generation Methods.** Table 4 compares DDIM, flow matching, and DPM-Solver++ to examine how trajectory generation affects change-conditioned planning. DPM-Solver++ achieves the highest PDMS of 88.53, compared with 88.19 for DDIM and 87.97 for flow matching. It also obtains the highest NC, EP, TTC, and comfort scores among the evaluated configurations. Relative to DDIM, TTC and comfort improve by 0.51 and 1.24 points, respectively, alongside a modest increase in ego progress. DDIM retains a 0.28-point advantage in DAC, indicating a small trade-off in road-boundary compliance. These results support the adopted DPM-Solver++ configuration for overall planning performance. The comparison evaluates complete generation configurations, as both formulation and sampling differ across the alternatives.

Table 4: Ablation of trajectory generation methods.

| Method | NC↑ | DAC↑ | EP↑ | TTC↑ | Comf.↑ | PDMS↑ |
|---|---|---|---|---|---|---|
| DDIM | 99.50 | **96.40** | 81.87 | 97.00 | 94.50 | 88.19 |
| Flow Matching | 99.40 | 96.20 | 81.69 | 96.90 | 94.60 | 87.97 |
| DPM-Solver++ | **99.68** | 96.12 | **82.03** | **97.51** | **95.74** | **88.53** |

## 5 Conclusion

AlterVLA integrates visual foresight with continuous trajectory generation using temporal change tokens learned in the feature space of a frozen VLM visual encoder. Predicted multi-horizon changes enter the shared backbone prefix and condition action denoising through a lightweight trajectory head, without a separate DiT planner. Experiments on NAVSIM demonstrate strong supervised planning performance, while ablations support temporal-change conditioning and its extension across multiple future horizons.

A remaining limitation is that the temporal change predictor is deterministic. Although change conditioning improves supervised trajectory generation, all RL rollouts for a given scene receive the same predicted change tokens. This shared condition may bias sampling toward a limited set of behaviors and reduce exploration, despite stochastic action denoising. Whether this effect constrains RL gains requires further investigation. Future work will measure trajectory diversity and explore stochastic or multimodal change prediction, together with diversity-aware rollout strategies, to improve exploration and the effectiveness of reinforcement fine-tuning.

## AI USE STATEMENT

Generative AI tools were used to assist with language polishing, including grammar, clarity, and readability. The authors reviewed the AI-assisted revisions and take full responsibility for the accuracy and integrity of the final manuscript.

## ETHICS STATEMENT

AlterVLA is evaluated on NAVSIM using recorded driving scenes and simulation-based metrics. These benchmark results do not establish safety for deployment on public roads. Real-world deployment requires further validation under diverse traffic conditions and appropriate safety safeguards. Responsible use of the data and model requires adherence to the source data licenses and protection of privacy in driving-scene imagery.

## REPRODUCIBILITY STATEMENT

The architecture and training objectives are described in Section 3. Appendix A.2 provides additional details on temporal change modeling, trajectory generation, and reinforcement fine-tuning. Model configurations, four-stage training hyperparameters, and sampling settings are documented in Tables 5, 6, and 7, respectively, to support reproduction of the reported experiments.

## 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 Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 4709–4722. PMLR, 27–30 Sep 2025.

Yuntao Chen, Yuqi Wang, and Zhaoxiang Zhang. Drivinggpt: Unifying driving world modeling and planning with multi-modal autoregressive transformers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 26890–26900. IEEE, 2025.

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

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

Renju Feng, Ning Xi, Duanfeng Chu, Rukang Wang, Zejian Deng, Anzheng Wang, Liping Lu, Jinxiang Wang, and Yanjun Huang. Artemis: Autoregressive end-to-end trajectory planning with mixture of experts for autonomous driving. *IEEE Robotics and Automation Letters*, 11(1):226–233, 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 *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24823–24834. IEEE, 2025.

