LEARNING NONLINGUISTIC DELIBERATION THROUGH LATENT CRITIQUES

Anonymous authors
Paper under double-blind review

ABSTRACT

Planners often use a learned scorer to select among candidate actions, then discard its assessment. We ask whether this assessment can instead guide revision, and what the critic should communicate to make revision useful. We introduce ReCoursive, a recurrent planner for end-to-end driving that deliberates without language. Its critic returns *latent critiques*, the continuous representations from which it predicts scores, to guide the proposer's subsequent revisions. The loop is trained end to end with trajectory and assessment supervision at each step, without reasoning traces or demonstrated corrections. On NAVSIMv2, critic feedback adds 2.83 EPDMS over recurrence alone at matched depth and proposal count. Additional revision steps improve planning with latent critiques but not with predicted scores alone; critiques also need assessment supervision and must be refreshed as candidates change. ReCoursive achieves the best results among compared methods on NAVSIMv2 and, in zero-shot transfer, on HUGSIM and among non-VLM methods on KITScenes LongTail. These results suggest that nonlinguistic deliberation benefits from a learned interface between assessment and generation rather than one restricted to scores.

1 INTRODUCTION

The first plan should be a proposal, not a verdict.

The benefits of deliberation need not be tied to language. Language models answer better when they reason and verify intermediate steps (Wei et al., 2022; Guo et al., 2025) or critique and refine their outputs (Madaan et al., 2023), and such reasoning has since been adapted to driving (Wang et al., 2024; Yuan et al., 2026). Yet the pattern behind these gains, assessing intermediate results and revising them, does not in principle require language (Redish, 2016). Text also adds latency that real-time systems can ill afford. We therefore study nonlinguistic deliberation.

By deliberation, we mean a *propose–assess–revise* process. In this process, assessments of the current candidates condition the next candidates before an action is chosen. It is *nonlinguistic* when these candidates and assessments are trajectories and continuous representations rather than text, and *learned* when the revision rule is a trained network rather than a hand-designed search procedure. Under this definition, choosing among assessed candidates is not yet deliberation, and neither is revising without an assessment.

Scorer-based driving planners propose and assess, but stop short of revision. A generator proposes candidate trajectories, and a learned scorer predicts scores for collision avoidance, drivable-area compliance, and progress (Li et al., 2025b; Kirby et al., 2026). The assessment is used only to select among existing candidates and never reaches the generator. If every candidate is flawed, the planner can only pick the least flawed one.

Revision itself is well studied; the interface between assessment and revision is not. Imagination-based agents feed predicted outcomes of proposed actions back into a learned controller (Hamrick et al., 2017; Pascanu et al., 2017). Deliberation networks, which use the term *deliberation* more broadly, refine a first-pass draft with a second decoder but without an explicit assessment (Xia et al., 2017). Latent reasoning replaces text with continuous states, both in language models and in driving (Hao et al., 2025; Tan et al., 2026). Driving planners refine trajectories itera-

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.

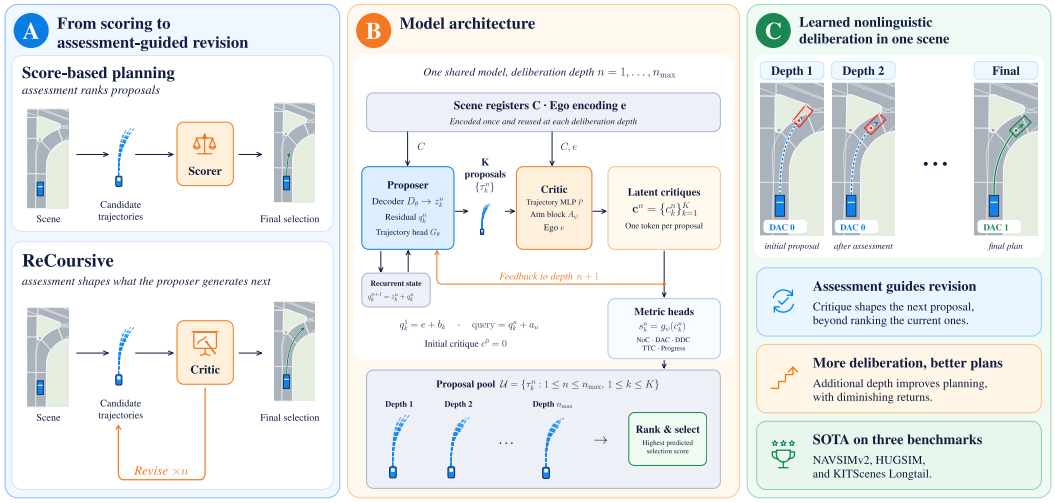

Figure 1: **Overview of ReCoursive.** (A) Extending score-based selection with assessment-guided revision. (B) Latent critiques guide the recurrent proposer, while metric heads score candidates for selection. (C) Successive revisions, with drivable-area violations boxed in red.

tively (Guo et al., 2026; Yin et al., 2026), search using rewards or scorer outputs (Yang et al., 2024; Xu et al., 2026), or edit them with an external safety oracle (Li et al., 2025a). These works leave open what should cross from assessment to revision. When a planner already has a learned scorer, should the scorer pass its predicted scores to the generator, or its internal representation? And should it be trained only to assess, or also for how its assessment is used? Analogous questions have been studied for language feedback (Madaan et al., 2023; Chen et al., 2024; Akyürek et al., 2023), but to our knowledge not for nonlinguistic feedback.

ReCoursive turns the scorer into a critic whose latent critiques condition the next revision. At each deliberation step, the proposer generates candidate trajectories and the critic assesses them (Figure 1). The critic's metric heads rank candidates for final selection, while its continuous representation of each candidate, the latent critique, returns to the proposer. Each module reuses its weights across steps, so deliberation adds computation but no parameters.

We specify where critiques flow; the model learns what they carry and how to use them. The unrolled loop is trained with trajectory and assessment supervision at every step, without reasoning traces or demonstrated corrections. Unlike a scorer, which only learns to judge, the critic also learns to advise: later planning losses reach it through its critiques.

Critic feedback improves planning beyond recurrence alone, and its form matters for revision. Against a recurrent control without feedback that generates the same number of candidates, ReCoursive scores 2.83 Extended Predictive Driver Model Score (EPDMS) higher on NAVSIMv2 at six deliberation steps. From one to six steps, explicit-score feedback yields no gain (-0.49 EPDMS), whereas latent critiques improve EPDMS by 4.49. Intuitively, scores summarize candidate quality, while latent critiques can retain richer information for revision. Deliberation remains cheap: increasing from one to four steps adds only 8.4 ms of latency.

Our contributions are:

- **Deliberation through latent critiques.** We identify the interface between a learned critic and a generator as a design choice for learned planning. We introduce ReCoursive, whose critic, trained inside the loop, conditions a weight-shared proposer through latent critiques rather than scores.
- **Characterizing learned deliberation.** Matched recurrent controls and feedback ablations show that critiques improve planning beyond recurrence alone and help most when they are latent, supervised, and fresh. Depth and runtime analyses quantify the diminishing returns and cost of additional deliberation.

- **State-of-the-art performance.** ReCoursive achieves the best EPDMS on NAVSIMv2. Without target-domain fine-tuning, it achieves the highest average RC and HD-Score on HUGSIM, and the highest MMS among non-VLM methods on KITScenes LongTail.

2 Related Work

We organize prior work by whether and how assessments guide revision, highlighting the closest connections to ReCoursive. Table 5 in Appendix A summarizes these comparisons.

2.1 Scorer-based End-to-End Planners

Scorer-based planning has moved from fixed trajectory vocabularies to generated proposals and richer evaluators. Hydra-MDP distills simulator sub-scores into prediction heads that rate every trajectory in a fixed vocabulary, and combines them with imitation scores for selection (Li et al., 2024). GTRS extends this scorer lineage toward dynamic proposals by combining a diffusion generator with a scorer trained to generalize from dense vocabularies to smaller proposal sets (Li et al., 2025c). WoTE instead uses a BEV world model to forecast the future state induced by a candidate and evaluate that trajectory online (Li et al., 2025b). DrivoR uses separate generation and scoring decoders on compact camera-aware registers; its scorer imitates an oracle and predicts interpretable safety, comfort, and efficiency sub-scores (Kirby et al., 2026).

ReCoursive builds on this scorer lineage and feeds the critic's representation back into generation. We adopt DrivoR's register-based perception and comparable metric heads to rank candidates for final selection as in the methods above. Unlike them, the latent critique of each candidate also returns to the proposer, and later planning losses shape the critic through that path.

2.2 Revision in Driving: Refinement, Search, and Editing

Planners revise candidates using their own features or confidence, scalar scores, or external checks. iPad refines proposals and their features through proposal-anchored attention (Guo et al., 2026), and DiffRefiner applies a scene-conditioned iterative diffusion refiner to coarse proposals (Yin et al., 2026). ReflexDiffusion amplifies critical conditioning signals within denoising through gradient-based adjustments (Yao et al., 2026). Diffusion-ES alternates black-box reward evaluation with evolutionary selection and diffuse–denoise mutations (Yang et al., 2024). TOAD treats DrivoR's scorer as a learned reward and maximizes it at test time with the cross-entropy method, warm-started from the planner's proposals (Xu et al., 2026). ReflectDrive uses local search to find unsafe tokens in a goal-conditioned discrete trajectory and feasible replacements, which anchor an inpainting-based regeneration (Li et al., 2025a). ReflectDrive-2 learns token-level self-editing with the same masked-diffusion model that drafts; editing is pretrained on structured trajectory perturbations and coupled to drafting through reinforcement learning (Wang et al., 2026).

ReCoursive differs from the closest driving methods in what passes from assessment to revision. TOAD shares our starting point that the scorer should shape the candidates rather than only select among them, but passes its scalar scores to a hand-designed search. ReflectDrive-2 revises its drafts using its own token confidence. Even with a learned critic and proposer, the feedback representation matters: in our ablation, additional revision improves planning with latent critiques but not with predicted metric scores (Section 4.3).

2.3 Deliberation in Language and Latent Space

Language-based self-refinement studies how to produce useful feedback for revision. Self-Refine prompts a single model to generate, critique, and revise an output, and finds specific, actionable feedback more useful than generic feedback (Madaan et al., 2023). Self-Debugging shows that execution feedback and natural-language code explanations can improve correction beyond correctness-only feedback (Chen et al., 2024). RL4F trains a critique generator using improvements in a fixed reviser's outputs as rewards (Akyürek et al., 2023). Driving systems also explicitly supervise intermediate reasoning: DriveCoT predicts rule-generated driving aspects and final decisions (Wang et al., 2024), while AutoDrive-R^2 trains a vision-language-action model on a four-

stage reasoning chain containing self-reflection, followed by reinforcement learning with physics-grounded rewards (Yuan et al., 2026).

Nonlinguistic planning and refinement already use learned iterative update mechanisms. Imagination-based agents aggregate proposed actions, predicted outcomes, and rewards into a learned memory that conditions subsequent proposals, while learning how much computation to allocate before acting (Hamrick et al., 2017; Pascanu et al., 2017). MCTSnets learn search operations over vector embeddings (Guez et al., 2018), and model-free recurrent agents can develop internal plans that benefit from additional computation (Bush et al., 2025). Outside action planning, RAFT updates optical flow using estimate-dependent correlation lookups, shared update weights, and supervision across iterations (Teed & Deng, 2020). Iterative amortized inference instead learns posterior updates from gradients of its variational objective (Marino et al., 2018).

Latent reasoning provides continuous intermediate computation without expressing each step in language. Coconut feeds hidden states back as inputs, using a training curriculum that progressively replaces language reasoning steps with continuous states (Hao et al., 2025). Recurrent-depth models iterate a shared latent block to increase test-time computation without generating additional intermediate tokens (Geiping et al., 2025). In driving, LCDrive interleaves action-proposal tokens with learned world-model tokens representing their predicted consequences, followed by reinforcement-learning post-training (Tan et al., 2026).

ReCoursive passes a different signal: a latent critique of each candidate, supervised to assess and trained through its use. Unlike language-based critics, it generates no text and requires no reasoning traces. Unlike the nonlinguistic loops above, which pass imagined outcomes, search embeddings, matching costs, gradients, or latent states without an assessment target, its feedback is grounded by assessment supervision.

