TRACKFORESIGHT: POINT-TRACK-BASED VISUAL COT FOR AUTONOMOUS VLAS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive vision-language-action (VLA) models have recently attracted growing interest in autonomous driving, owing to their inherited reasoning capabilities, strong pretrained initializations, and well-established architectures. Despite the elegance of this design, standard end-to-end training asks the model to predict sparse action targets directly from raw sensor inputs, which is a difficult supervision problem that leaves little direct supervision for scene structure and agent behavior. We propose *TrackForesight*, which guides the policy by tracking driving relevant objects in the context via a point tracker, and training the model to predict their future positions as visual chain-of-thought before emitting the final trajectory. We compare point tracks against dense RGB and semantic foresight modalities and show that they offer an excellent balance between predictability, richness, and efficiency, preserving short-horizon motion with far fewer intermediate tokens. Relative to standard action-only training, After GRPO, TrackForesight improves PDMS on NAVSIM navtest by 1.9, reaching a state-of-the-art 92.1 among VLA-based planners on the NAVSIM v1 benchmark, with only 60 generated foresight tokens and no additional training stage. The results suggest that effective foresight for driving VLAs hinges on the information content of the prediction target, the training balance between foresight and action objectives, and the cost of generating intermediate tokens before action.

1 INTRODUCTION

End-to-end training of driving policies has attracted considerable interest with the availability of large-scale driving datasets and powerful pretrained models (Caesar et al., 2021; Dauner et al., 2024; Wang et al., 2024b). Autoregressive vision-language-action (VLA) models make this setup conceptually simple: driving context is tokenized, fed to an autoregressive model, and decoded into a tokenized future trajectory (Zhou et al., 2026; 2025). Despite this elegance, reproducing sparse actions from high-dimensional observations is a difficult supervision problem. The driving expert acts with access to scene structure and other agents' behavior, while the model is asked to imitate only the final trajectory. This bottleneck is especially important in supervised fine-tuning, the stage that turns a pretrained vision-language model into a driving policy before any later post-training (Zhou et al., 2025; Shang et al., 2025).

A recently explored approach to this problem is to introduce chain-of-thought(CoT) tokens that the model predicts before the final trajectory (Wei et al., 2022; Zhou et al., 2025). In driving, the most immediate source of such tokens is language. DriveGPT4 (Xu et al., 2024), DriveVLM (Tian et al., 2024), and AutoVLA (Zhou et al., 2025) use textual reasoning to summarize the scene, restate route intent, describe traffic rules, or declare a high-level maneuver. Text is well suited to these abstract factors, but the evidence behind a good trajectory is also spatial and temporal. A compact textual CoT cannot readily specify how surrounding agents, lanes, and obstacles evolve. The visual domain offers a richer spatio-temporal content, and the future provides a natural supervision signal. That means, anticipating how the environment evolves should guide safer driving.

An ideal foresight modality must be relevant to driving, carrying information that affects the action. It has to be easy to obtain as supervision, and remain predictable when generated before planning. The most direct modality is the future RGB frame. It is readily available from driving logs and preserves the visible future scene, however, it requires predicting hundreds of visual tokens per

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

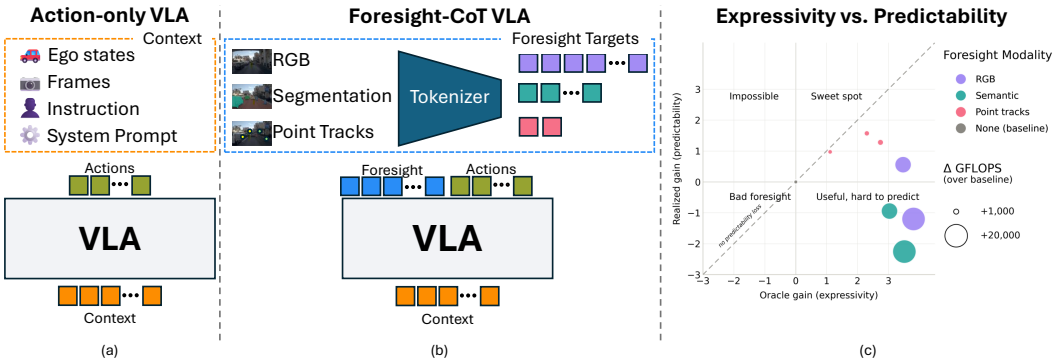


Figure 1: (a) A standard VLA maps the context input directly to an action trajectory. (b) A foresight-CoT VLA first predicts intermediate foresight tokens, which guide it toward more robust planning. (c) We compare dense RGB and semantic-segmentation targets with sparse point-track targets in terms of richness and predictability. Each model is evaluated with ground-truth foresight (oracle) and with its own generated foresight (deployable). Point tracks offer the best balance between the two.

frame and forces the model to synthesize details that may be irrelevant to driving (Shang et al., 2026). And imperfect predictions at inference time can also propagate into errors in the actions conditioned on them. DynVLA reduces the sequence length by inserting compact learned future-dynamics tokens before action generation (Shang et al., 2026); however, in addition to extra training complexity, the learned target is less directly specified by the data, and deciding which information such a compressed representation should retain remains an open problem.