Yufeng Hong, Xiaotian Zhou, Yingyan Li, Xiangpo Zhou, Lin Liu, Yadan Luo, Shaoqing Xu, Lei Yang, and Ziying Song. Drivefuture: Future-aware latent world models for autonomous driving. *arXiv preprint arXiv:2605.09701*, 2026.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023a.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17853–17862, June 2023b.

Yuzhou Huang, Benjin Zhu, Hengtong Lu, Victor Shea-Jay Huang, Haiming Zhang, Wei Chen, Jifeng Dai, Yan Xie, and Hongsheng Li. Mindvla-u1: Vla beats va with unified streaming architecture for autonomous driving. *arXiv preprint arXiv:2605.12624*, 2026.

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, Hao Gao, Bencheng Liao, Qian Zhang, Wenyu Liu, and Xinggang Wang. Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. In *International Conference on Learning Representations*, volume 2026, pp. 68886–68900, 2026a.

Bo Jiang, Shaoyu Chen, Bencheng Liao, Xingyu Zhang, Wei Yin, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Senna: Bridging large vision-language models and end-to-end autonomous driving. *International Journal of Computer Vision*, 134(9):407, 2026b.

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.

Yingyan Li, Lue Fan, Jiawei He, Yuqi Wang, Yuntao Chen, Zhaoxiang Zhang, and Tieniu Tan. Enhancing end-to-end autonomous driving with latent world model. In *International Conference on Learning Representations*, volume 2025, pp. 42942–42959, 2025b.

Yingyan Li, Lue Fan, Jiawei He, Yuqi Wang, Yuntao Chen, Zhaoxiang Zhang, and Tieniu Tan. Enhancing end-to-end autonomous driving with latent world model, 2025c. URL https://arxiv.org/abs/2406.08481.

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

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, LU HOU, Lue Fan, and Zhaoxiang Zhang. Drivevla-w0: World models amplify data scaling law in autonomous driving. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 7890–7911, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0d70423f59c5fdd24f0dd3fa52e34623-Paper-Conference.pdf.

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

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

Lin Liu, Ziying Song, Caiyan Jia, Hangjun Ye, Xiaoshuai Hao, Long Chen, et al. Driveworld-vla: Unified latent-space world modeling with vision-language-action for autonomous driving. *arXiv preprint arXiv:2602.06521*, 2026.

Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui Qiu, Yuqiang Yang, et al. Internvla-a1. 5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint arXiv:2607.04988*, 2026.

Xiaodong Mei, Diankun Zhang, Hongwei Xie, Guang Chen, Hangjun Ye, and Dan Xu. Lvdrive: Latent visual representation enhanced vision-language-action autonomous driving model. *arXiv preprint arXiv:2605.22089*, 2026.

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

Pengjie Wang, Linger Deng, Zujia Zhang, Shaojie Zhang, Zhenbo Luo, Pei Fu, Jian Luan, Xiang Bai, and Yuliang Liu. Deltav: Thinking with visual state updates in unified large multimodal models. *arXiv preprint arXiv:2607.08434*, 2026.

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

Tianze Xia, Yongkang Li, Lijun Zhou, Jingfeng Yao, Kaixin Xiong, Haiyang Sun, Bing Wang, Kun Ma, Guang Chen, Hangjun Ye, et al. Drivelaw: Unifying planning and video generation in a latent driving world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 39701–39712, 2026.

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:2601.05611*, 2026.

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

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

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10678–10688, 2026.

Yuqi Ye, Zijian Zhang, Junhong Lin, Shangkun Sun, Changhao Peng, and Wei Gao. $autodrive\backslash$text {-} pˆ 3: $Unifiedchainofperception - prediction - planningthoughtviareinforcementfine - tuning$. *arXiv preprint arXiv:2603.28116*, 2026.