3 Method

3.1 Problem Formulation and Overview

ReCoursive generates K candidate ego trajectories through a recurrent proposer–critic loop. Given multi-view observations o and ego state x, the proposer generates $\mathcal{T}^n = \{\tau_k^n\}_{k=1}^K$ at depth $n \le n_{\max}$, with $\tau_k^n \in \mathbb{R}^{H\times 3}$ encoding H future poses: planar position (p_x, p_y) and heading θ; k indexes candidates. The critic returns latent critiques $\mathbf{c}^n = \{c_k^n\}_{k=1}^K$ to condition the next revision:

$$o \longrightarrow \mathcal{T}^1 \longrightarrow \mathbf{c}^1 \longrightarrow \mathcal{T}^2 \longrightarrow \cdots \longrightarrow \mathcal{T}^{n_{\max}}. \tag{1}$$

Each propose–assess–revise transition is a *deliberation step*, up to maximum depth $n_{\max}$. The proposer carries a recurrent state; both modules share weights across steps and access the same scene features. Trajectory and assessment losses train the unrolled loop, while metric outputs rank all generated candidates for final selection.

3.2 Trajectory Proposer

The proposer uses DrivoR's camera-register and ego-state representation (Kirby et al., 2026). The image encoder E_ϕ, with parameters ϕ, produces N scene registers $C \in \mathbb{R}^{N\times d}$; d is the hidden width. The ego encoder produces $e = E_{\text{ego}}(x) \in \mathbb{R}^d$. Each slot k has a learned embedding b_k, and its initial proposal state is $q_k^1 = e + b_k$. Slots have distinct initializations but share proposer and critic parameters.

A shared residual update carries each proposal state across deliberation steps. Let $a_n \in \mathbb{R}^d$ encode depth sinusoidally, $c_k^0 = \mathbf{0}$, and $[\,;\,]$ denote token concatenation. The decoder D_ϑ and output head G_ϑ share parameters ϑ across steps, z_k^n is the decoder feature and $\bar{z}_k^n$ its residual update:

$$\begin{aligned} z_k^n &= D_\vartheta\big(q_k^n + a_n, [C; \mathbf{c}^{n-1}]\big), & \bar{z}_k^n &= z_k^n + q_k^n, \\ \tau_k^n &= G_\vartheta(\bar{z}_k^n), & q_k^{n+1} &= \bar{z}_k^n. \end{aligned} \tag{2}$$

The next step receives state q_k^{n+1} and current critiques, without earlier proposals.

3.3 TRAJECTORY CRITIC

The critic evaluates trajectory geometry against the same scene representation used by the proposer. For every proposal, a learned trajectory embedder P maps its geometry to a critic query. The critic-attention block A_ψ, parameterized by ψ, combines self-attention among proposal queries, cross-attention to the same scene registers, and a feed-forward network. Its slot-k output, with the ego embedding added, gives the continuous pre-head feature

$$c_k^n = \left[A_\psi\left(\{P(\tau_j^n)\}_{j=1}^K, C\right)\right]_k + e \in \mathbb{R}^d, \tag{3}$$

Here j ranges over candidates, $[\,\cdot\,]_k$ selects slot k, and c_k^n is its latent critique. Heads $g_{\psi,m}$, indexed by scoring objective m, map it to logits $s_{k,m}^n = g_{\psi,m}(c_k^n)$. For NAVSIM, we instantiate six metric-derived heads: no-at-fault collision (NoC), drivable-area compliance (DAC), driving-direction compliance (DDC), time-to-collision (TTC), ego progress (EP), and comfort (COMF).

The proposer receives latent critiques, not metric probabilities. Each c_k^n enters the next decoder's key–value set as a distinct trajectory-conditioned token, without projection, discretization, or scene-level pooling. This allows feedback to retain information that individual scalar heads need not expose. The K proposals form B branches of K/B; feedback attention stays within each branch (Appendix B). Thus, the critic's heads assess candidates for selection, while its latent critiques condition their revision.

3.4 TRAINING OBJECTIVES

Trajectory and assessment supervision are applied at every deliberation step. Let τ^* be the demonstrated trajectory and Q_m the geometry-based target function, giving $y_{k,m}^n = Q_m(o, \tau_k^n)$. For the standard recipe, which supervises the proposal head of the last decoder layer,

$$\mathcal{L}_{\text{traj}}^n = \min_k \frac{1}{H} \sum_{h=1}^{H} \|\tau_{k,h}^n - \tau_h^*\|_1, \tag{4}$$

$$\mathcal{L}_{\text{score}}^n = \frac{1}{K} \sum_{k=1}^{K} \sum_{m \in \mathcal{M}_{\text{train}}} \text{BCE}(s_{k,m}^n, y_{k,m}^n), \tag{5}$$

$$\mathcal{L}_{\text{on}} = \sum_{n=1}^{n_{\max}} (\mathcal{L}_{\text{traj}}^n + \mathcal{L}_{\text{score}}^n). \tag{6}$$

Here h indexes future poses, $\|\cdot\|_1$ sums absolute coordinate errors, and BCE is binary cross-entropy on logits. $\mathcal{L}_{\text{on}}$ sums trajectory loss $\mathcal{L}_{\text{traj}}^n$ and score loss $\mathcal{L}_{\text{score}}^n$ on generated candidates. $\mathcal{M}_{\text{train}}$ lists supervised objectives. Losses are averaged over samples. Appendix B.5 gives the general layer-wise weighting and loss-reduction details.

A lightweight geometry-based proxy replaces simulator rollout during critic training. We compute targets directly from scene metric caches instead of DrivoR's CPU-intensive Predictive Driver Model (PDM) rollouts. Five heads receive supervision at every step, while the COMF loss is masked. On 32,000 fixed training-cache trajectories, binary targets agree with simulator targets in 88–94 % of cases, and training is 1.3–1.6× faster (Appendices B.3 and B.4).

Borrowed ground-truth trajectories broaden assessment supervision beyond the proposer's own proposals. In a distributed batch of $B = 64$ scenes, each scene receives the other $B - 1$ demonstrations as critic-only training candidates. Targets are recomputed in the receiving scene: a valid demonstration elsewhere may violate its lanes, obstacles, or progress requirements. The complete objective is

$$\mathcal{L} = \mathcal{L}_{\text{on}} + \alpha \mathcal{L}_{\text{cf}}, \tag{7}$$

where $\mathcal{L}_{\text{cf}}$ sums per-head BCE averaged over valid receiver–donor pairs and its weight is $\alpha = 1$. Borrowing is absent at inference; its equations and reductions appear in Appendix B.5.

Later trajectory losses train earlier latent critiques. Recurrent proposal states and latent critiques are not detached, so later planning losses reach the critic through feedback during back-propagation through time. Trajectory inputs to the critic and geometry-based targets are detached (Appendix B.5).

3.5 Proposal Selection

Final selection ranks candidates from all deliberation steps, not only the last. Deliberation produces $Kn_{\max}$ candidates. We retain the union $\mathcal{U}^{n_{\max}} = \bigcup_{n=1}^{n_{\max}} \mathcal{T}^n$ rather than require the last step to dominate. For any candidate τ, head m gives logit $s_m(\tau)$ and probability $p_m(\tau) = \sigma(s_m(\tau))$, where σ is sigmoid. The learned PDM-style selection surrogate is

$$R_\lambda(\tau) = \lambda_{\text{NoC}} \log p_{\text{NoC}} + \lambda_{\text{DAC}} \log p_{\text{DAC}} + \lambda_{\text{DDC}} \log p_{\text{DDC}} + \log(\lambda_{\text{TTC}} p_{\text{TTC}} + \lambda_{\text{EP}} p_{\text{EP}} + \lambda_{\text{COMF}} p_{\text{COMF}}), \quad (8)$$

$$\hat{\tau} = \arg \max_{\tau \in \mathcal{U}^{n_{\max}}} R_\lambda(\tau). \quad (9)$$

Here p_m abbreviates $p_m(\tau)$, $\lambda_m \geq 0$ weights metric m, and $\hat{\tau}$ is the selected trajectory. Coefficients are fitted without network updates (Appendix C.5).

4 Experiments

4.1 Experimental Setup

Benchmarks and configurations. We choose three benchmarks to cover complementary evaluation settings. NAVSIMv2 is the primary benchmark (EPDMS on `navhard`); HUGSIM tests zero-shot closed-loop driving in reconstructed scenes, and KITScenes LongTail tests zero-shot open-loop planning in long-tail scenarios. All models train on NAVSIM, with or without SimScale augmentation. Names indicate size (S/L) and maximum training depth $n_{\max}$ (numeric suffix; six when omitted). Inference uses $n = n_{\max}$ unless stated otherwise.

Proposal selection. At inference depth $n \leq n_{\max}$, selection ranks the $64n$ cumulative proposals. NAVSIMv2 coefficients are grid-searched following Hydra-MDP (Li et al., 2024) and fixed per checkpoint across depths (Appendix C.5). HUGSIM and LongTail use uniform coefficients with comfort masked.

4.2 Deliberation Beyond Recurrence

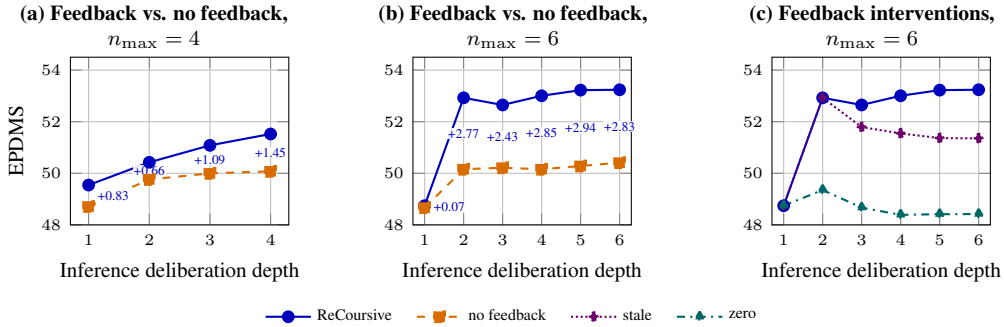

Figure 2: Deliberation beyond recurrence on NAVSIMv2. (a,b) Separately trained feedback/no-feedback models; $n_{\max}$ is the maximum training depth. Labels show ReCoursive minus no feedback (EPDMS points). (c) Feedback interventions on the six-step model.

Critic feedback improves planning beyond recurrence alone. We train a six-step no-feedback control that disables critic feedback while retaining proposal-state backpropagation through time, assessment supervision, and the remaining recipe. Both models generate the same number of proposals at each depth. Their depth-one scores are close (48.74 with feedback; 48.67 without), but at depth six feedback adds 2.83 EPDMS (53.24 versus 50.41; Figure 2b). Independently trained four-step models show the same direction: 51.52 versus 50.08, a 1.45-point gain (Figure 2a).

Critiques must be refreshed as candidates change. On the frozen feedback model, we compare normal recomputed critiques with stale first-step critiques and zeroed critiques. All conditions retain

proposal-state updates and current scores for selection. At depth six, normal, stale, and zero feedback yield 53.24, 51.35, and 48.42 EPDMS (Figure 2c). Normal and stale coincide through depth two, when they use the same initial assessment; their later separation supports refreshing feedback. These inference interventions test the use of learned feedback and are distinct from training a planner without it.

4.3 Learning from Feedback

Latent critiques support revision more effectively than explicit scores. We replace latent feedback with a learned affine projection of the five supervised metric probabilities, with no latent bypass. Feedback gradients, proposal-state backpropagation, per-step assessment supervision, and cross-scene borrowing remain active. From depths one to six, latent critiques gain 4.49 EPDMS (48.74 to 53.24), whereas explicit-score feedback loses 0.49 (50.50 to 50.01; Table 1A). Despite its higher starting score, the score-only interface yields no net improvement from additional steps in this run.

Assessment supervision helps latent critiques become useful. We remove proposal metric losses and the borrowed-trajectory assessment loss, retaining latent feedback, trajectory supervision, and gradients through feedback and proposal states. Since the ablated metric heads are untrained, both planners use one frozen no-feedback critic for final selection while retaining their own feedback during revision. Coefficients are fitted independently per proposer–critic pair under the same protocol. With this common critic, assessment supervision raises depth-six EPDMS from 46.37 to 51.69; the corresponding depth-one-to-six gains are 1.29 and 3.10 (Table 1B).

Table 1: Feedback representation and supervision on NAVSIMv2. Changes use unrounded results. Panel A uses each model's own critic; panel B uses one common frozen no-feedback critic. Absolute scores across panels are not a matched comparison.