Building on progress in core vision tasks, we study several modalities to find a balance between predictability, richness, and efficiency as illustrated in Fig. 1. We evaluate each modality in two settings: a realistic setting, in which trajectory prediction is conditioned on the model's own forecast, and an oracle setting, in which it is conditioned on the ground-truth future. Comparing the two isolates the information contained in a target from the model's ability to predict it. Pixel-space RGB futures sit at the expressive end of this tradeoff, preserving appearance, layout, and visible detail. Semantic segmentation removes texture and appearance while keeping object and road structure. Compact point tracks replace dense reconstruction with the future locations of selected scene points. This sparse motion signal is less expressive than an image, but it is closer to the quantities that affect short-horizon driving trajectories and substantially cheaper to generate. Pretrained trackers such as CoTracker3 make these point trajectories available without learning a latent dynamics-token space from scratch (Karaev et al., 2025).

We implement TrackForesight on top of an Emu3-8B autoregressive driving VLA (Wang et al., 2024b), following the DriveVLA-W0 setup (Li et al., 2026). The model first generates a visual foresight-CoT and then the final trajectory, giving a common interface for RGB, semantic, and point-track futures. Our experiments show that the best target is not simply the most expressive one: dense RGB and semantic futures contain planning-relevant information when supplied correctly, but their long image-grid sequences make them difficult to use reliably when generated before action. Point-track foresight offers a more favorable balance, exposing short-horizon scene motion with only a small generation overhead and without introducing an additional training phase. On NAVSIM navtest (Dauner et al., 2024), the 20-track, 3 s model reaches 92.1 PDMS after GRPO using only 60 generated foresight tokens, yielding state-of-the-art result among world-model and VLA planners while keeping the intermediate prediction explicit and sparse.

2 Related Work

2.1 Intermediate Structure in Driving Planners

Learning-based planners often predict more than a final ego trajectory. TransFuser uses waypoint prediction with fused sensor features (Chitta et al., 2023); UniAD couples perception, prediction,

and planning through a planning-oriented scene representation (Hu et al., 2023); and VADv2 uses vectorized representations for probabilistic end-to-end planning (Jiang et al., 2026). These systems show that intermediate scene structure can support planning, but the structure is built into the planner architecture. In an autoregressive VLA, the intermediate variable becomes part of the sequence that the policy must generate before action. We study this representation choice on NAVSIM, which evaluates planned ego trajectories with PDM-style driving metrics (Dauner et al., 2024). NAVSIM builds on nuPlan logs and map structure, so it gives a standardized closed-loop planning proxy while retaining large-scale real-world driving context (Caesar et al., 2021).

2.2 Vision-Language-Action Driving Planners

Driving VLA work reuses the autoregressive form of language models for perception, reasoning, and control. DriveGPT4 frames autonomous driving as a vision-language reasoning problem with scene descriptions and interpretable decisions (Xu et al., 2024). DriveVLM similarly uses a vision-language model to connect visual understanding, reasoning, and planning-oriented outputs (Tian et al., 2024). OpenDriveVLA moves the formulation closer to action generation in an open VLA driving stack (Zhou et al., 2026), and AutoVLA studies adaptive reasoning and reinforcement learning for driving VLAs (Zhou et al., 2025). Tokens between observation and action recall chain-of-thought prompting (Wei et al., 2022), but language rationales alone do not specify which visual evidence should be predicted. Recent driving VLAs address this with future-image supervision or learned dynamics tokens: DriveVLA-W0 uses future-image prediction as world-model supervision (Li et al., 2026), and DynVLA inserts learned future-dynamics tokens before action generation (Shang et al., 2026). Our study isolates the target choice: with the backbone, action tokenizer, and decoder fixed, which explicit foresight target should the policy generate?

2.3 Representations for Visual Foresight

Language can be an imperfect medium for this intermediate prediction: chain-of-thought can reduce accuracy on tasks where deliberation also hurts humans (Liu et al., 2025), and vision-language models often underuse the visual channel, at times scoring below their own language backbones on spatial reasoning (Wang et al., 2024a). Work on interleaved multimodal CoT makes a related argument from the modeling side: ThinkMorph (Gu et al., 2026) treats verbal and visual thoughts as complementary rather than isomorphic and trains on traces with manipulated images, while AdaReasoner (Song et al., 2026) learns *when* to reason visually by tuning visual tool choice and ordering against end-task success.