Kaiwen Zhang, Zhenyu Tang, Xiaotao Hu, Xingang Pan, Xiaoyang Guo, Yuan Liu, Jingwei Huang, Li Yuan, Qian Zhang, Xiao-Xiao Long, et al. Epona: Autoregressive diffusion world model for autonomous driving. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 27220–27230. IEEE, 2025.

Lunjun Zhang, Yuwen Xiong, Ze Yang, Sergio Casas, Rui Hu, and Raquel Urtasun. Copilot4d: Learning unsupervised world models for autonomous driving via discrete diffusion. In *International Conference on Learning Representations*, volume 2024, pp. 7269–7299, 2024.

Xinyu Zhang, Zhengtong Xu, Yutian Tao, Yeping Wang, Yu She, and Abdeslam Boularias. Learning visual feature-based world models via residual latent action. *arXiv preprint arXiv:2605.07079*, 2026.

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

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

# A Appendix

## A.1 Motivation and Qualitative Analysis

**Why predict temporal changes?** Current observations already provide the planner with scene appearance and spatial context. Future representations contain both this shared content and information about how the scene evolves. AlterVLA therefore represents future evolution relative to the current observation:

$$\Delta_{t,h} = \mathcal{N}(\mathbf{S}_{t+h}) - \mathcal{N}(\mathbf{S}_t), \qquad h \in \mathcal{H}, \tag{7}$$

where $\mathcal{N}$ denotes a shared feature normalization and $\mathcal{H} = \{0.5, 1, 2, 4\}$ seconds. All horizons use the same current reference. The resulting change tokens complement the current visual prefix with information about future evolution.

This formulation introduces a representation bias rather than additional information: given the current features, an uncompressed feature difference and the corresponding future features determine each other. The motivation is to allocate a limited latent representation explicitly to temporal variation. Such variation can reflect ego motion, changing viewpoints, and the evolution of surrounding traffic; it should not be interpreted as isolated object motion. Multiple horizons provide supervision over both near-term and longer-term scene evolution.

**Qualitative evidence.** Figure 5 visualizes attention from change tokens to visual patches in turning and low-light scenes. Attention is distributed over road structure, surrounding vehicles, and other scene regions whose appearance evolves over time. These observations illustrate the visual information aggregated by the change representation. They do not establish a causal explanation of planning decisions or demonstrate superiority over full-state prediction. The planning ablation in Table 3 provides complementary evidence for the benefit of change conditioning in the evaluated configurations.

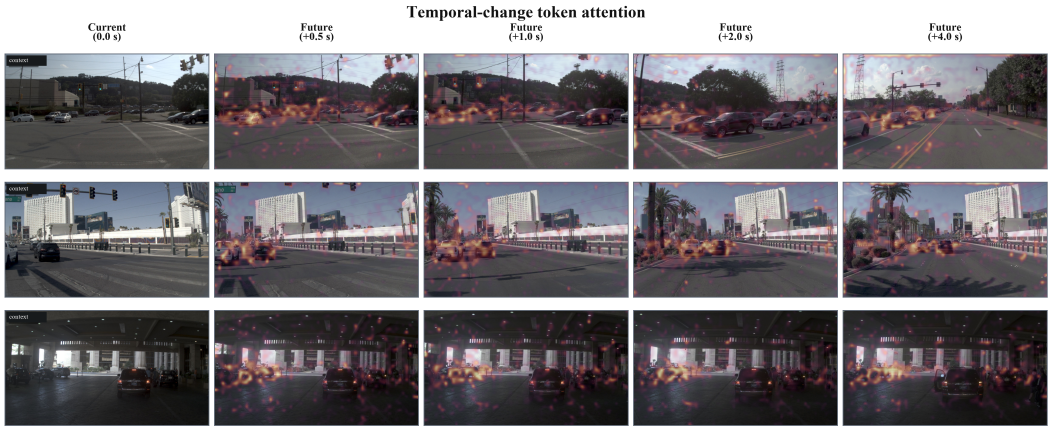



Figure 5: **Attention of temporal-change tokens.** Each row shows the current observation and future views at four horizons. Heatmaps aggregate attention from change tokens to visual patches, averaged over the last four layers of the change encoder, eight attention heads, and 32 change tokens. These visualizations analyze the oracle change encoder using current and future observations. Future observations are not required by the deployed predictor or planner.

