CLOVER: CLOSED-LOOP VALUE ESTIMATION & RANKING FOR END-TO-END AUTONOMOUS DRIVING PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

End-to-end autonomous driving planners are commonly trained by imitating a single logged trajectory, yet they are evaluated by rule-based planning metrics that measure safety, feasibility, progress, and comfort. This creates a training–evaluation mismatch: trajectories close to the logged path may still violate planning rules, while alternative trajectories farther from the demonstration can remain valid and high-scoring. The mismatch is especially limiting for proposal-selection planners, whose performance depends on both candidate-set coverage and scorer ranking quality. We propose **CLOVER**, a **C**losed-**LO**op **V**alue **E**stimation and **R**anking framework for end-to-end driving planning. CLOVER first expands single-trajectory imitation into set-level proposal coverage by constructing evaluator-filtered pseudo-expert trajectories. It then performs conservative closed-loop self-distillation: a trajectory-level scorer is fitted to true evaluator sub-scores on generated proposals, while the generator is refined toward teacher-selected top-k and vector-Pareto proposal targets with stability regularization. We also analyze when an imperfect scorer can improve the generator, showing that scorer-mediated refinement is reliable under local scorer accuracy, conservative updates, and selected-set enrichment. On NAVSIM, CLOVER achieves 94.5 PDMS and 90.4 EPDMS, establishing a new state-of-the-art performance. On the more challenging NavHard split, it obtains 48.3 EPDMS, matching the strongest reported result. Supplementary evaluations on nuScenes and Bench2Drive cover open-loop planning and reactive closed-loop driving, respectively: CLOVER achieves the lowest nuScenes L2 error and collision rate among compared methods and reaches a 70.5% success rate and an 89.2 Driving Score on Bench2Drive. Code will be released.

1 INTRODUCTION

End-to-end autonomous driving planning has become a promising paradigm for mapping sensor observations directly to future ego trajectories (Hu et al., 2023a; Jiang et al., 2023; Chitta et al., 2022; Guo et al., 2025; Liao et al., 2025; Li et al., 2025c; Chen et al., 2024). However, planning is not a standard regression problem. Most planners are trained to imitate a single logged human trajectory, whereas they are evaluated by rule-based planning metrics such as NAVSIM PDMS/EPDMS (Dauner et al., 2024; Cao et al., 2025), which measure safety, feasibility, progress, and comfort. A trajectory close to the logged path may still collide, leave the drivable area, or induce unsafe interactions, while another trajectory farther from the demonstration may remain valid and high-scoring. This training–evaluation mismatch limits purely imitation-based planning.

The mismatch is particularly important for proposal-selection planners. These methods first generate multiple candidate trajectories and then select one using a learned scorer or ranker. Their final performance is constrained by two factors: whether the generator covers high-quality alternatives, and whether the scorer can identify them. Single-trajectory supervision can collapse candidates around one realized behavior, even when multiple decisions are valid in the same scene, such as maintaining speed, braking, or choosing a different interaction strategy. Conversely, even a strong candidate set can fail if the scorer ranks unsafe or infeasible trajectories above better alternatives. Thus, effective proposal-selection planning requires both broad proposal coverage and conservative use of scorer feedback.

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.