In driving, future scene prediction can be supervised in several visual spaces, each with a different balance between information content and predictability. RGB futures preserve appearance and layout, but not every pixel is equally relevant to control. Semantic segmentation removes texture while retaining object and road layout; our semantic targets are extracted with SAM3 (Carion et al., 2026). Motion representations make a different choice. RAFT estimates dense optical flow through recurrent field updates (Teed & Deng, 2020), while MemFlow adds memory to improve video optical-flow estimation over time (Dong & Fu, 2024). TAPIR tracks arbitrary image points across video (Doersch et al., 2023), and CoTracker3 provides a scalable point-tracking model used here to generate sparse motion supervision (Karaev et al., 2025). Recent work also studies point-trajectory forecasting directly (Zhang et al., 2026). We use point tracks as compact foresight tokens for planning rather than as a tracking benchmark output. Comparing pixel, semantic, and sparse-motion targets under the same VLA lets us separate the information available in a target from the benefit that remains when the model has to generate it.

3 Method

TrackForesight is a foresight-CoT autoregressive driving VLA. The model generates an explicit prediction of the future scene and conditions its action on that prediction. In our main instantiation, this intermediate prediction is a sparse set of point tracks: previous and current point locations appear in the context, and the model completes their future locations as visual foresight-CoT. The resulting trajectory tokens are decoded into continuous ego-frame poses with a fixed action decoder. We first define the foresight-CoT planner, then describe the target spaces used to instantiate the intermediate

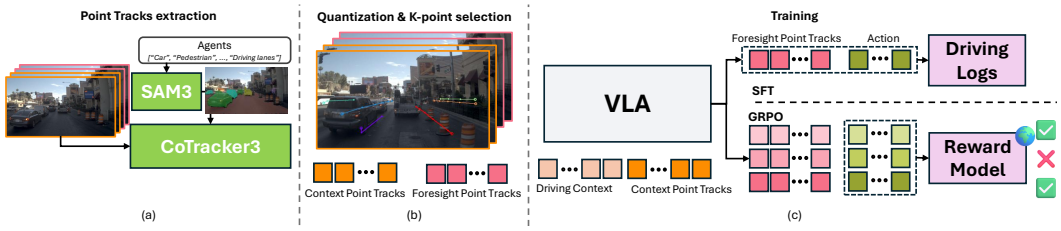


Figure 2: Foresight-CoT driving VLA. (a) We segment a context frame and initialize point tracks on the relevant objects. (b) We track these points through the context frames and the future frames. (c) The context tracks are fed to the VLA together with the raw camera frames, route, navigation command, and ego state. The VLA first predicts the future tracks as a foresight CoT and then emits the action tokens, which are decoded into an ego trajectory. Training proceeds in two stages. During SFT, the same cross-entropy loss is applied to both the foresight-CoT tokens and the action tokens. During GRPO, each rollout samples both the foresight CoT and the action tokens, but the policy-gradient objective is applied only to the action tokens.

prediction, including point tracks, RGB, and semantics, and finally give the SFT/GRPO training objectives.

3.1 Driving VLA with Foresight-CoT

Let c_t denote the driving context encoded as an autoregressive token prefix: camera history, route context, a navigation command, and ego-state history. Camera observations are represented as visual tokens, while route, command, and ego-motion values are appended as tokens in the same sequence. The model generates a future ego trajectory $a_{t:t+H}$ through discrete action tokens, where H is the pallning horizon. An action-only VLA parameterizes a conditional policy over future trajectories:

$$p_\theta(a_{t:t+H} \mid c_t). \tag{1}$$

A foresight-CoT planner augments this policy with a foresight target sequence $z_{t:t+M}$ at foresight horizon M. The policy first predicts this future scene representation and then generates the action tokens conditioned on it:

$$p_\theta(z_{t:t+M}, a_{t:t+H} \mid c_t) = p_\theta(z_{t:t+M} \mid c_t) p_\theta(a_{t:t+H} \mid c_t, z_{t:t+M}). \tag{2}$$

We are interested in foresight-CoT represented in visual domain because of the rich spatio-temporal content relevant to driving. Specifically, we analyze three representations that justifies the rational behind our design choices.

RGB. The pixel-space target uses the future camera frame itself. This target preserves road layout, agents, background context, and appearance changes. It is also the least selective target. Each future frame occupies an 18×32 image-token grid, or 576 generated tokens, much of which may be irrelevant to the ego trajectory.

Semantic. The semantic target keeps the image-grid structure but removes texture and photometric detail. We extract semantic masks from future frames with SAM3, using driving-relevant classes including vehicles, vulnerable road users, road surface, markings, curbs, cones, and barriers (Carion et al., 2026). The resulting target preserves object and road layout while discarding appearance variation. It has the same 576-token-per-frame cost as RGB, and tests whether an abstract dense target is easier to use than raw pixels.