## A.2 Additional Method Details

**Change encoder and reconstruction decoder.** The frozen visual encoder extracts current and future visual features. After shared normalization, their differences are projected to the representation width and concatenated with learned query tokens. A Transformer encoder processes this sequence, and the query outputs are projected to compact change tokens.

During representation learning, a reconstruction decoder combines the change tokens with current visual features to reconstruct the corresponding future features:

$$\widehat{\mathbf{S}}_{t+h} = D_\psi\big(\mathbf{S}_t, E_\phi(\Delta_{t,h})\big). \tag{8}$$

The decoder projects normalized current features and change tokens into a shared space, processes their concatenation with Transformer blocks, and reads reconstructed features from the visual-token positions. Its targets are the original future visual features. The representation loss is

$$\mathcal{L}_{\text{repr}} = \frac{1}{|\mathcal{H}|}\sum_{h\in\mathcal{H}}\Big[\text{MAE}(\widehat{\mathbf{S}}_{t+h}, \mathbf{S}_{t+h}) + \text{MSE}(\widehat{\mathbf{S}}_{t+h}, \mathbf{S}_{t+h})\Big], \tag{9}$$

where both errors are averaged over feature elements. Reconstruction encourages the compressed tokens to retain information about future evolution. The decoder supplies training supervision and is not used during trajectory generation.

**Change prediction.** The predictor uses horizon-specific learned queries to estimate change tokens from current inputs. Its memory contains current visual features, compressed visual tokens, and ego-context tokens, with separate projections and token-type embeddings. Each predictor block contains query self-attention, cross-attention to this memory, and a feed-forward network. Queries for all horizons are processed together, allowing information exchange across horizons.

The predictor is supervised by frozen oracle change tokens and future-feature reconstruction through the frozen decoder:

$$\begin{aligned}\mathcal{L}_{\text{pred}} = \ &\text{SmoothL1}\left(\widehat{\mathbf{Z}}_t, \text{sg}(\mathbf{Z}_t^*)\right) \\ &+ \frac{1}{|\mathcal{H}|}\sum_{h\in\mathcal{H}}\text{MAE}\left(D_\psi(\mathbf{S}_t, \widehat{\mathbf{Z}}_{t,h}), \mathbf{S}_{t+h}\right).\end{aligned} \tag{10}$$

Although the decoder parameters remain frozen, the reconstruction loss propagates gradients to the predictor through its predicted tokens. No trajectory loss is used in this stage.

**Planning prefix and action tokens.** Current visual features are compressed to 256 tokens. Two additional tokens encode ego history and vehicle dynamics with the navigation command. The predictor produces 32 change tokens per horizon, giving 128 change tokens in total. After projection to the backbone width and addition of horizon embeddings, these tokens form a 386-token planning prefix:

$$\mathbf{P}_t = [\mathbf{V}_t; \mathbf{C}_t; \mathbf{T}_{t,0.5}; \mathbf{T}_{t,1}; \mathbf{T}_{t,2}; \mathbf{T}_{t,4}]. \tag{11}$$

Eight noisy action tokens are appended to this prefix. Each action token combines an embedded noisy waypoint with a diffusion-time embedding. The shared backbone processes the resulting sequence, and a lightweight multilayer perceptron maps action hidden states to clean trajectory estimates.