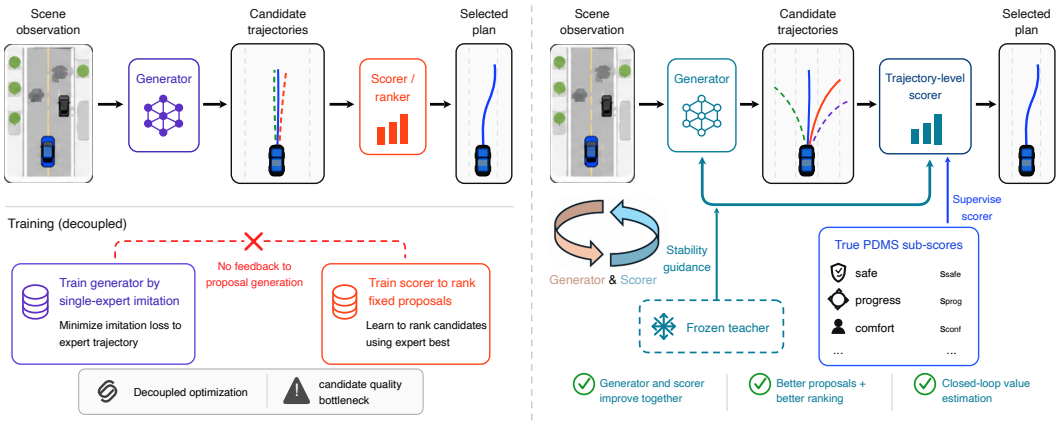

Figure 1: **CLOVER closes the loop between proposal generation and trajectory ranking. Left:** Conventional proposal-selection planners use a generator to produce candidate trajectories and a scorer/ranker to select the final plan. During training, however, proposal generation is often optimized by single-expert imitation and remains decoupled from ranking feedback. **Right:** CLOVER uses PDMS sub-scores to train a trajectory-level scorer and conservatively refines the generator through teacher-guided self-distillation. This closed-loop design improves proposal coverage and ranking together, increasing the chance that high-value trajectories are both generated and selected.

We propose **CLOVER**, a **C**losed-**LO**op **V**alue **E**stimation and **R**anking framework for proposal-selection based end-to-end autonomous driving planning. Given multi-view camera observations and ego state, CLOVER uses a lightweight generator to produce candidate future ego trajectories, and a trajectory-level scorer to predict planning-metric sub-scores for each candidate. At inference time, candidates are ranked by the predicted score and the top-ranked trajectory is selected. During training, the true rule-based evaluator supervises scorer learning and generator refinement, but is not required at deployment.

CLOVER addresses the training–evaluation mismatch in two stages. First, it expands single-trajectory imitation into **set-level proposal coverage**: evaluator-filtered pseudo-expert trajectories are constructed from diverse trajectory families, and the generator is trained to cover them while retaining the logged trajectory as a driving prior. This exposes the generator to multiple feasible driving modes and raises the oracle upper bound of the proposal set. Second, CLOVER performs **conservative closed-loop self-distillation**. We alternate fitting the scorer to true evaluator sub-scores on generated proposals with refining the student generator toward top-k and vector-Pareto targets from a teacher frozen within each refinement phase, using stability regularization. This lets the scorer guide generator improvement without unrestricted maximization of an imperfect learned reward.

A key question is when such scorer-mediated refinement is trustworthy. CLOVER does not require a globally perfect scorer. Our analysis shows that it is sufficient for scorer-selected targets to be statistically better under the true evaluator than the current proposal distribution. Under this selected-set enrichment condition, conservative set-level distillation increases the generator's probability mass on high-quality trajectories. This provides a direct condition under which an imperfect learned scorer can serve as a useful mediator for proposal refinement.

We evaluate CLOVER on NAVSIM v1, NAVSIM v2 EPDMS, and NavHard, with supplementary tests on nuScenes open-loop planning and Bench2Drive reactive closed-loop driving. CLOVER matches or surpasses the strongest reported results across the main NAVSIM-style benchmarks, achieving 94.5 PDMS and 90.4 EPDMS on NAVSIM and 48.3 EPDMS on NavHard, with competitive performance on nuScenes and Bench2Drive. Further analyses show that CLOVER improves both proposal quality and proposal diversity, indicating that its gains come from a stronger candidate distribution rather than only from selecting a better top-1 trajectory.

Our contributions are summarized as follows:

- We propose **CLOVER**, a closed-loop value estimation and ranking framework that couples proposal generation and trajectory scoring under rule-based planning feedback.

- We introduce evaluator-filtered pseudo-expert coverage supervision, extending single-trajectory imitation to set-level multi-modal proposal training and improving candidate-set quality.
- We develop conservative closed-loop self-distillation, alternating between scorer fitting on true planning sub-scores and teacher-guided generator refinement with top-k, vector-Pareto, and stability objectives; we further analyze when imperfect-scorer guidance is reliable under selected-set enrichment.

2 Related Work

End-to-end driving planners predict ego trajectories from sensor observations, but single-trajectory imitation is not fully aligned with rule-based planning metrics (Hu et al., 2023a; Jiang et al., 2023; Dauner et al., 2024). Proposal-selection planners use anchor sets, generative models, or proposal refinement to produce candidates and learned scorers to select among them (Li et al., 2024a; Liao et al., 2025; Guo et al., 2025; Kirby et al., 2026). Their effectiveness depends on both candidate coverage and ranking quality. CLOVER couples evaluator-filtered pseudo-expert coverage with sub-score estimation and conservative teacher-guided refinement, addressing limited supervision and the risk of exploiting scorer errors. Although its generator–scorer structure resembles actor–critic RL, CLOVER uses evaluator-guided ranking and self-distillation on offline logged scenes, without online environment interaction or exploration, rather than RL-style planning (Jiao et al., 2025). The full discussion is provided in Appendix A.