Point-track. We are particalaey intetested in point-tracks as motion is a strong prior for driving, and point-tracks concentrates the prediction budget on motion. We sample object-aware query points from SAM3-derived masks and track them through the context as well as the future clip with CoTracker3 (Karaev et al., 2025). Each track is initialized in the context prefix with its previous

and current image locations; the model then autoregressively completes the track by generating its future locations at the target foresight horizons. This generated continuation is the foresight-CoT for point tracks. It gives the model a compact motion target before action generation, replacing dense appearance reconstruction with the predicted motion of the selected scene points.

3.2 TRAINING

We follow a two-stage training pipeline: supervised fine-tuning teaches imitation from logged trajectories and foresight targets, and GRPO refines the policy using trajectory-level feedback on safety, progress, and comfort derived from Predictive Driver Model Score (PDMS) reward.

Supervised Fine-Tuning. Each SFT training sample consists of the driving prefix c_t, the future ego trajectory to be predicted, and for foresight-CoT variants, the future scene target derived from the same log. Let $y = [y_{t+1}, \ldots, y_{t+L}]$ denote the supervised completion after c_t. For the action-only baseline, the completion contains only the action span, $y = [a_{t:t+H}]$. For a foresight-CoT model, the completion is ordered as

$$y = [z_{t:t+M}; a_{t:t+H}], \tag{3}$$

Delimiter tokens mark the boundary between the foresight-CoT sequence and the action span. We train with a masked autoregressive loss:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\frac{1}{\sum_i \alpha_i} \sum_{i=1}^{L} \alpha_i \log p_\theta(y_i \mid c_t, y_{<i}), \tag{4}$$

Where α_i can be different for CoT and trajectory completion. Emu3 visual tokens of future frames and SAM3 mask images supervised the RGB and semantic foresight-CoT targets respectively. The point-track locations in the future frames supervise the point-track targets where previous and current point locations are placed in the prefix to identify the tracks in the context. The action span is supervised in every variant.

RL Post-Training SFT learns from logged actions through token-level imitation, but good driving also depends on trajectory-level properties such as collision avoidance, drivable-area compliance, progress, and comfort. Following the common practice, we therfore post-train selected SFT checkpoints with GRPO (Shao et al., 2024). Because dense RGB and semantic foresight require hundreds of generated tokens per future frame, the main GRPO study focuses on point tracks, whose completion length is comparable to the action-token span. For each driving prefix c_t, the policy samples a group of G completions. Each completion has a generated foresight part $y_z^{(g)}$ followed by an action part $y_a^{(g)}$, matching the ordering used during SFT:

$$y^{(g)} = [y_z^{(g)}; y_a^{(g)}] = [z_{t:t+M}^{(g)}; a_{t:t+H}^{(g)}]. \tag{5}$$

Thus, each group member includes both sampled point-track foresight tokens and sampled action tokens. We parse only the action suffix $y_a^{(g)}$, and compute a PDM reward from safety, compliance, progress, and comfort terms (Dauner et al., 2024);

$$r^{(g)} = R(a_{t:t+H}^{(g)}). \tag{6}$$

This scalar reward measures the planning behavior induced by the sampled completion. The group-relative advantage is

$$A^{(g)} = r^{(g)} - \frac{1}{G} \sum_{j=1}^{G} r^{(j)}. \tag{7}$$

Although the sampled foresight tokens remain in the autoregressive context for the action, we exclude them from the RL loss and update only the action tokens. More specifically, Let $\mathcal{M}^{(g)} = \left[\mathbf{0}_P;\ \mathbf{1}_H\right] \in \{0,1\}^{|y^{(g)}|}$ be the token mask for rollout $y^{(g)}$, $\mathcal{I}^{(g)} = \{\, i \mid \mathcal{M}_i^{(g)} = 1 \,\}$ the set of supervised positions, π_{old} be the rollout policy, and π_{ref} denote the SFT reference policy. The GRPO update minimizes