Feedback condition	Depth 1	Depth 6	Change
A. Feedback interface			
Latent critiques	48.74	53.24	+4.49
Explicit-score feedback	50.50	50.01	−0.49
B. Feedback training			
With assessment supervision	48.59	51.69	+3.10
Without assessment supervision	45.09	46.37	+1.29

Editing assessment-related latents can change subsequent revision. We hold donor trajectories and the initial proposal state fixed, comparing native donor feedback with feedback edited to invert DAC assessments (Figure 3). In this example, the final proposal's minimum vehicle-corner margin changes from 18.19 cm inside the drivable area to −7.53 cm under edited feedback. Selected NoC/TTC edits similarly introduce front-vehicle collisions (Appendix E.3). These interventions demonstrate an effect on revision, without establishing reliable directional control.

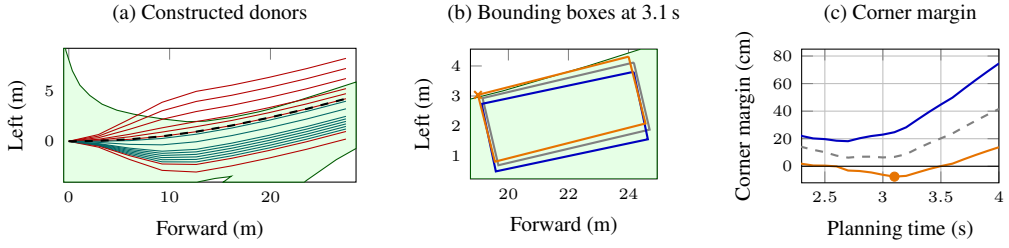

Figure 3: Editing feedback changes revision. (a) Fixed donors: eight DAC-compliant (teal) and eight noncompliant (red); target proposal dashed. Donor-flip inverts DAC readouts while protecting other assessments. (b) Depth-six ego boxes at 3.1 s: normal (gray), donor-native (blue), donor-flip (orange); the cross is 7.53 cm outside. (c) Minimum corner margin; negative is off-road. Protocol and further cases: Appendix E.1.

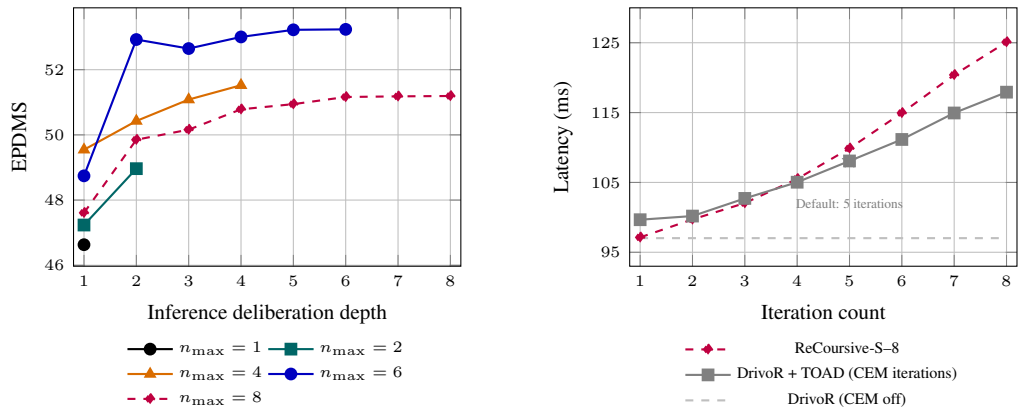

Figure 4: Depth and runtime. Left: NAVSIMv2 EPDMS for independently trained depths $n_{\max}$, selecting from $64n$ proposals at inference depth n. Right: median forward latency on one RTX 6000 Ada, batch size one. Iterations denote recurrent steps or TOAD CEM iterations (star: default five iterations); gray line: DrivoR with CEM off. Full values and timing protocol: Appendix F.

4.4 Deliberation Depth Scaling and Efficiency

Additional deliberation helps up to a point. Across independently trained NAVSIM-only checkpoints, EPDMS peaks at training depth of six steps (53.24) and declines at eight (51.19; Figure 4). The eight-step model trails the six-step model at every inference depth, so the drop reflects the eight-step training run, not extra inference steps. Within checkpoints, early revisions enable most of the gain; matched controls isolate feedback from added computation and proposal count (Section 4.2). HUGSIM shows similar saturation: route completion (RC) and HD-Score peak at inference depth three, while PDM score (PDMS) continues to improve at four (Appendix F).

Deliberation introduces modest computational overhead. For the eight-step config, ReCoursive-S-8, increasing inference depth from one to four adds 8.38 ms (8.62 %) to median forward latency, 97.13 to 105.51 ms. Eight steps take 125.09 ms, versus 117.92 ms for eight TOAD CEM iterations under the same timing settings (Figure 4). Both reuse visual features, but recurrent steps and CEM iterations are different computations, not matched budgets.

4.5 Planning Performance Across Benchmarks

NAVSIMv2. ReCoursive-S reaches 53.2 EPDMS with NAVSIM-only training, while ReCoursive-L reaches 57.2 with SimScale augmentation. Both lead the compared learned methods in their respective training regimes (Table 2). Because selection coefficients affect EPDMS, we also apply our coefficient search to the released baseline checkpoints (Appendix C.6). The search raises them by only 0.3–0.7 EPDMS (1.7 for ZTRS), since most released selectors already score near the top of the search grid. ReCoursive-L remains 1.0 point above TOAD's reproduced score (56.24) and 2.0 points above the searched DrivoR NAVSIM+SimScale checkpoint (55.23).

Table 2: NAVSIMv2 `navhard` results. †: SimScale's 53.2 is a leaderboard score that we could not reproduce with the repository's released checkpoints.

Method	Training regime	EPDMS↑
PDM-Closed (Dauner et al., 2023)	rule-based	56.6
LTF (Chitta et al., 2023)	NAVSIM	25.1
LTFv6 (Nguyen et al., 2026)	NAVSIM	31.9
RAP (Feng et al., 2026)	NAVSIM	39.6
ZTRS (Li et al., 2026)	NAVSIM	48.1
DrivoR (Kirby et al., 2026)	NAVSIM	48.3
Gigapixel-DrivoR (Rowe et al., 2026)	Self-play + NAVSIM adaptation	50.1
ReCoursive-S	NAVSIM	**53.2**
SimScale (Tian et al., 2026)	NAVSIM + SimScale	53.2†
DrivoR (Kirby et al., 2026)	NAVSIM + SimScale	54.5
DrivoR + TOAD (Xu et al., 2026)	NAVSIM + SimScale	56.3
ReCoursive-L	NAVSIM + SimScale	**57.2**

HUGSIM. ReCoursive-S–4 achieves 52.5 average RC and 43.6 average HD-Score. While TOAD achieves a strong NAVSIMv2 EPDMS of 56.3, its HUGSIM HD-Score gains over DrivoR are accompanied by reduced route completion. ReCoursive-S–4 achieves highest average RC and HD-Score among the methods in Table 3, exceeding the strongest external baseline by over 13 % in HD-Score.

Table 3: Zero-shot HUGSIM evaluation on the 345-scenario pre-challenge split. †: Because DrivoR + TOAD does not report the per-difficulty results, the averages are approximately reconstructed from its rounded per-dataset results.

	Route completion (RC) ↑					HD-Score ↑				
Method	Easy	Medium	Hard	Extreme	Avg.	Easy	Medium	Hard	Extreme	Avg.
VAD (Jiang et al., 2023)	51.3	31.1	25.3	26.5	31.4	36.3	9.5	8.0	11.5	13.4
LTF (Chitta et al., 2023)	67.8	35.1	26.2	40.5	38.9	58.9	18.0	9.8	25.9	23.7
UniAD (Hu et al., 2023)	78.4	60.5	33.6	17.8	45.9	64.9	45.8	20.6	6.6	32.7
DrivoR (Kirby et al., 2026)	**80.9**	50.5	33.8	**47.1**	49.8	73.3	34.6	18.8	**32.5**	35.7
DrivoR + SimScale (Rowe et al., 2026)	70.4	54.4	28.2	39.2	46.4	64.8	50.3	16.8	25.8	38.1
Gigapixel-DrivoR (Rowe et al., 2026)	78.6	60.7	32.7	35.4	50.1	67.4	51.9	19.1	21.6	38.5
DrivoR + TOAD† (Xu et al., 2026)	–	–	–	–	48.5	–	–	–	–	38.3
ReCoursive-S–4	80.6	**60.9**	**35.2**	41.4	**52.5**	**73.8**	**56.2**	**23.6**	27.5	**43.6**

KITScenes Longtail. ReCoursive-L–4 reaches 3.58 MMS on KITScenes Long-Tail. The NAVSIM+SimScale checkpoint achieves a mean Euclidean trajectory error of 10.79 m (L_2) on the 400-scenario test (Table 4; Wagner et al., 2026). The methods use different source training data and evaluation implementations; the cross-paper rows provide context rather than a matched re-evaluation.

Table 4: Zero-shot KITScenes LongTail test-set results. Our result is from the official leaderboard.

Method	MMS ↑	L_2 (m) ↓
VLM methods (reference only)		
AutoVLA (Zhou et al., 2025)	3.13	22.12
OneVL (Lu et al., 2026)	3.91	3.70
Alpamayo 1.5 (Wang et al., 2025)	4.69	3.30
Non-VLM methods		
Epona (Zhang et al., 2025)	2.74	17.67
UniAD (Hu et al., 2023)	3.29	10.90
DMAD (Shen et al., 2025)	3.52	**10.04**
ReCoursive-L–4	**3.58**	10.79

5 Conclusion

In end-to-end driving, latent critiques turn selection into deliberation without language. ReCoursive returns the critic's representation of each candidate to a weight-shared proposer and trains the loop with trajectory and assessment supervision at every step. These critiques improve planning beyond recurrence alone at modest latency and without adding parameters as depth increases.

Whether feedback helps depends on what the critic passes. Additional revision yields no gain with the explicit-score feedback, while removing assessment supervision or reusing stale critiques reduces the benefit. Together, these results suggest that useful feedback benefits from both representational richness and assessment supervision. ReCoursive connects the two by supervising the critic to assess while training it through its use in revision.

Several limitations qualify these findings. The critic's geometry-based targets agree with simulator targets on 88–94 % of binary labels, while all reported results use official benchmark metrics. Gains concentrate in early revisions, and depth is fixed across scenes. Interventions show that critique content affects revision, but do not establish reliable control over its direction.

Future work includes learned stopping and applications beyond driving. A learned halting rule could allocate computation where further revision helps, as in imagination-based agents (Hamrick et al., 2017; Pascanu et al., 2017). Reward models in best-of-N sampling select among generated candidates; whether their representations can similarly support learned revision remains open.

AI USE STATEMENT

We used OpenAI Codex and Anthropic Claude to assist with literature retrieval and discovery, manuscript outlining, drafting and polishing text and LaTeX preparation. We also used it to support research ideation and execution, including developing experiment code, organizing experiment records and checking cross-references and compilation. The authors reviewed AI-assisted content and checked research claims, citations, and reported results against the original sources and experimental artifacts. No experimental measurements were generated by the language model. The authors inspected the underlying implementation and evaluation artifacts, verified every reported numerical result and citation, and take responsibility for the complete manuscript.

ETHICS STATEMENT

This work studies autonomous-driving planning, a safety-critical application. All reported evaluations use simulation, replay simulation, or open-loop data; the method has not been validated for deployment on a public road. Benchmark performance does not establish safety under unmodeled actors, sensor failures, or distribution shift, and the system should not be deployed without independent closed-loop safety validation and appropriate human oversight.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix B specify the model, objective, training recipe, proposal count, deliberation depth, and evaluation metrics. Appendix C.5 specifies the NAVSIMv2 coefficient search, selection data, and frozen vectors. Experiment artifacts connect each evaluated checkpoint to its proposal pool, selector, and results. The code will be released upon acceptance.