3 Method

Overview. Figure 2 summarizes CLOVER. CLOVER follows a proposal-selection formulation: a generator produces candidate ego trajectories, and a trajectory-level scorer ranks them by predicted planning-metric sub-scores. Training has two stages: Stage 1 expands proposal support with evaluator-filtered pseudo-expert coverage, and Stage 2 performs conservative closed-loop self-distillation by alternating scorer fitting and teacher-guided generator refinement.

3.1 Proposal Selection and Scoring

Given a driving observation o, CLOVER predicts a future ego trajectory $\tau = \{(x_t, y_t, \theta_t)\}_{t=1}^{T}$ in the ego-centric frame. Instead of outputting a single trajectory, the generator produces K candidates $\mathcal{T}_\theta(o) = G_\theta(o) = \{\tau_i\}_{i=1}^{K}$. A trajectory-level scorer predicts PDMS-related sub-scores $\hat{\mathbf{r}}_i = S_\phi(o, \tau_i)$, covering components such as collision avoidance, drivable-area compliance, time-to-collision, progress, and comfort. These sub-scores are composed by the official PDMS rule $\Gamma_{\mathrm{PDMS}}(\cdot)$, and inference selects $\tau^* = \arg\max_{\tau_i \in \mathcal{T}_\theta(o)} \Gamma_{\mathrm{PDMS}}(S_\phi(o, \tau_i))$. During training and analysis, the rule-based evaluator E assigns true targets $\mathbf{r}_i^{\mathrm{true}} = E(o, \tau_i)$. Since E involves map queries, future-obstacle states, geometric checks, and discrete rules, it is non-differentiable; CLOVER uses it only to supervise the scorer and construct distillation targets. Exact PDMS/EPDMS definitions are provided in Appendix C.

We instantiate this formulation with a lightweight DrivoR-style generator–scorer architecture (Kirby et al., 2026). A visual encoder maps the observation to compact scene tokens, which are combined with ego-state features and decoded by a Transformer trajectory generator with learnable trajectory queries. The scorer embeds each candidate trajectory through its waypoints, attends to the same scene tokens, and predicts planning-metric sub-scores. Thus, the generator determines which behaviors are available for selection, while the scorer estimates their closed-loop trajectory quality. We use $K = 64$ candidates unless otherwise specified; implementation details are in Appendix F.

3.2 Stage 1: Evaluator-filtered Proposal Coverage

Single-trajectory imitation treats the logged future trajectory as the only positive target, which is insufficient for training a high-coverage proposal generator. CLOVER therefore constructs a training-time pseudo-expert set $\mathcal{E}(o)$ for each scene. These pseudo-experts are not simple perturbations of the logged trajectory; they are generated from interpretable candidate families using privileged