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) = -\frac{1}{G} \sum_{g=1}^{G} \frac{1}{|\mathcal{I}^{(g)}|} \sum_{i \in \mathcal{I}^{(g)}} \left[\min\left(\rho_i^{(g)} A^{(g)},\ \mathrm{clip}(\rho_i^{(g)}, 1-\epsilon, 1+\epsilon)\, A^{(g)}\right) - \beta\, \mathbb{D}_{\mathrm{KL}}^{(g,i)}\left[\pi_\theta \| \pi_{\mathrm{ref}}\right] \right], \tag{8}$$

where $\rho_i^{(g)} = \frac{\pi_\theta(y_i^{(g)}|c_t, y_{<i}^{(g)})}{\pi_{\text{old}}(y_i^{(g)}|c_t, y_{<i}^{(g)})}$. We tried two more extreme design choices of excluding the tracks from the rollouts, i.e. $y^{(g)} = [y_a^{(g)}], \mathcal{M}^{(g)} = [\mathbf{1}_H]$, and also including them in the PDMS reward, i.e. $y^{(g)} = [y_z^{(g)}; y_a^{(g)}], \mathcal{M}^{(g)} = [\mathbf{1}_{M+H}]$, and found the proposed setup yields the best performance. In the main GRPO runs, the sampled completion follows the same grammar as SFT, but the scalar reward is assigned only to the action suffix. The point-track continuation affects the update by conditioning the action distribution rather than by receiving direct RL gradients. Fig. 2 illustrates the components of our method and training details.

4 Experiments

Data and task. We use NAVSIM (Dauner et al., 2024) as our primary benchmark, as it evaluates planned ego trajectories with simulation-based driving metrics across diverse, realistic scenarios. All models are trained on the NAVSIM trainval split and evaluated on the navtest split, both derived from nuPlan (Caesar et al., 2021). Each example comprises front-camera observations, route context, a textual navigation command, the ego-state history, and the logged future ego trajectory. The policy observes the current frame and the frame 1 s in the past, and outputs a 4 s future trajectory. For foresight CoT policies, the intermediate target is extracted from future frames up to 1 s, 2 s, or 3 s ahead of the current frame.

Model and tokenization. All controlled comparisons share the same autoregressive backbone, Emu3-8B (Wang et al., 2024b), and the same FAST action tokenizer (Pertsch et al., 2025). The action-token representation is held fixed; the only experimental variable is the foresight CoT inserted before the action tokens. RGB foresight is encoded with the Emu3 visual tokenizer on an 18×32 grid, yielding 576 tokens per future frame. Semantic foresight uses the same grid applied to SAM3 segmentation maps (Carion et al., 2026). Point tracks are initialized from object-aware query points from SAM3-derived masks and extracted with CoTracker3 (Karaev et al., 2025). We quantize each point location on a 36×64 image grid and use a separate invisible token for points that are occluded or leave the image. A point-track foresight target with K tracks and M horizons contains KM generated location tokens. This design yields a controlled trade-off between expressivity and sequence length: dense targets at 1 s and 2 s require 576 and 1152 foresight CoT tokens, respectively, whereas 20-track targets at 1 s, 2 s, and 3 s require only 20, 40, and 60.

Training. Training proceeds in two stages: supervised fine-tuning (SFT) followed by reward-based post-training with group-relative policy optimization (GRPO) (Shao et al., 2024). During SFT, the model is trained via teacher forcing on the same output order used at inference: foresight tokens, when present, followed by action tokens. SFT runs for 4,300 steps on eight A100 GPUs in bf16 precision, with 100 warmup steps and cosine learning-rate decay. GRPO is initialized from the corresponding SFT checkpoint and runs for 6,000 steps. Each update samples four completions per prompt, decodes the action suffix of each, and assigns a reward based on the safety, progress, and comfort scores of the resulting trajectory (Dauner et al., 2024).

4.1 Planning evaluation

We evaluate planners with the PDM score (PDMS) on a 0–100 scale and, where available, report its standard sub-scores: no at-fault collision (NC), drivable-area compliance (DAC), driving-direction compliance (DDC), time-to-collision (TTC), ego progress (EP), and comfort (Cmf.) (Dauner et al., 2024). In this section, we compare TrackForesight, using 20 point tracks per frame and a 3 s foresight horizon, against prior work; Table 4 ablates the number of tracks and the horizon. For fair comparison, we only compare our methods with the VLA based planners.

NAVSIM v1. Table 1 compares TrackForesight with state-of-the-art methods on the 12,146 valid navtest scenes of NAVSIM v1. For a more transparent comparison, we group methods into three families: planning and sensor-fusion baselines; generative, world-model, and preference-tuned planners; and VLA planners. TrackForesight achieves a PDMS of 92.1, outperforming all prior methods, including the strongest published VLA planner, DynVLA (Shang et al., 2026) (91.7), and our reproduction of it under the same scoring pipeline (91.3), while keeping its intermediate prediction

Table 1: Comparison with prior work on NAVSIM v1 navtest. Published baselines differ in architecture, sensors, pretraining, and evaluation protocol. Gray: published result; "repro.": reproduced under our scoring pipeline. DDC is reported for our runs only, as prior work does not report it consistently.