REFERENCES

Afra Feyza Akyürek, Ekin Akyürek, Ashwin Kalyan, Peter Clark, Derry Tanti Wijaya, and Niket Tandon. RL4F: Generating Natural Language Feedback with Reinforcement Learning for Repairing Model Outputs. In *ACL*, 2023. 2, 3

Thomas Bush, Stephen Chung, Usman Anwar, Adrià Garriga-Alonso, and David Krueger. Interpreting Emergent Planning in Model-Free Reinforcement Learning. In *ICLR*, 2025. 4

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving, 2020. URL https://arxiv.org/abs/1903.11027. 17

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 *CoRL*, 2025. 16

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching Large Language Models to Self-Debug. In *ICLR*, 2024. 2, 3

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. 8, 9

Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with Misconceptions about Learning-Based Vehicle Motion Planning. In *CoRL*, 2023. 8

Lan Feng, Yang Gao, Eloi Zablocki, Quanyi Li, Wuyang Li, Sichao Liu, Matthieu Cord, and Alexandre Alahi. RAP: 3D Rasterization Augmented End-to-End Planning. In *ICLR*, 2026. 8

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach. In *NeurIPS*, 2025. 4, 14

Arthur Guez, Theophane Weber, Ioannis Antonoglou, Karen Simonyan, Oriol Vinyals, Daan Wierstra, Remi Munos, and David Silver. Learning to Search with MCTSnets. In *ICML*, 2018. 4

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 Incentivizes Reasoning in LLMs through Reinforcement Learning. *Nature*, 645(8081):633–638, 2025. 1

Ke Guo, Haochen Liu, Xiaojun Wu, Jia Pan, and Chen Lv. iPad: Iterative Proposal-Centric End-to-End Autonomous Driving. *IEEE Robotics and Automation Letters*, 11(10):11142–11149, 2026. 2, 3, 14

Jessica B. Hamrick, Andrew J. Ballard, Razvan Pascanu, Oriol Vinyals, Nicolas Heess, and Peter W. Battaglia. Metacontrol for Adaptive Imagination-Based Optimization. In *ICLR*, 2017. 1, 4, 9, 14

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training Large Language Models to Reason in a Continuous Latent Space. In *Conference on Language Modeling (COLM)*, 2025. 1, 4, 14

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 *CVPR*, 2023. 9

Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. VAD: Vectorized Scene Representation for Efficient Autonomous Driving. In *ICCV*, 2023. 9

Ellington Kirby, Alexandre Boulch, Yihong Xu, Yuan Yin, Gilles Puy, Eloi Zablocki, Andrei Bursuc, Spyros Gidaris, Renaud Marlet, Florent Bartoccioni, Anh-Quan Cao, Nermin Samet, Tuan-Hung Vu, and Matthieu Cord. Driving on Registers. In *CVPR*, 2026. 1, 3, 4, 8, 9, 14

Pengxiang Li, Yinan Zheng, Yue Wang, Huimin Wang, Hang Zhao, Jingjing Liu, Xianyuan Zhan, Kun Zhan, and Xianpeng Lang. Discrete Diffusion for Reflective Vision-Language-Action Models in Autonomous Driving, 2025a. URL https://arxiv.org/abs/2509.20109. 2, 3, 14

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 *ICCV*, 2025b. 1, 3, 14

Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, Yu-Gang Jiang, and Jose M. Alvarez. Hydra-MDP: End-to-end multimodal planning with multi-target hydra-distillation, 2024. URL https://arxiv.org/abs/2406.06978. 3, 6, 14

Zhenxin Li, Wenhao Yao, Zi Wang, Xinglong Sun, Joshua Chen, Nadine Chang, Maying Shen, Zuxuan Wu, Shiyi Lan, and Jose M. Alvarez. Generalized Trajectory Scoring for End-to-End Multimodal Planning, 2025c. URL https://arxiv.org/abs/2506.06664. 3, 14

Zhenxin Li, Nadine Chang, Wenhao Yao, Xinglong Sun, Zi Wang, Maying Shen, Jingde Chen, Jingyu Song, Kailin Li, Zuxuan Wu, Shiyi Lan, and Jose M. Alvarez. ZTRS: Zero-Human Demonstration End-to-End Autonomous Driving with Trajectory Scorer. In *ECCV*, 2026. 8

Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D, 2021. URL https://arxiv.org/abs/2109.13410. 17

Jinghui Lu, Jiayi Guan, Zhijian Huang, Jinlong Li, Guang Li, Lingdong Kong, et al. OneVL: One-Step Latent Reasoning and Planning with Vision-Language Explanation, 2026. URL https://arxiv.org/abs/2604.18486. 9

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. In *NeurIPS*, 2023. 1, 2, 3, 14

Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative Amortized Inference. In *ICML*, 2018. 4, 14

Long Nguyen, Micha Fauth, Bernhard Jaeger, Daniel Dauner, Maximilian Igl, Andreas Geiger, and Kashyap Chitta. LEAD: Minimizing Learner-Expert Asymmetry in End-to-End Driving. In *CVPR*, 2026. 8

Razvan Pascanu, Yujia Li, Oriol Vinyals, Nicolas Heess, Lars Buesing, Sebastien Racanière, David Reichert, Théophane Weber, Daan Wierstra, and Peter Battaglia. Learning Model-Based Planning from Scratch, 2017. URL https://arxiv.org/abs/1707.06170. 1, 4, 9, 14

A. David Redish. Vicarious trial and error. *Nature Reviews Neuroscience*, 17(3):147–159, 2016. 1

Luke Rowe, Roger Girgis, Rodrigue de Schaetzen, Daphne Cornelisse, Alaap Grandhi, Felix Heide, Eugene Vinitsky, Christopher Pal, and Liam Paull. Scaling Self-Play for End-to-End Driving, 2026. URL https://arxiv.org/abs/2606.19641. 8, 9

Yinzhe Shen, Omer Sahin Tas, Kaiwen Wang, Royden Wagner, and Christoph Stiller. Divide and Merge: Motion and Semantic Learning in End-to-End Autonomous Driving. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2502.07631. 9

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *CVPR*, 2020. 17

Shuhan Tan, Kashyap Chitta, Yuxiao Chen, Ran Tian, Yurong You, Yan Wang, Wenjie Luo, Yulong Cao, Philipp Krahenbuhl, Marco Pavone, and Boris Ivanovic. Latent Chain-of-Thought World Modeling for End-to-End Autonomous Driving. In *CVPR*, 2026. 1, 4, 14

Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *ECCV*, 2020. 4, 14

Haochen Tian, Tianyu Li, Haochen Liu, Jiazhi Yang, Yihang Qiu, Guang Li, Junli Wang, Yinfeng Gao, Zhang Zhang, Liang Wang, Hangjun Ye, Long Chen, and Hongyang Li. SimScale: Learning to drive via real-world simulation at scale. In *CVPR*, 2026. 8, 14

Royden Wagner, Omer Sahin Tas, Jaime Villa, Felix Hauser, Yinzhe Shen, Marlon Steiner, Dominik Strutz, Carlos Fernandez, Quentin Delfosse, Christoph Weinhuber, Christian Kinzig, Guillermo S. Gutierrez-Cabello, Hendrik Koenigshof, Fabian Immel, Richard Schwarzkopf, Nils Alexander Rack, Kevin Roesch, Kaiwen Wang, Jan-Hendrik Pauls, Martin Lauer, Igor Gilitschenski, Holger Caesar, and Christoph Stiller. Reasoning Models Do Not Yet Follow Their Reasoning in Autonomous Driving: The KITScenes LongTail Dataset, 2026. URL https://arxiv.org/abs/2603.23607v3. 9, 17

Huimin Wang, Yue Wang, Bihao Cui, Pengxiang Li, Ben Lu, Mingqian Wang, Tong Wang, Chuan Tang, Teng Zhang, and Kun Zhan. ReflectDrive-2: Reinforcement-Learning-Aligned Self-Editing for Discrete Diffusion Driving, 2026. URL https://arxiv.org/abs/2605.04647. 3, 14

Tianqi Wang, Enze Xie, Ruihang Chu, Zhenguo Li, and Ping Luo. DriveCoT: Integrating Chain-of-Thought Reasoning with End-to-End Driving, 2024. URL https://arxiv.org/abs/2403.16996. 1, 3

Yan Wang, Wenjie Luo, Junjie Bai, Yulong Cao, Tong Che, Ke Chen, et al. Alpamayo-R1: Bridging Reasoning and Action Prediction for Generalizable Autonomous Driving in the Long Tail, 2025. URL https://arxiv.org/abs/2511.00088. 9

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*, 2022. 1

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Deliberation Networks: Sequence Generation Beyond One-Pass Decoding. In *NeurIPS*, 2017. 1, 14

Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, Yunlong Wang, and Diange Yang. PandaSet: Advanced Sensor Suite Dataset for Autonomous Driving, 2021. URL https://arxiv.org/abs/2112.12610. 17

Yihong Xu, Eloi Zablocki, Yuan Yin, Elias Ramzi, Ellington Kirby, Alexandre Boulch, and Matthieu Cord. Test-time trajectory optimization for autonomous driving, 2026. URL https://arxiv.org/abs/2606.07170. 2, 3, 8, 9, 14

Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Jain, Jeff Schneider, and Katerina Fragkiadaki. Diffusion-ES: Gradient-free planning with diffusion for autonomous driving and zero-shot instruction following, 2024. URL https://arxiv.org/abs/2402.06559. 2, 3, 14

Xuemei Yao, Xiao Yang, Jianbin Sun, Liuwei Xie, Xuebin Shao, Xiyu Fang, Hang Su, and Kewei Yang. ReflexDiffusion: Reflection-Enhanced Trajectory Planning for High-Lateral-Acceleration Scenarios in Autonomous Driving. In *AAAI*, 2026. 3

Liuhan Yin, Runkun Ju, Guodong Guo, and Erkang Cheng. DiffRefiner: Coarse-to-fine trajectory planning via diffusion refinement with semantic interaction for end-to-end autonomous driving. In *AAAI*, 2026. 2, 3, 14

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 *ICLR*, 2026. 1, 4

Kaiwen Zhang, Zhenyu Tang, Xiaotao Hu, Xingang Pan, Xiaoyang Guo, Yuan Liu, Jingwei Huang, Li Yuan, Qian Zhang, Xiao-Xiao Long, Xun Cao, and Wei Yin. Epona: Autoregressive Diffusion World Model for Autonomous Driving. In *ICCV*, 2025. 9

Hongyu Zhou, Longzhong Lin, Jiabao Wang, Yichong Lu, Dongfeng Bai, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. HUGSIM: A Real-Time, Photo-Realistic and Closed-Loop Simulator for Autonomous Driving. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 48(04):4673–4691, 2026. 17

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 *NeurIPS*, 2025. 9

A Overview on works that condition the next revision

Table 5: Prior work, organized by what conditions the next revision.

Family	Representative works	Input to revision	Revision rule
Scorer-based selection	Hydra-MDP (Li et al., 2024) GTRS (Li et al., 2025c) WoTE (Li et al., 2025b) DrivoR (Kirby et al., 2026)	— (selection only)	—
Refinement without assessment	iPad (Guo et al., 2026) DiffRefiner (Yin et al., 2026)	Proposal and scene features	Learned
Reward- or scorer-guided search	Diffusion-ES (Yang et al., 2024) TOAD (Xu et al., 2026)	Scalar rewards or scores	Hand-designed (evolutionary search, CEM)
Oracle-guided editing	ReflectDrive (Li et al., 2025a)	External safety-oracle checks	Oracle edits, then inpainting
Learned self-editing	ReflectDrive-2 (Wang et al., 2026)	Token confidence	Learned
Linguistic self-refinement	Self-Refine (Madaan et al., 2023)	A natural-language critique	Prompted
Imagination-based planning	(Hamrick et al., 2017) (Pascanu et al., 2017)	Imagined outcomes	Learned
Two-pass decoding	Deliberation networks (Xia et al., 2017)	A first-pass draft	Learned
Recurrent refinement	RAFT (Teed & Deng, 2020) Iterative amortized inference (Marino et al., 2018)	Matching costs or gradients	Learned
Latent reasoning	Coconut (Hao et al., 2025) LCDrive (Tan et al., 2026) Recurrent depth (Geiping et al., 2025)	Latent states without an assessment target	Learned
ReCoursive		**Supervised latent critique of each candidate**	**Learned; critic trained in the loop**

B Implementation and Training Details

B.1 Model Configurations and Training Recipes

The S and L variants use DINOv2 ViT-S/14 and ViT-L/14 image encoders, respectively. Table 6 gives their shared architecture. NAVSIM-only models use `navtrain`, 10 training epochs, seed 2, and batch size of 64. AdamW uses a base learning rate of 2×10^{-4}, 10 % warm-up, and cosine decay. For NAVSIM+SimScale training, we add SimScale (Tian et al., 2026) `synthetic_reaction_pdm_v1.0` shards 0–2 to NAVSIM `navtrain` and train for 12 epochs.

Table 6: Architecture shared by the S and L variants. Images use width × height notation.