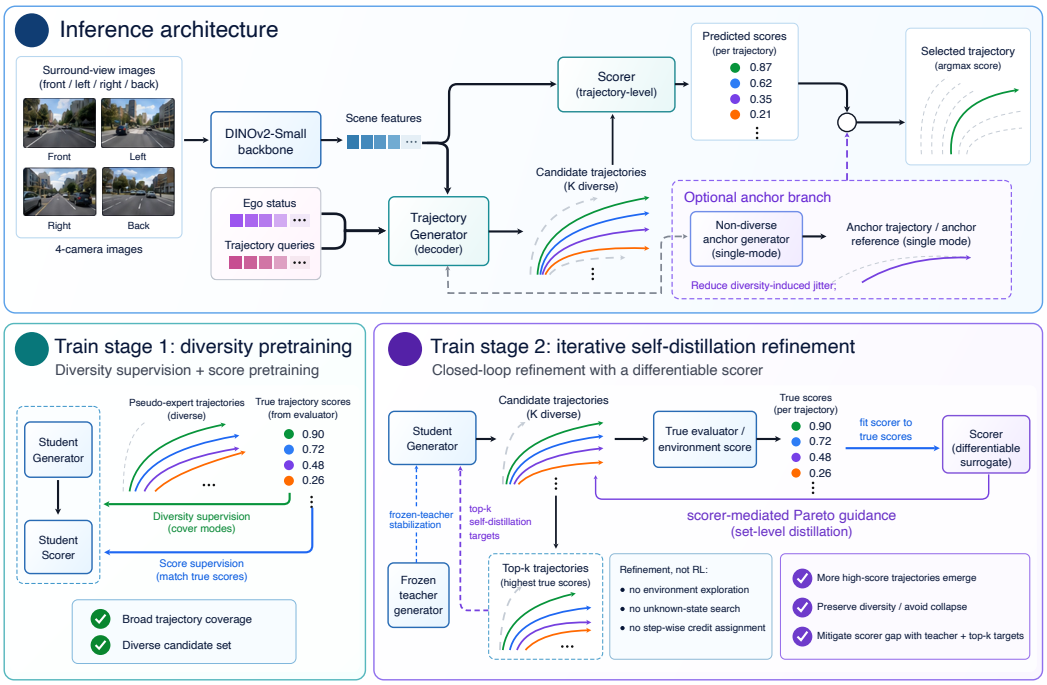

Figure 2: **Overview of CLOVER.** At inference time, a generator produces K candidate trajectories and a trajectory-level scorer ranks them by predicted planning-metric sub-scores. Stage 1 expands proposal coverage with evaluator-filtered pseudo-expert trajectories. Stage 2 fits the scorer to evaluator-provided sub-scores and refines the generator through teacher-guided top-k and vector-Pareto distillation with stability regularization.

metric-cache information, including route centerlines, drivable-area maps, and future obstacle occupancy. The candidate pool covers lateral offsets, acceleration/deceleration profiles, stop-go behaviors, approach-brake trajectories, boundary cases, and overshoot-style motions. Candidates are pre-filtered by lightweight drivable-area and occupancy checks, scored by the evaluator, and selected with a coverage-aware strategy over score patterns and bins.

During training, we further keep high-quality candidates above a score threshold and apply greedy farthest-point sampling in trajectory space to obtain at most M pseudo-experts, $\mathcal{E}(o) = \{e_j\}_{j=1}^{M}$. This set is used only during training. Detailed generation rules and examples are provided in Appendix B.

Given proposals $\mathcal{T}_\theta(o) = \{\tau_i\}_{i=1}^{K}$ and the logged trajectory τ^{gt}, Stage 1 optimizes

$$\mathcal{L}_{\mathrm{stage1}} = \lambda_{\mathrm{gt}} \underbrace{\min_i \|\tau_i - \tau^{\mathrm{gt}}\|_1}_{\mathcal{L}_{\mathrm{gt}}} + \underbrace{\frac{\lambda_{\mathrm{pe}}}{|\mathcal{E}(o)|} \sum_{e \in \mathcal{E}(o)} \min_i \|\tau_i - e\|_1}_{\mathcal{L}_{\mathrm{pe}}} + \underbrace{\frac{\lambda_{\mathrm{score}}}{|\mathcal{Q}(o)|} \sum_{\tau \in \mathcal{Q}(o)} \ell\big(S_\phi(o,\tau), E(o,\tau)\big)}_{\mathcal{L}_{\mathrm{score}}}. \tag{1}$$

Here, $\mathcal{Q}(o)$ denotes trajectory proposals, $\mathcal{L}_{\mathrm{gt}}$ preserves the logged driving prior, $\mathcal{L}_{\mathrm{pe}}$ encourages the proposal set to cover multiple pseudo-expert modes, and $\mathcal{L}_{\mathrm{score}}$ pretrains the scorer to predict sub-scores. For all scorer sub-score components, including EP, ℓ uses binary cross-entropy with logits (BCE-with-logits), with the evaluator's normalized scores in $[0, 1]$ as targets. EP retains its continuous soft target rather than being binarized; sigmoid outputs are used for score composition. The same loss is used in both training stages (Appendix F.1). This stage expands the candidate distribution beyond a single realized human trajectory while retaining stable imitation supervision.

3.3 Stage 2: Conservative Closed-Loop Self-Distillation

Stage 2 refines the expanded proposal distribution using evaluator feedback. Directly maximizing a learned scalar score can exploit scorer errors, so CLOVER uses an alternating procedure initialized

from Stage 1: a scorer phase fits the value estimator on current proposals, and a generator phase distills high-value teacher proposals under stability regularization.