Method	Family	Safety and compliance				Driving quality		PDMS
		NC	DAC	DDC	TTC	EP	Cmf.	
Human (Dauner et al., 2024)	Reference	100.0	100.0	–	100.0	87.5	99.9	94.8
Planning and sensor-fusion baselines								
VADv2 (Jiang et al., 2026)	Planning E2E	97.2	89.1	–	91.6	76.0	100.0	80.9
UniAD (Hu et al., 2023)	Planning E2E	97.8	91.9	–	92.9	78.8	100.0	83.4
TransFuser (Chitta et al., 2023)	Sensor fusion	97.7	92.8	–	92.8	79.2	100.0	84.0
PARA-Drive (Weng et al., 2024)	Planning E2E	97.9	92.4	–	93.0	79.3	99.8	84.0
Generative, world-model, and preference-tuned planners								
DiffusionDrive (Liao et al., 2025)	Generative planner	98.2	96.2	–	94.7	82.2	100.0	88.1
WoTE (Li et al., 2025)	BEV world model	98.5	96.8	–	94.9	81.9	99.9	88.3
DriveDPO (Shang et al., 2025)	Preference-tuned planner	98.5	98.1	–	94.8	84.3	99.9	90.0
VLA planners								
DriveVLA-W0 (Li et al., 2026)	VLA planner	98.7	99.1	–	95.3	83.3	99.3	90.2
AutoDrive-R2 (Yuan et al., 2026)	VLA + text reasoning	98.3	94.4	–	95.6	81.6	100.0	90.3
DynVLA (Shang et al., 2026)	VLA planner	98.6	98.7	–	95.5	86.8	100.0	91.7
DynVLA (repro.)	VLA planner	98.4	98.1	97.6	95.0	87.3	100.0	91.3
TrackForesight (ours)	VLA + point tracks	98.4	98.2	96.8	94.9	**89.2**	100.0	**92.1**

Table 2: NAVSIM v2-hard evaluation shell. Entries will be filled after matched scoring uses the same model family, checkpoint-selection rule, and reporting protocol as the navtest experiments.

Method	NC	DAC	TTC	EP	Stage-2	EPDMS
DynVLA (repro., GRPO)	81.1	74.3	**77.6**	93.7	39.6	30.5
Ours (20 tracks, 3 s, GRPO)	**83.0**	**76.1**	77.3	**94.0**	**41.7**	**31.7**

explicit and sparse. Its safety and compliance scores are on par with the strongest published results, its ego progress of 89.2 is better than the human reference of 87.5, and comfort is near saturation for most high-scoring methods.

NAVSIM v2-hard. We further evaluate on the 5.2K pseudo-closed-loop scenarios of the NAVSIM v2 benchmark in Table 2. Here we compare TrackForesight with DynVLA (Shang et al., 2026) because they are reproducible, share our backbone design, and differ from TrackForesight only in the CoT tokens that precede the action. Moreover, they are best performing VLA based planner to our knowledge.

4.2 Ablations

In this section, we first justify our tracking-based visual CoT by analyzing the richness and predictability of different foresight modalities. We then discuss the design choices for the number of point tracks and the foresight horizon, and finally evaluate the quality of the generated point tracks quantitatively.

Richness vs. Predictability. An ideal foresight CoT should satisfy three criteria. First, it should be *rich*: it should contain information that can change the decoded trajectory. Second, it should be *predictable*: the model should forecast it accurately enough for that information to remain useful at inference time. Third, it should be *efficient*: it should add little computational overhead at inference, which for autoregressive models means generating few tokens. To assess richness and predictability, we compare an action-only planner with three foresight-CoT planners that use RGB frames, semantic segmentation, or point tracks as foresight. We evaluate each foresight-CoT planner in two setups. In the *oracle* setup, which serves as a diagnostic, the model receives ground-truth foresight at inference. In the *deployable* setup, the model generates its own foresight before decoding actions.

Table 3: Comparison of foresight targets after the SFT stage on NAVSIM navtest. *Gen.*: the model generates its own foresight before decoding actions (deployable). *Oracle*: ground-truth foresight is provided at inference. *Diff.*: Oracle − Gen.; a smaller gap indicates higher predictability. Rich.: richness; Pred.: predictability; Eff.: efficiency. Point tracks use 20 tracks.

Target	Foresight horizon	#Tok. ↓	PDMS			Rich.	Pred.	Eff.
			Gen. ↑	Oracle ↑	Diff. ↓			
Action-only	–	0	85.30	–	–	–	–	–
RGB	1 s	576	85.86	88.77	+2.91	✓✓	✗	✗
	2 s	1152	84.10	89.11	+5.01			
Semantic	1 s	576	84.36	88.33	+3.97	✓✓	✗	✗
	2 s	1152	83.05	88.81	+5.76			
Point tracks	1 s	20	86.54	86.81	+0.27	✓	✓	✓
	2 s	40	86.62	87.33	+0.71			
	3 s	60	86.57	87.86	+1.29			

High oracle performance indicates high richness, and a small gap between oracle and deployable performance indicates high predictability. We compare all models after the SFT stage, because the large number of tokens that RGB and semantic-segmentation foresight generate put GRPO beyond our computational budget.