**Trajectory normalization and supervision.** Each trajectory contains eight ego-frame waypoints $(x, y, \theta)$ spanning four seconds. For each coordinate, the first and 99th percentiles of the training trajectories define normalization bounds $\mathbf{l}$ and $\mathbf{u}$:

$$\mathbf{A}_0 = 2\frac{\mathbf{A}-\mathbf{l}}{\mathbf{u}-\mathbf{l}} - 1, \tag{12}$$

with elementwise operations. Predicted normalized coordinates are clipped to $[-1, 1]$ when converted back to physical coordinates.

Training corrupts the normalized trajectory using a variance-preserving diffusion process, $\mathbf{A}_s = \alpha(s)\mathbf{A}_0 + \sigma(s)\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ and $s$ is sampled uniformly from $[10^{-3}, 1]$. The planning objective combines normalized trajectory regression with a position loss in physical coordinates:

$$\begin{aligned}\mathcal{L}_{\text{act}} = \ &\frac{1}{8}\sum_{j=1}^{8}\left\|\widehat{\mathbf{A}}_{0,j} - \mathbf{A}_{0,j}\right\|_2^2 \\ &+ 0.1\ \text{SmoothL1}\left(\widehat{\mathbf{A}}^{xy}, \mathbf{A}^{xy}\right).\end{aligned} \tag{13}$$

DPM-Solver++ is used for trajectory sampling at inference.

**Reinforcement-learning implementation.** Reinforcement fine-tuning updates only the trajectory head; the remaining modules are frozen, and no LoRA adapters are used. For each scene, stochastic DDIM sampling generates a group of $G$ candidate trajectories. Their rewards are computed as

$$r_i = \mathrm{NC}_i \mathrm{DAC}_i \mathrm{DDC}_i \frac{w_{\mathrm{EP}} \mathrm{EP}_i + w_{\mathrm{TTC}} \mathrm{TTC}_i + w_{\mathrm{C}} \mathrm{C}_i}{w_{\mathrm{EP}} + w_{\mathrm{TTC}} + w_{\mathrm{C}}}. \tag{14}$$

All reward components are evaluated on the $[0, 1]$ scale. Training uses weights $(10, 5, 2)$ for progress, time-to-collision, and comfort, whereas reported NAVSIM v1 evaluation uses $(5, 5, 2)$.

The group advantage subtracts the mean reward without dividing by the group standard deviation. For stochastic denoising transition $k$, the optimization surrogate averages clipped Gaussian log densities over the 24 trajectory coordinates:

$$\ell_{i,k} = \frac{1}{24} \sum_{d=1}^{24} \mathrm{clip}\left(\log p_\omega(a_{i,k+1,d} \mid \mathbf{A}_{i,k}, \mathbf{P}_t), -5, 2\right). \tag{15}$$

The final deterministic transition is excluded. The resulting objective is

$$\mathcal{L}_{\mathrm{RL}} = -\mathbb{E}\left[\frac{1}{GK_s} \sum_{i=1}^{G} \sum_{k=0}^{K_s-1} \mathrm{sg}(a_i) \gamma^{K_s-1-k} \ell_{i,k}\right] + \beta \mathcal{L}_{\mathrm{BC}}, \tag{16}$$

where $K_s$ is the number of stochastic transitions. The reference loss $\mathcal{L}_{\mathrm{BC}}$ is the negative mean transition log-density surrogate on chains sampled from a frozen pre-RL policy, without the denoising discount. It regularizes the updated policy toward its supervised initialization. Each fresh rollout batch is used for one optimization update, without a PPO likelihood ratio or ratio clipping.

### A.3 Four-Stage Training Configuration

Training separates representation learning, planning, change prediction, and reward-based refinement. The visual encoder remains frozen throughout. Stage I trains the change encoder and reconstruction decoder. Stage II trains the language backbone and planning modules using frozen oracle change tokens. Stage III trains only the change predictor while keeping the representation model and planner frozen. Stage IV uses predicted changes and updates only the trajectory head.

Tables 5–7 summarize the architecture, optimization, and sampling settings. Batch sizes count scenes before group sampling. Gradient accumulation is adjusted to obtain the specified global scene batch size.