Component	Shared setting
Visual adaptation	Rank-32 LoRA; pretrained backbone weights frozen
Input	Four current views, 1148×672 pixels
Proposer / critic width	256; feed-forward width 1024
Proposal budget	64 slots in four branches of 16 per step
Decoder refinement	Four inner layers per recurrent step; one attention head
Output	Eight (p_x, p_y, θ) poses at 0.5 s intervals
Feedback	Latent critiques; zero at step one

B.2 Geometry-Based Critic Targets

For each proposal at every outer training step, we compute the geometry-based targets $y^n_{k,m}$ used in the scorer loss of Eq. 6. The target generator reads the training scene's metric cache: initial ego state and vehicle dimensions, drivable-area polygons, route lane identifiers, centerline, and future tracked-object occupancies. It transforms the predicted 2 Hz ego-frame poses into world coordinates, interpolates position and unwrapped heading to 40 future samples at 0.1 s intervals, and obtains velocities by finite differences. The checks in Table 7 compute NoC, DAC, DDC, TTC, EP, and COMF values for each proposal; only the first five supervise critic's metric heads. Target generation operates directly on these interpolated trajectories, without a controller or `PDMSimulator` rollout.

Table 7: Geometry-based target construction. NoC and DDC intermediate values of 0.5 are mapped to zero for binary cross-entropy supervision.

Head	Computation
NoC	Intersect oriented ego boxes with future object polygons. At-fault front/stopped-track collisions, and lateral collisions when off-road, receive 0 for traffic participants and 0.5 for other objects; otherwise 1. Previously collided tracks and red-light tokens are excluded.
DAC	1 iff all four ego corners lie in the union of roadblock, intersection, drivable-area, and carpark polygons at every sample; otherwise 0.
DDC	Sum traveled distance while the ego center is outside on-route lanes/connectors in a sliding window (sample indices $t-10$ through t, clipped at the start). Maximum distance below 2 m gives 1, from 2 m to below 6 m gives 0.5, and at least 6 m gives 0.
TTC	At each sample, project the ego box at constant speed/heading by 0, 0.3, 0.6 and 0.9 s and intersect future occupancies. A relevant collision ahead, or not behind when off-road/in an intersection, gives 0; otherwise 1. Speeds below 0.05 m s^{-1} are exempt.
EP	Let $d_k \geq 0$ be centerline progress from the first to last sampled center and $a_k = \mathrm{NoC}_k \mathrm{DAC}_k \mathrm{DDC}_k$. Set $u_k = a_k d_k$. If $\max_j u_j > 5$ m, use $u_k / \max_j u_j$; otherwise use $\mathbf{1}[a_k > 0]$. $\mathbf{1}[\cdot]$ is the indicator; j ranges over jointly scored proposals.
COMF	Binary comfort check on interpolated ego states using finite-difference kinematics.

We exclude COMF from both assessment supervision and proposal selection by setting its loss weight and selection coefficient to zero: its geometry-based target has no positive rank association with simulator-rolled COMF in the fixed training-cache audit below.

B.3 Geometry-Based Targets Agreement Audit

We compare geometry-based targets with the true PDM score targets on fixed trajectories. The audit samples 500 of 103,288 training-cache entries; the selected scenes span 341 logs. Each receives 64 synthetic fan trajectories with speeds in $[1, 12]$ m s^{-1} and headings in $[-0.5, 0.5]$ rad, giving 32,000 paired target comparisons.

Table 8 reports pooled Spearman correlation ρ on raw targets and binary agreement after the training-loss remapping of intermediate NoC/DDC values ($0.5 \mapsto 0$). Violation recall is the fraction of rollout violations also identified by the geometry-based proxy. EP is continuous, so its binary columns are inapplicable. No invalid TTC reference targets occurred.

Table 8: Agreement with PDM simulation targets on 500 training-cache scenes and 32,000 fixed candidate trajectories. Agreement and violation recall are percentages.

Head	Pooled ρ	Binary agreement	Violation recall
NoC	0.76	87.67	86.80
DAC	0.87	93.70	96.10
DDC	0.91	93.37	95.76
TTC	0.78	89.17	89.26
EP	0.67	—	—
COMF	–0.05	48.68	42.43

The five supervised heads have positive pooled rank association. COMF has no positive association and identifies fewer than half of rollout comfort violations.

B.4 Training Efficiency of Geometry-Based Targets

We measure the training cost of replacing PDM simulator targets with the geometry-based proxy in ReCoursive-S at training deliberation depths $n_{\max} \in \{1, 2, 4, 6, 8\}$. Both measures use four NVIDIA RTX 6000 Ada GPUs, an effective batch size of 64, 64 proposals in four branches, supervision at every recurrent step, and cross-scene ground-truth borrowing. The host uses AMD EPYC 7763 CPUs, with 32 Ray workers and four data-loader workers per rank. We perform two runs per setup in PDM–proxy–proxy–PDM order, each with 20 warm-up updates followed by 100 measured optimizer updates, and average the per-run seconds per step. The full training interval includes data waiting, device transfer, forward computation, target generation, backpropagation, distributed synchronization, AdamW, and the learning-rate scheduler; it excludes initialization, warm-up, validation, and checkpoint writing. Table 9 shows a 1.33–$1.56\times$ improvement in training throughput, reducing full-step time by 24.62–35.73 % over the measured depths.

Table 9: Training efficiency of geometry-based targets under matched hardware and training settings. Full-step times include optimizer updates. Speedup is PDM time divided by proxy time. Target-path speedup uses rank-maximum timings of ordinary and borrowed-target calls, including their transfers and Ray calls.

Training deliberation depth	PDM Simulator (s/step)	Proxy (s/step)	Full-step speedup	Target-path speedup
1	3.63	2.73	1.33×	1.81×
2	4.59	3.27	1.40×	1.83×
4	6.49	4.34	1.49×	1.83×
6	8.41	5.47	1.54×	1.80×
8	10.34	6.65	1.56×	1.79×

B.5 Gradient Flow and Cross-Scene Supervision

The trajectory coordinates are detached before the critic's trajectory embedder. Latent critiques passed to the next proposer and recurrent proposal states are not detached in the full model. Thus later planning losses train earlier latent critiques and the critic, while not differentiating through its geometric trajectory input or the geometry-based target computation. Scene features remain shared and trainable through the task losses.

The 64 slots form four fixed branches of 16. Proposal self-attention, critic candidate attention, and feedback cross-attention preserve these branches; scene features are shared. The trajectory loss takes the minimum across all 64 proposals, after summing absolute error over pose coordinates and averaging over timestamps, then averages over samples. The standard inner-layer weighting retains the final inner prediction; outer deliberation-step losses are summed with equal weight.

For receiving scene i and donor $j \neq i$ in the B-scene batch, o_i denotes receiver observations and τ_j^* the donor demonstration. The borrowed trajectory and receiver target are

$$\tilde{\tau}_{i\leftarrow j} = \tau_j^*, \qquad \tilde{y}_{i\leftarrow j,m} = Q_m(o_i, \tilde{\tau}_{i\leftarrow j}). \tag{10}$$

Writing $S_m(o, \tau)$ for the critic logit in scene o, the auxiliary objective is

$$\mathcal{L}_{\text{cf}} = \frac{1}{B(B-1)} \sum_i \sum_{j\neq i} \sum_{m\in\mathcal{M}_{\text{train}}} \text{BCE}(S_m(o_i, \tilde{\tau}_{i\leftarrow j}), \tilde{y}_{i\leftarrow j,m}), \tag{11}$$

with each head averaged over its valid pairs as detailed below. The general inner-head trajectory objective is $\mathcal{L}^n_{\text{traj}} = \sum_{\ell=0}^{L} w_\ell \min_k H^{-1} \sum_h \|\tau^{n,\ell}_{k,h} - \tau^*_h\|_1$, where $\ell \in \{0, \ldots, L\}$ indexes inner proposal heads, L is the final head index, w_ℓ its loss weight, and $\tau^{n,\ell}_k$ its prediction. Thus $\tau^{n,L}_k = \tau^n_k$; the standard recipe sets $w_L = 1$ and earlier weights to zero.

Each receiving scene borrows all other 63 ground-truth trajectories from the effective distributed batch, expressed in their donor-local frame and assessed as hypothetical trajectories in the receiver's scene. The auxiliary critic call uses the final scene features and disables the branch attention mask for the borrowed set. For each supervised head, BCE is averaged over receiver–donor pairs (TTC averages only valid targets); the five head losses are summed. This auxiliary sum is added once, after the outer-step loss sum, with unit coefficient. NoC/DDC use the same binary remapping as ordinary proposal supervision. Borrowing is absent at inference.

C Evaluation and Proposal Selection

C.1 NAVSIMv2

We evaluate on the `navhard` split of NAVSIMv2 (Cao et al., 2025), which contains 5,912 scenes. The first stage scores the selected trajectory in the source scene; the second conditions a continuation evaluation on its endpoint. The primary quantity is the resulting Extended Predictive Driver Model Score (EPDMS), reported on the conventional 0–100 scale. EPDMS aggregates no-collision, drivable-area, driving-direction, traffic-line, progress, time-to-collision, lane-keeping, and two comfort terms over the two stages.

C.2 HUGSIM

HUGSIM is a photorealistic closed-loop replay simulator (Zhou et al., 2026), with scenes reconstructed from KITTI-360, Waymo, nuScenes, and PandaSet (Liao et al., 2021; Sun et al., 2020; Caesar et al., 2020; Xiao et al., 2021). We evaluate on its 345-scenario pre-challenge split, reporting route completion (RC), HD-Score, and PDM Score.

C.3 KITScenes LongTail

KITScenes LongTail is an open-loop benchmark for long-tail, instruction-conditioned driving scenarios (Wagner et al., 2026). Its 1,000 scenarios are split into 500 training, 100 validation, and 400 test scenarios. We report the Multi-Maneuver Score (MMS) and trajectory L_2 error.

C.4 Cross-Benchmark Adaptation

We report the interface adaptation used for zero-shot evaluation on HUGSIM and KITScenes LongTail in Table 10.

Table 10: Input/output adapters with frozen model parameters. Both benchmarks use the source image resize and normalization, with corresponding intrinsic scaling.

Interface	HUGSIM	KITScenes LongTail
Cameras	Front, front-left, front-right, rear where available; KITTI-360 and Waymo duplicate the front view into the rear slot. Simulator extrinsics are inverted to sensor-to-ego transforms.	Released front, front-left, front-right, rear images; released pinhole calibration and histogram-stretched RGB. No missing-view substitution.
Ego state	Simulator speed and acceleration as forward components; zero lateral components and local origin.	Backward differences of the last three released past positions at 0.2 s spacing; local origin, x forward and y left.
Commands	Simulator right/left/straight codes map to the source left/straight/right/unknown one-hot order.	Explicit turns and lane-use instructions map to direction; U-turn and overtaking an object in front map left; overtaking side objects maps straight. Unsupported text is rejected.
Output	Eight 0.5 s waypoints mapped from $(x_{\text{forward}}, y_{\text{left}})$ to simulator $(-y_{\text{left}}, x_{\text{forward}})$ and sent to the simulator's trajectory-following interface.	Anchor at the origin, extend the 4 s prediction to 5 s at terminal velocity, and linearly interpolate to 25 future positions at 5 Hz for official submission.

C.5 Selection Protocols and Coefficients

Each checkpoint is tuned on all 5,912 `navhard` evaluation scenes in 225 complete reactive groups, maximizing official two-stage EPDMS with `human_penalty_filter=True`. The five coefficients (NoC, DAC, DDC, TTC, EP) range over $\{0.1, 0.3, 1, 3, 10, 30, 100\}^5$ (16,807 vectors). We refine the best vector with coordinate-wise factors $\{0.5, 0.7, 1, 1.4, 2\}$ (3,124 additional vectors), and fix COMF to zero. Selection is fitted at maximum cumulative depth and frozen across depths.

At depth n, selection ranks all $64n$ accumulated proposals. Each proposal retains scores computed in its original iteration and branch context.

C.6 Released-Checkpoint Selection Audit

Because the published baselines do not specify how their selection coefficients were searched, we audit available released checkpoints using the same coefficient-search algorithm as ReCoursive, and additionally test coefficient transfer from DrivoR to TOAD (Table 12).

For TOAD, w_m denotes its selection coefficient. We transfer the coefficients selected for the same released DrivoR / NAVSIM+SimScale checkpoint: $(w_{\text{NoC}}, w_{\text{DAC}}, w_{\text{DDC}}, w_{\text{TTC}}, w_{\text{EP}}, w_{\text{COMF}}) = (0.005, 0.2, 5, 0.005, 0.014, 0.1)$. We rerun native CEM with these coefficients. Across all 5,912 scenes and 225 reactive groups, this gives 55.72 EPDMS, compared with 56.24 using the released coefficients and 55.23 for the tuned DrivoR without CEM. The transferred coefficients retain a 0.49-point benefit from CEM but reduce performance by 0.52 points relative to the original TOAD configuration. LTF and LTFv6 output a single trajectory, RAP uses a native total-score selector, and

Table 11: Selection coefficients for the NAVSIMv2 ReCoursive checkpoints and ablations. "Sup." and "unsup." refer to assessment supervision of the critic producing latent critiques; the critic label identifies which critic is used for final selection.