As shown in Table 3, RGB frames and semantic segmentation are rich: with ground-truth foresight, they improve PDMS by up to 3.8 points over the action-only planner. However, they are poorly predictable, with oracle–deployable gaps of 2.9–5.8 points. As a result, their deployable gains are marginal or negative. Only RGB at 1 s outperforms the action-only planner, and semantic segmentation falls below it at both horizons. Dense targets also fail the efficiency criterion, since they require 576 tokens per future frame. Point tracks, in contrast, are less rich than dense targets but far more predictable and efficient. Their oracle–deployable gap stays at or below 1.4 points, and they improve over the action-only planner at every horizon while using only 20 tokens per frame. The two kinds of foresight also respond differently to the horizon. Extending dense foresight from 1 s to 2 s lowers deployable performance for both RGB and semantic segmentation, even though their oracle performance improves. Point tracks, on the other hand, benefit from a longer horizon: their performance peaks at 2 s and stays above the action-only planner at 3 s.

Point Selection and Track Budget. To select the number of point tracks and the foresight horizon, we train the model with different combinations of the two and evaluate planning performance for each in Table 4. SFT performance improves as the number of point tracks increases from 8 to 16 and saturates beyond 16, with 20 tracks giving no consistent gain. The foresight horizon follows a similar trend: SFT performance peaks at 2 s and does not improve further at 3 s. Due to our limited computational budget, we apply GRPO only to the six best SFT models. Among them, the configuration with 20 point tracks and a 3 s foresight horizon performs best after GRPO.

Table 4: Point-track development ablation.

Selection variant	Grid	SFT PDMS			GRPO PDMS		
		1 s	2 s	3 s	1 s	2 s	3 s
8 points, early selection	18×32	85.37	85.87	85.60	–	–	–
12 points, motion-ranked	36×64	85.34	85.65	85.44	–	–	–
16 points, smooth motion-ranked	36×64	86.27	**86.87**	86.58	91.52	91.78	92.00
20 points, final tokenizer	36×64	86.54	86.62	86.57	91.77	91.78	**92.10**

Point-Track Foresight Quality. Table 4 measures whether the generated point-track foresight improves the final trajectory, not whether the foresight itself is accurate. To assess prediction quality directly, we evaluate the generated tracks in their native token space. For each future step, each of the K points is predicted either as a quantized cell on a 36×64 grid or as an invisible state, and

we compare the generated tokens with the ground truth using three metrics. *Vis.* is the visibility-prediction accuracy (visible vs. invisible), independent of location. *Coord@1* is the fraction of points that lie within one grid cell (Chebyshev distance) of the ground-truth cell, computed over points where the prediction and ground truth are both visible. *ADE* and *FDE* convert the predicted cells to pixel coordinates and report the Euclidean error averaged over all future steps and at the final step, respectively.

As shown in Table 5, visibility accuracy remains stable across horizons (76–78%), whereas localization degrades steadily with lookahead. For 20 tracks, Coord@1 drops from 76.9% at 1 s to 63.9% at 3 s, and FDE increases about 2.5× (from 5.31 to 13.20 px). Predicting whether a point remains visible is therefore easier than predicting where it will be, and localization error accumulates over the horizon. Using 20 tracks is slightly more accurate at 1 s, but the error at 3 s matches that of 16 tracks. A larger track budget therefore does not yield proportionally better predictions.

Table 5: Quality of the generated point-track foresight, evaluated independently of trajectory PDMS. Coord@1: fraction of visible points within one grid cell of the ground truth. ADE / FDE: average / final pixel displacement error. Vis.: visibility-prediction accuracy.

Tracks	Lookahead	Coord@1 (%)↑	ADE / FDE (px)↓	Vis. (%)↑
16	1 s	76.9	5.31 / 5.31	76.7
	2 s	70.1	6.33 / 8.38	77.5
	3 s	63.9	8.08 / 13.20	76.7
20	1 s	78.6	4.87 / 4.87	78.4
	2 s	70.6	6.18 / 8.16	78.4
	3 s	63.9	8.05 / 13.20	77.3

5 Conclusion

In this work, we proposed TrackForesight, a tracking-based visual foresight CoT for automotive VLAs. We studied several foresight modalities and showed that dense RGB and semantic futures carry planning-relevant information when provided as oracle foresight, but do not reliably improve planning when the model must generate them before acting. In contrast, sparse point tracks offer a favorable balance of richness, predictability, and efficiency: they represent short-horizon motion with far fewer tokens and remain effective when optimized with a planning reward. TrackForesight achieves a state-of-the-art PDMS of 92.1 among VLA-based planners while generating only 60 foresight tokens. We believe these findings help advance the recent line of VLAs that use visual CoT.