Table 5: Architecture configuration of AlterVLA.

| Configuration | Value |
|---|---|
| Backbone | Qwen3.5-2B |
| Input cameras | One front-view camera |
| Future horizons (s) | $\{0.5, 1, 2, 4\}$ |
| Queries per horizon | 32 |
| Total change tokens | 128 |
| Change latent dimension | 256 |
| Encoder / decoder / predictor layers | 12 / 12 / 12 |
| Representation hidden width | 512 |
| Attention heads | 8 |
| Feed-forward width | 2048 |
| Visual prefix tokens | 256 |
| Ego-context tokens | 2 |
| Total prefix tokens | 386 |
| Action tokens | 8 |
| Trajectory duration | 4 s |
| Trajectory-head widths | $512 \to 256 \to 3$ |

Table 6: Four-stage optimization settings from the provided training configurations. Global batch size counts scenes.

| Setting | Stage I | Stage II | Stage III | Stage IV |
|---|---|---|---|---|
| Objective | Representation | Oracle planning | Change prediction | RL refinement |
| Trainable modules | Change encoder and decoder | Language backbone and planning modules | Change predictor | Trajectory head |
| Optimizer | AdamW | AdamW | AdamW | AdamW |
| Peak learning rate | $10^{-4}$ | $2 \times 10^{-5}$ | $2 \times 10^{-5}$ | $10^{-6}$ |
| Global scene batch size | 16 | 16 | 16 | 8 |
| Per-device micro-batch size | 1 | 1 | 1 | 1 |
| Maximum optimization steps | 20,000 | 20,000 | 20,000 | 2,000 |
| Warmup ratio | 0.05 | 0.05 | 0.05 | 0 |
| Learning-rate schedule | Cosine | Cosine | Cosine | Constant |
| Weight decay | 0 | $10^{-5}$ | $10^{-5}$ | 0 |
| Gradient clipping norm | 1.0 | 1.0 | 1.0 | 1.0 |
| Precision | BF16 | BF16 | BF16 | BF16 |
| Random seed | 42 | 42 | 42 | 42 |
| Reconstruction MAE / MSE weights | $1/1$ | – | – | – |
| Position-loss weight | – | 0.1 | – | – |
| Latent / reconstruction loss weights | – | – | $1/1$ | – |
| Trajectory-loss weight in predictor training | – | – | 0 | – |
| Reference-policy loss weight | – | – | – | 0.1 |
| LoRA | No | No | No | No |

Table 7: Trajectory diffusion and reinforcement-learning sampling settings.

| Setting | Value |
|---|---|
| *Supervised trajectory generation* | |
| Normalization percentiles | 1st / 99th |
| Diffusion process | Linear VP |
| Minimum / maximum diffusion rate | $0.1/20$ |
| Minimum diffusion time | $10^{-3}$ |
| Inference sampler | DPM-Solver++ |
| Inference denoising steps | 10 |
| Inference initial-noise scale | 0.5 |
| *Reinforcement fine-tuning* | |
| Rollout sampler | Stochastic DDIM |
| Candidates per scene $G$ | 8 |
| Total denoising transitions $K$ | 5 |
| Optimized stochastic transitions $K_s$ | 4 |
| DDIM stochasticity $\eta$ | 1.0 |
| Rollout initial-noise scale | 1.0 |
| Minimum transition standard deviation | 0.04 |
| Clean-trajectory clipping range | $[-1, 1]$ |
| Transition-noise clipping range | $[-5, 5]$ |
| Per-coordinate log-density clipping | $[-5, 2]$ |
| Denoising discount $\gamma$ | 0.6 |
| Reference-policy coefficient $\beta$ | 0.1 |
| Updates per rollout batch | 1 |
| Training reward weights (EP / TTC / C) | $10/5/2$ |
| Evaluation weights (EP / TTC / C) | $5/5/2$ |