Training	Checkpoint / condition	NoC	DAC	DDC	TTC	EP
NAVSIM	ReCoursive-S–1	0.2	5	3	50	0.2
NAVSIM	ReCoursive-S–2	0.1	3	30	100	0.05
NAVSIM	ReCoursive-S–4	0.14	0.3	4.2	7	0.07
NAVSIM	ReCoursive-S	0.14	0.1	2.1	21	0.21
NAVSIM	ReCoursive-S–8	0.1	0.3	6	140	1
NAVSIM	ReCoursive-S–4, no feedback	0.07	42	42	50	0.2
NAVSIM	ReCoursive-S, no feedback	0.21	1	1.4	200	2
NAVSIM	ReCoursive-S, explicit-score feedback	0.1	21	70	15	0.1
NAVSIM	ReCoursive-S, unsup. / full critic	0.21	10	30	50	0.2
NAVSIM	ReCoursive-S, sup. / no-feedback critic	0.05	0.42	4.2	100	0.2
NAVSIM	ReCoursive-S, unsup. / no-feedback critic	0.3	3	3	60	0.1
NAVSIM+SimScale	ReCoursive-S–4	200	42	5	0.3	0.7
NAVSIM+SimScale	ReCoursive-L–4	7	15	0.42	2	0.21
NAVSIM+SimScale	ReCoursive-L	15	5	0.3	0.1	0.05

Table 12: Search protocol audit on NAVSIMv2. Gain is the change from the locally evaluated released coefficients. Percentage is the released-coefficient score's percentile among evaluated coarse coefficients. Search pools are capped at 20,000 coefficient vectors; for larger grids, we sample without replacement to 16,800 coarse vectors, and reserve up to 3,200 for local refinement. †: SimScale's 53.2 is a leaderboard score that we could not reproduce with the repository's released checkpoints. ‡: TOAD uses the coefficients selected for DrivoR / NAVSIM+SimScale.

Method / checkpoint	Leaderboard	Released coefficients	Searched coefficients	Gain	Percentage	Search pool size
DrivoR / NAVSIM	48.3	48.31	48.60	+0.29	100.00	18,826
DrivoR / NAVSIM+SimScale	54.5	54.57	55.23	+0.65	99.84	18,826
ZTRS / NAVSIM	48.1	48.10	49.81	+1.71	87.01	20,000
SimScale / V2-99 reward	53.2†	51.28	51.94	+0.66	99.92	20,000
DrivoR + TOAD‡	56.3	56.24	55.72	–0.52	–	1
ReCoursive-L	–	–	57.24	–	–	19,931

PDM-Closed is rule-based, so this coefficient search audit does not apply to them. Gigapixel-DrivoR checkpoint was not available at the time of this evaluation.

D Supplementary Feedback Ablations

D.1 Explicit-Score Feedback

The affine projection takes sigmoid predictions for NoC, DAC, DDC, TTC, and EP; Comfort is excluded because its supervision is masked in the lightweight critic. The projected tokens enter the branch-local cross-attention interface. The projection uses ordinary nonzero linear-layer initialization, while first-step feedback tokens remain zero.

Table 13: **Explicit-score feedback versus ReCoursive-S.** Δ is EPDMS with latent critiques minus EPDMS with explicit-score feedback, in percentage points.

Depth	Latent critiques	Explicit-score feedback	Δ
1	48.74	50.50	−1.75
2	52.93	50.27	+2.65
3	52.65	49.32	+3.33
4	53.01	49.76	+3.24
5	53.22	50.00	+3.23
6	53.24	50.01	+3.23

D.2 Assessment Supervision under Shared Critics

Both proposers use ReCoursive-S with six deliberation steps, 64 proposals per step, four isolated branches, and the same 10-epoch recipe, evaluated at their final checkpoints. The variant without assessment supervision also disables the auxiliary borrowing branch and training-time metric-target computation. For evaluation, we rescore each generator's unchanged proposal arrays in their original branch context using either the full model's critic or the independently trained no-feedback critic. Each critic includes its own scene and ego encoders, trajectory embedder, attention blocks, and metric heads; coefficients are fitted independently for each generator–critic pair (Table 11). With the full-model critic, the depth-six gap is 6.48 EPDMS points (53.24 versus 46.76).

Table 14: **Official EPDMS across depth under both frozen critics.** "Sup." and "Unsup." refer only to assessment supervision of the critic producing latent critiques; both final selectors are metric-trained.

		Full n6 critic		No-feedback n6 critic	
Depth	Pool	Sup.	Unsup.	Sup.	Unsup.
1	64	48.74	45.30	48.59	45.09
2	128	52.93	46.59	51.06	46.10
3	192	52.65	47.14	51.67	46.23
4	256	53.01	46.95	51.45	46.08
5	320	53.22	46.81	51.71	46.16
6	384	53.24	46.76	51.69	46.37

The difference under the common critic already appears at depth one (48.59 versus 45.09), so the depth-six gap cannot be attributed entirely to later feedback-driven revision. Removing metric losses can also affect the shared scene representation during training.

E Feedback Intervention Analysis

E.1 Fixed-Geometry Donor Editing Protocol

Setup. These comparisons freeze the ReCoursive-S checkpoint and preserve the initial scene, proposal state, proposal slots, and initial trajectories across conditions. The 64 candidates form four branches of 16; a target can attend to its branch's 16 feedback tokens. Raw DAC and NoC are produced by the PDM simulator. Endpoint displacement measures generated proposals. The signed minimum corner margin measures simulated vehicle corners over the four-second planning horizon: positive is inside, negative outside. Editing success requires DAC probability within 0.05 of its target and all other head probability changes at most 0.03.

Donor construction and intervention. Each scene uses eight raw DAC-compliant and eight raw DAC-noncompliant donors, all NoC-compliant. Donors are smooth lateral variants of the initial target. Donors supply feedback only.

Donor-native injects the unmodified 16 donor latents. *Donor-flip* uses the same donors but edits DAC readouts to 0.05 for true DAC-compliant donors and 0.95 for true noncompliant donors, protecting the other five head logits. Both repeat their fixed bank at all five revisions. Original native feedback is a contextual reference. For the four displayed cases, maximum other-head probability drift is 1.54×10^{-5}.

Editing algorithm. In this subsection, $h \in \mathbb{R}^d$ denotes the editable donor latent, h_0 its initial value, and $g(h) = (g_{\psi,m}(h))_m \in \mathbb{R}^6$ its head-logit vector; $\|\cdot\|_2$ is the Euclidean norm and logit the inverse sigmoid. The desired vector y retains the five non-DAC logits of $g(h_0)$ and replaces DAC by $\mathrm{logit}(0.05)$ or $\mathrm{logit}(0.95)$ according to the donor's true label. We minimize $\|g(h) - y\|_2^2$ subject to $\|h - h_0\|_2 / \|h_0\|_2 \le 1$. Each of at most 30 iterations recomputes the Jacobian $J = \partial g(h)/\partial h$ and the local latent update $\delta = J^\top (JJ^\top + \eta I)^{-1}(y - g(h))$, with $\eta = \max(10^{-7}\,\mathrm{mean}\,\mathrm{diag}(JJ^\top), 10^{-12})$ the regularization strength and I the 6×6 identity. Here $\top$ denotes transpose, diag extracts the diagonal and mean averages its entries. Updates are capped at $0.1\|h_0\|_2$; backtracking tries factors $1, 1/2, 1/4, 1/8, 1/16$, projects into the relative-norm ball, and accepts the first decrease in squared logit residual. Optimization stops below residual 10^{-8} or when no step improves it. Protection is optimized in logit space; the reported success tolerances are checked in probability space.

E.2 Results and Scene Visualizations

Figure 5 shows four cases for DAC editing. These cases are selected for illustration and do not estimate how often editing changes compliance.

E.3 Joint NoC and TTC Feedback Editing

Protocol and cohort. We retain the paired setup and feedback-editing procedure in Appendix E.1, but construct collision donors using smooth longitudinal speed profiles along reference paths, with optional lateral offsets. Each fixed bank contains eight NoC-compliant and eight NoC-noncompliant donors, all DAC-compliant. Each class includes at least four pure longitudinal variants spanning at least 2 m of terminal path progress. We select several scenes for illustration. Joint donor-flip sets both NoC and TTC probabilities to 0.05 for NoC-compliant donors and to 0.95 for NoC-noncompliant donors, while protecting the other four head logits. Both targets therefore follow NoC truth, rather than independently inverting each metric's ground truth. After editing, the maximum protected probability drift is below 2.14×10^{-5}.

Results. Figure 6 shows four example scenes comparing donor-native and joint donor-flip feedback. The four illustrations are selected from distinct driving logs. Donor-native has no at-fault collision over the four-second horizon in these cases, whereas joint donor-flip introduces a front-vehicle collision. These examples illustrate candidate-level revision sensitivity to the edited feedback.

F Additional Depth and Runtime Results

F.1 HUGSIM Depth Results

Figure 7 shows the HUGSIM inference deliberation depth curves for the fixed $n_{\max} = 4$ checkpoint; Table 15 lists the corresponding values.

Table 15: HUGSIM depth scaling for the fixed $n_{\max} = 4$ checkpoint. All scores use the 0–100 scale.

Inference depth	RC	HD-Score	PDMS
1	50.15	41.48	64.76
2	51.83	42.98	64.82
3	52.61	43.69	64.89
4	52.54	43.57	64.97

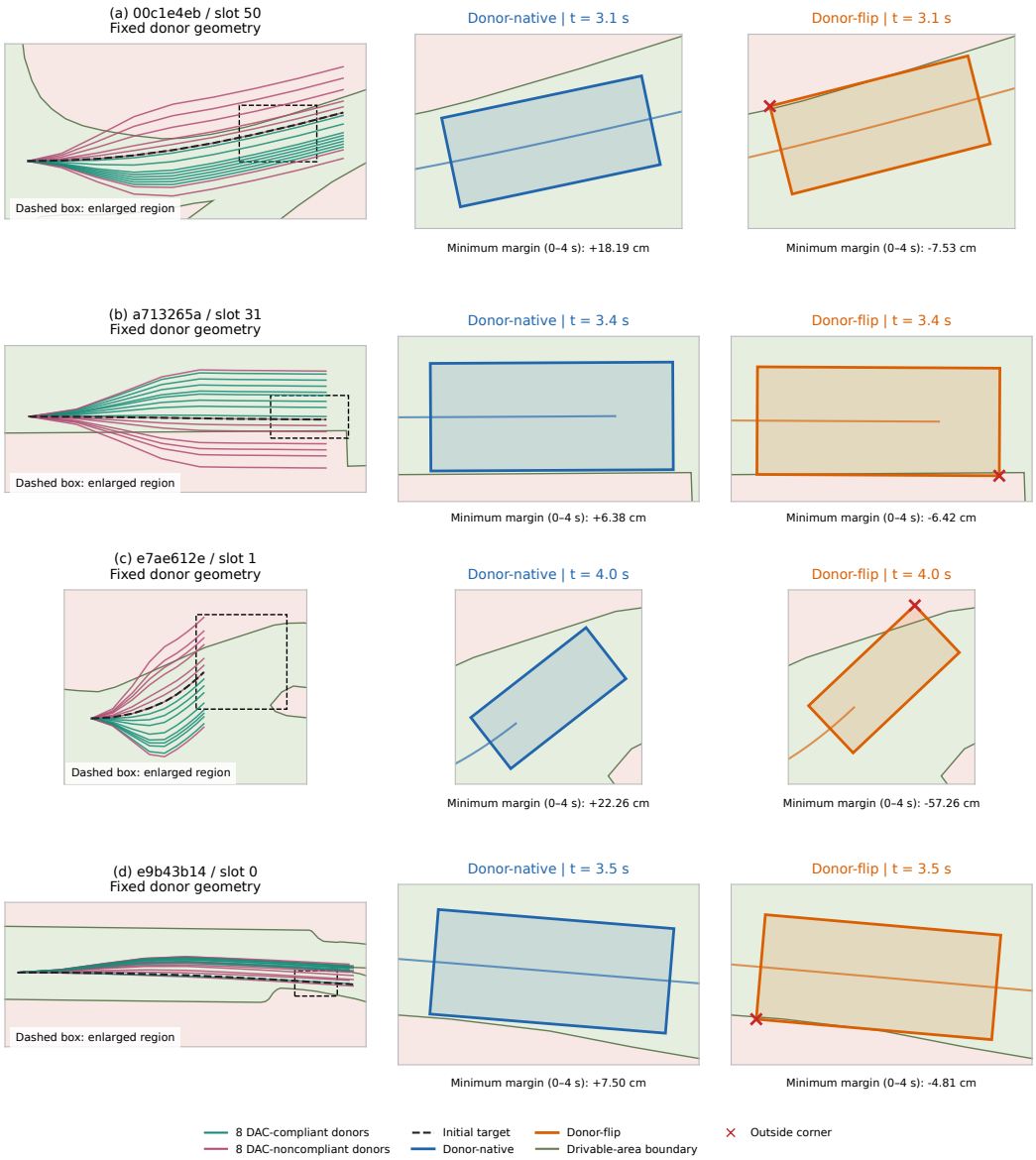

Figure 5: **Scene-level effects of DAC feedback editing at $n = 6$.** Left: the fixed bank of 16 donor trajectories, colored by raw DAC labels (eight compliant in teal, eight noncompliant in purple), with the initial target shown as a black dashed line. Donor geometry is identical in both conditions. Middle/right: matched zoom-in views of the simulated target under donor-native (blue) and donor-flip (orange) feedback. Green denotes the drivable area; pink denotes the area outside it. Red crosses mark outside corners. Annotated margins are full-horizon minima. Donor-native remains within the drivable area throughout the horizon, whereas donor-flip crosses the boundary.