Limitations

The main limitation of our approach is its reliance on auxiliary computer vision models. Specifically, we use a segmentation model to initialize the point tracks and a point tracker to obtain their trajectories in the context and future frames. Although segmentation and tracking have progressed rapidly, these models are not perfectly accurate. A segmentation failure can cause a critical scene element to be missed, and a tracking failure can produce incorrect context tracks at inference or noisy supervision during training. These models also add computational overhead to the pipeline. We expect both limitations to diminish as segmentation and tracking models continue to improve in accuracy and efficiency, and in our experiments the added cost is small relative to the performance gain.

AI use statement

We used generative AI tools to assist with implementing code, analyzing data, and interpreting results, and for grammar correction. The authors reviewed all AI-assisted work: code was tested before use, and every reported number and interpretation was checked by the authors. We take responsibility for the final content of this work.

Ethics statement

This work uses the public NAVSIM benchmark, built on nuPlan driving logs, under its license terms; it collects no new data and involves no human subjects. TrackForesight is a research study evaluated in simulation, and benchmark scores alone do not establish on-road safety. Like any learned driving policy, it would require extensive closed-loop and real-world validation before deployment. The explicit point-track foresight makes part of the model's intermediate reasoning inspectable, which we see as a step toward more transparent driving policies.

Reproducibility statement

Section 3.2 specifies the SFT and GRPO objectives, and Section 4 describes the data, the foresight targets and their tokenization, and the training schedule. Table 4 reports how the track budget and foresight horizon were chosen. All components we build on are publicly available, including the Emu3 backbone, the FAST action tokenizer, SAM3, and CoTracker3, and all results use the public NAVSIM benchmark splits.

References

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. SAM 3: Segment anything with concepts. In *International Conference on Learning Representations*, 2026.

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, 2023. doi: 10.1109/TPAMI.2022.3200245.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 37, pp. 28706–28719, 2024.

Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10061–10072, 2023.

Qiaole Dong and Yanwei Fu. MemFlow: Optical flow estimation and prediction with memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19068–19078, 2024.

Jiawei Gu, Yunzhuo Hao, Huichen Wang, Linjie Li, Michael Qizhe Shieh, Yejin Choi, Ranjay Krishna, and Yu Cheng. ThinkMorph: Emergent properties in multimodal interleaved chain-of-thought reasoning. In *International Conference on Learning Representations*, 2026.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17853–17862, 2023.

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*, 2026.

Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6013–6022, 2025.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 27137–27146, 2025.

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, et al. DriveVLA-W0: World models amplify data scaling law in autonomous driving. In *International Conference on Learning Representations*, 2026.

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

Ryan Liu, Jiayi Geng, Addison J. Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L. Griffiths. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 38489–38517, 2025.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems (RSS)*, 2025.

Shuyao Shang, Yuntao Chen, Yuqi Wang, Yingyan Li, and Zhaoxiang Zhang. DriveDPO: Policy learning via safety DPO for end-to-end autonomous driving. In *Advances in Neural Information Processing Systems*, volume 38, pp. 81565–81585, 2025.

Shuyao Shang, Bing Zhan, Yunfei Yan, Yuqi Wang, Yingyan Li, Yasong An, Xiaoman Wang, Jierui Liu, Lu Hou, Lue Fan, et al. DynVLA: Learning world dynamics for action reasoning in autonomous driving. *arXiv preprint arXiv:2603.11041*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Mingyang Song, Haoyu Sun, Jiawei Gu, Linjie Li, Luxin Xu, Ranjay Krishna, and Yu Cheng. AdaReasoner: Dynamic tool orchestration for iterative visual reasoning. In *International Conference on Learning Representations*, 2026.

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, pp. 402–419, 2020.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. DriveVLM: The convergence of autonomous driving and large vision-language models. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4698–4726, 2024.

Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Yixuan Li, and Neel Joshi. Is a picture worth a thousand words? delving into spatial reasoning for vision language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 75392–75421, 2024a.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

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

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.

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

Jianing Zhang, Chenhao Zheng, Yajun Yang, Max Argus, Rustin Soraki, Winson Han, Taira Anderson, Chun-Liang Li, Shuo Liu, Jiafei Duan, Zhongzheng Ren, Jieyu Zhang, and Ranjay Krishna. MolmoMotion: Forecasting point trajectories in 3D with language instruction. *arXiv preprint arXiv:2606.18558*, 2026.

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

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. In *Advances in Neural Information Processing Systems*, volume 38, pp. 27920–27956, 2025.