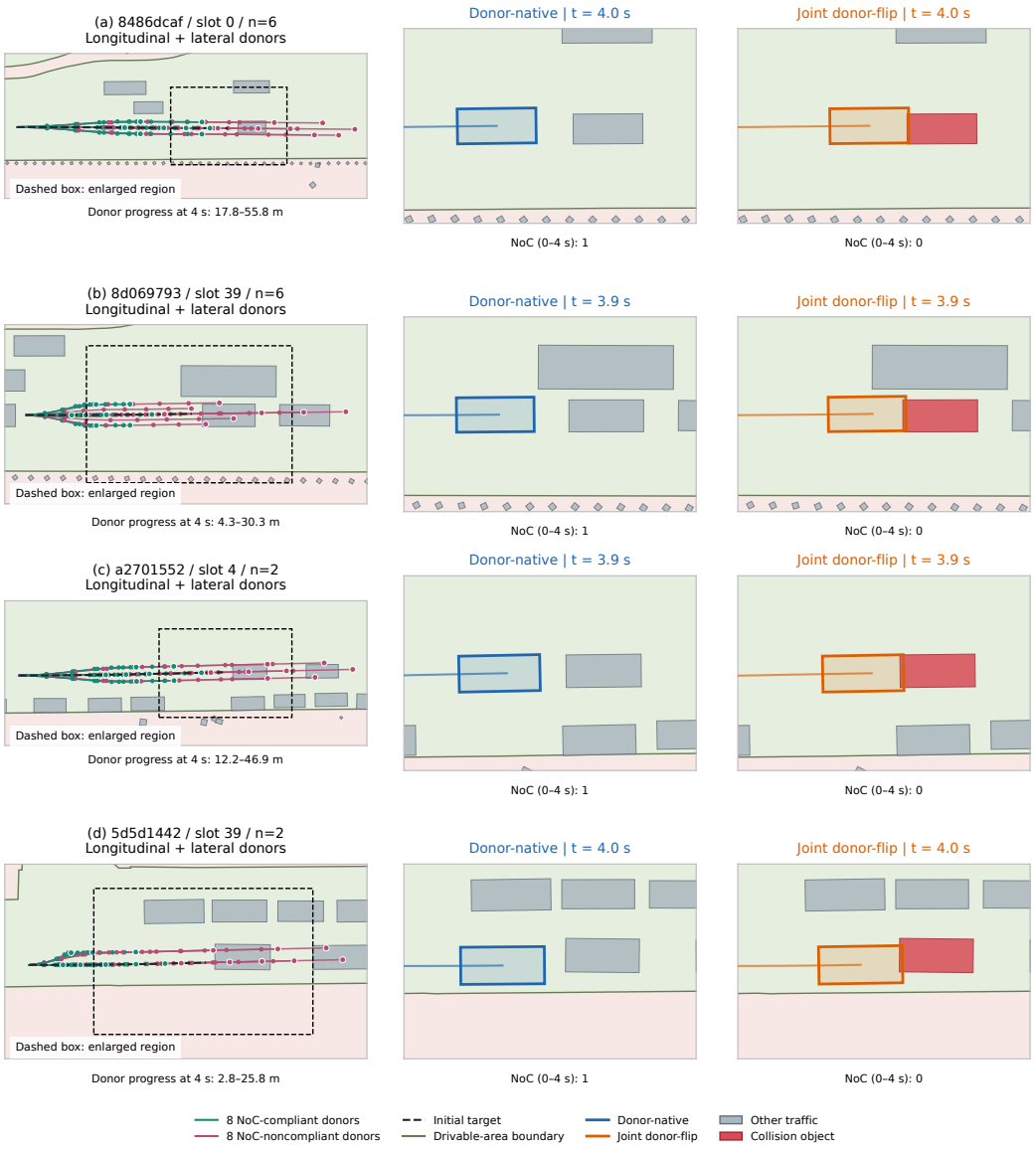

Figure 6: **Collision introductions at selected revisions under joint NoC+TTC editing.** Left: the fixed bank of 16 donors (eight NoC-compliant in teal and eight noncompliant in purple), the initial target (black dashed), and surrounding traffic. Donors vary longitudinal progress and optional lateral offset; dots mark 0.5 s samples, with larger dots at trajectory endpoints. The dashed box marks the zoom-in view region; donor geometry is shared between conditions. Middle/right: donor-native and joint donor-flip at the same prediction timestamp, the first at-fault collision under joint donor-flip. Green denotes the drivable area; pink denotes the area outside it. Blue and orange polygons are simulated ego bounding boxes; gray polygons are other agents in the scene, and red marks the object involved in the collision. Lines trace the simulated ego motion. All four cases show front-vehicle collisions under joint donor-flip; donor-native has no at-fault collision.

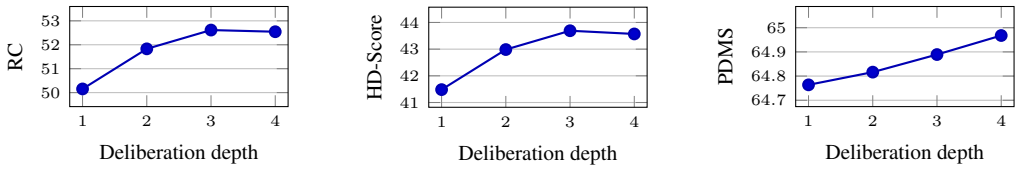

Figure 7: HUGSIM deliberation depth scaling. Each inference deliberation depth (1–4) runs all 345 scenarios independently.

F.2 Complete Training and Inference Deliberation Depth Results

Table 16 lists the NAVSIMv2 scores across training and inference deliberation depths shown in Figure 4.

Table 16: NAVSIMv2 depth scaling (EPDMS).

Inference depth	$n_{\max} = 1$	$n_{\max} = 2$	$n_{\max} = 4$	$n_{\max} = 6$	$n_{\max} = 8$
1	46.63	47.23	49.54	48.74	47.61
2	—	48.96	50.43	52.93	49.85
3	—	—	51.08	52.65	50.16
4	—	—	51.52	53.01	50.78
5	—	—	—	53.22	50.94
6	—	—	—	53.24	51.16
7	—	—	—	—	51.18
8	—	—	—	—	51.19

F.3 Runtime Measurement Protocol and Results

Figure 4 reports medians of 100 timed forwards per configuration on one NVIDIA RTX 6000 Ada at batch size one, cycling through four identical cached NAVSIM scenes across methods. We use FP32 with TF32 enabled, 20 warm-up forwards. CUDA events with synchronization measure the complete forward, including visual feature extraction, planning, learned scoring, and final trajectory selection, but excluding input preprocessing, host-to-device transfer, data loading, and PDM evaluation. The ReCoursive-S–8 curve uses the $n_{\max} = 8$ checkpoint with cumulative $64n$ proposal selection, while TOAD uses the official DrivoR + SimScale checkpoint, 64 candidates and eight elites per CEM iteration. The measured DrivoR reference disables CEM in that same checkpoint. Table 17 lists the latency values plotted in Figure 4.

Table 17: Median model-forward latency (ms) on RTX 6000 Ada, batch size one. DrivoR with CEM disabled takes 97.02 ms.

Iteration count	ReCoursive-S–8 (depth)	DrivoR + TOAD (CEM)
1	97.13	99.66
2	99.71	100.18
3	102.04	102.71
4	105.51	105.03
5	109.87	108.07
6	114.95	111.15
7	120.40	114.94
8	125.09	117.92

G QUALITATIVE DELIBERATION EXAMPLES

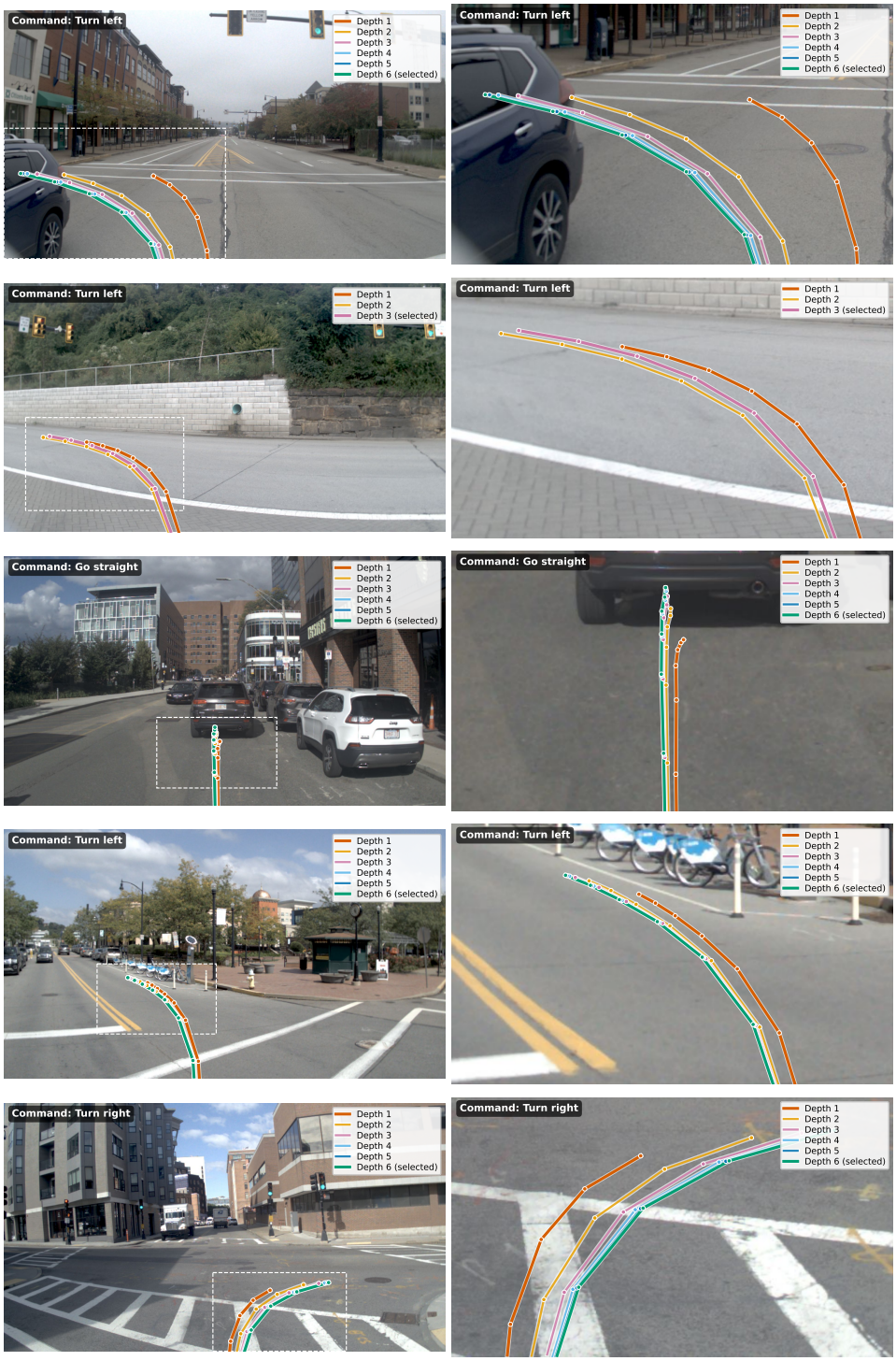

Figure 8: **Deliberation examples on NAVSIM (1/2).** Each row shows the full front-camera view (left) and the boxed region zoomed-in (right). Colored curves trace the same proposal slot of ReCoursive-L from depth 1 through the selected depth; dots mark 0.5-second waypoints over four seconds. The input driving command and selected depth are labeled.

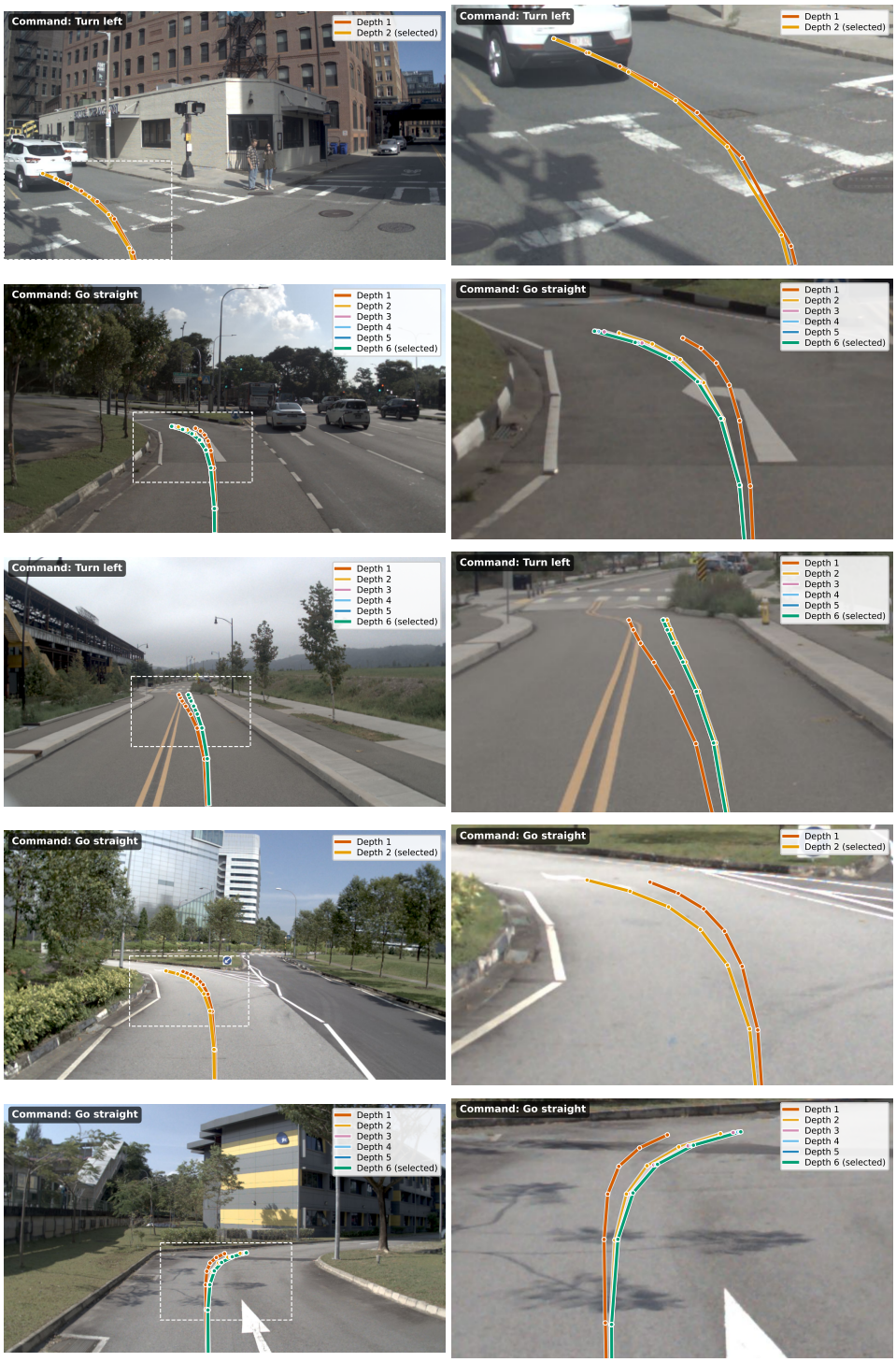

Figure 9: **Deliberation examples on NAVSIM (2/2).**

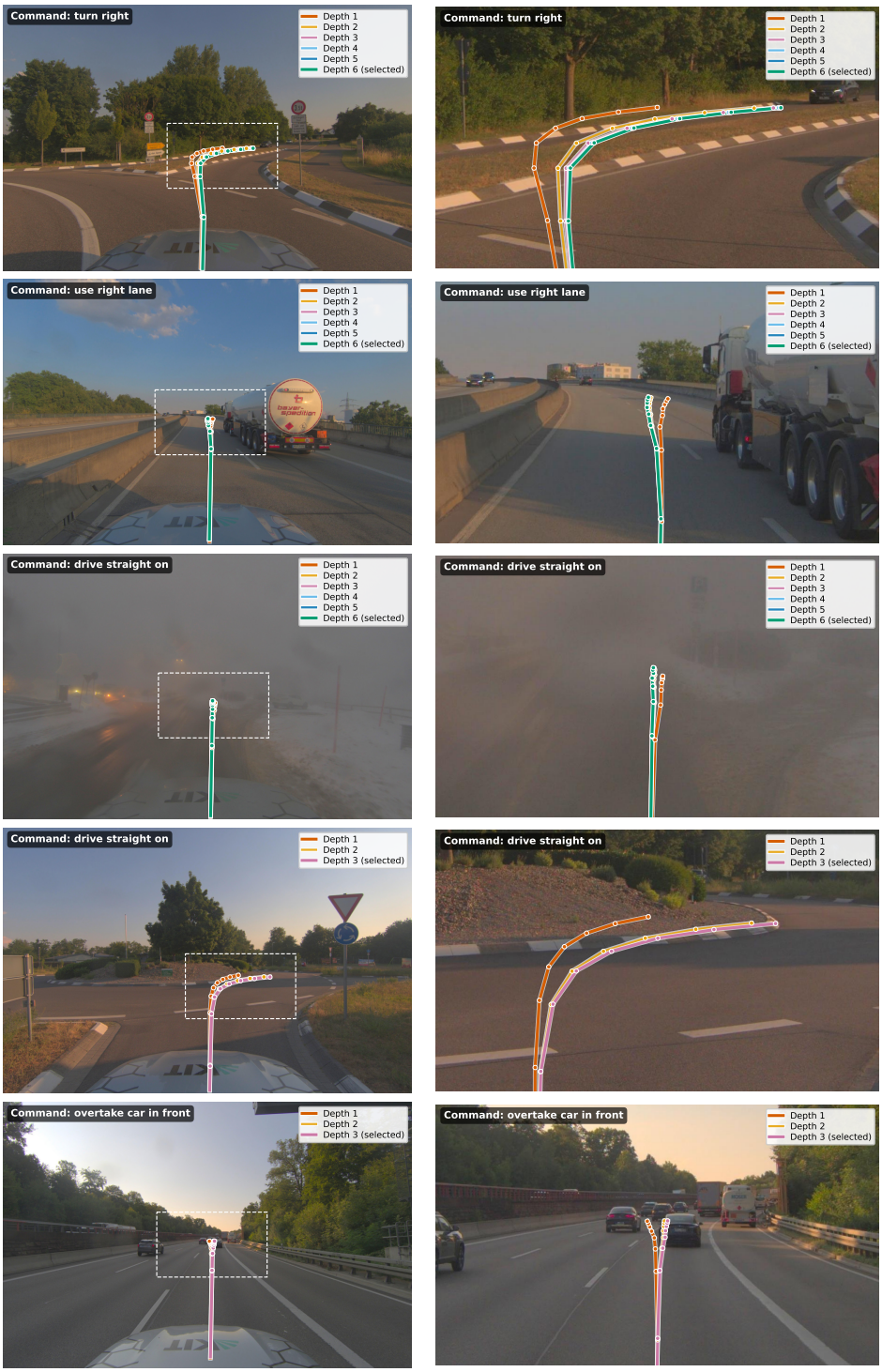

Figure 10: **Deliberation examples on KITScenes (1/2).**

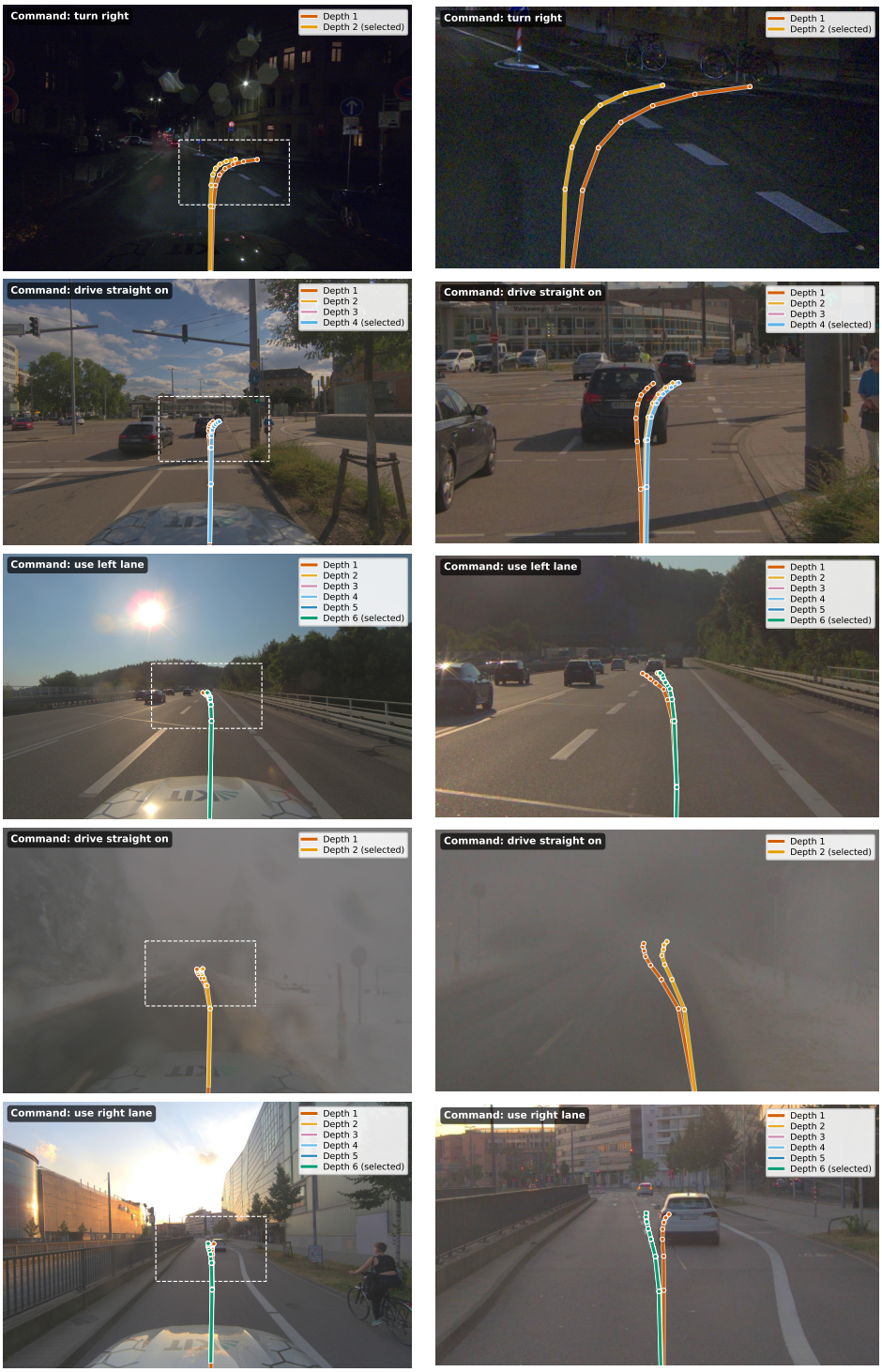

Figure 11: **Deliberation examples on KITScenes (2/2).**