Scorer phase. The current generator produces $\mathcal{T}_\theta(o) = \{\tau_i\}_{i=1}^K$, and the evaluator provides true sub-score targets $\mathbf{r}_i^{\text{true}} = E(o, \tau_i)$. The scorer minimizes $\mathcal{L}_{\text{critic}} = \frac{1}{K}\sum_{i=1}^K \ell(S_\phi(o, \tau_i), \mathbf{r}_i^{\text{true}})$, using the same component-wise BCE-with-logits objective as Stage 1 to fit the evaluator on the current proposal distribution.

Generator phase. A frozen teacher generates $\mathcal{T}^{\text{tea}}(o) = \{\tau_i^{\text{tea}}\}_{i=1}^K$. The scorer evaluates these proposals and constructs two target sets: $\mathcal{K}^{\text{tea}}$, the top-k proposals under $\Gamma_{\text{PDMS}}(S_\phi(o, \tau))$, and $\mathcal{P}^{\text{tea}}$, the non-dominated proposals in predicted sub-score space. A trajectory τ_a dominates τ_b if $r_a^m \geq r_b^m$ for all components m and $r_a^m > r_b^m$ for at least one component. The student proposal set $\mathcal{T}^{\text{stu}}(o) = \{\tau_i^{\text{stu}}\}_{i=1}^K$ is trained to cover these targets with

$$\mathcal{L}_{\text{set}}(\mathcal{T}^{\text{stu}}, \mathcal{A}) = \frac{1}{|\mathcal{A}|}\sum_{\tau\in\mathcal{A}} \min_i \|\tau_i^{\text{stu}} - \tau\|_1, \quad \mathcal{A} \in \{\mathcal{K}^{\text{tea}}, \mathcal{P}^{\text{tea}}\}. \tag{2}$$

The generator objective is

$$\mathcal{L}_{\text{gen}} = \lambda_{\text{traj}}\mathcal{L}_{\text{gt}} + \lambda_{\text{topk}}\mathcal{L}_{\text{set}}(\mathcal{T}^{\text{stu}}, \mathcal{K}^{\text{tea}}) + \lambda_{\text{pareto}}\mathcal{L}_{\text{set}}(\mathcal{T}^{\text{stu}}, \mathcal{P}^{\text{tea}}) + \lambda_{\text{stab}}\frac{1}{K}\sum_{i=1}^K \|\tau_i^{\text{stu}} - \text{sg}(\tau_i^{\text{tea}})\|_1. \tag{3}$$

The top-k term concentrates probability mass on high-scoring proposals, the vector-Pareto term preserves safety–progress–comfort trade-offs, and the stability term keeps refinement conservative.

3.4 Inference and Optional Anchor Reranking

Without the optional anchor described below, inference uses only the generator and learned scorer: candidates are generated, scored, and ranked by the composed predicted metric score. The evaluator, pseudo-expert generator, and self-distillation procedure are used only during training and analysis.

For NAVSIM v2 EPDMS, extended comfort (EC) measures consistency across adjacent planning decisions; independently selecting among near-tied high-quality candidates can therefore incur a temporal-consistency penalty. We optionally use the single-mode planner from our Stage-2-only ablation, which omits Stage 1, as a soft anchor. It uses the same observation inputs and base architecture, without future information, evaluator scores, or previous-frame predictions at inference. Candidates are reranked by combining the learned score with position and heading deviation penalties relative to the anchor. This is a consistency–progress trade-off; the formula, checkpoint provenance, and additional controls are provided in Appendix D.

4 Theoretical Analysis

We provide a concise theoretical justification for when CLOVER's scorer-mediated refinement can improve the generator. The central issue is not whether the learned scorer is globally perfect, but whether the scorer-selected targets are statistically better under the true evaluator than the current proposal distribution. Full proofs, auxiliary results on approximate monotonicity, scorer refitting, and Pareto consistency, and additional empirical validation are provided in Appendix E.

All scores are normalized to $[0, 1]$. For a scene o, let $\mu_t^o = \frac{1}{K}\sum_{i=1}^K \delta_{\tau_i^t(o)}$ denote the empirical distribution of the current K proposals, and let $A_t(o)$ be the scorer-selected target set, e.g., top-k or vector-Pareto teacher proposals. Let ν_t^o be the empirical distribution over $A_t(o)$. For a high-score region $\mathcal{H}_o = \{\tau : R^*(o, \tau) \geq r_{\text{high}}\}$, define $p_t(o) = \mu_t^o(\mathcal{H}_o)$ and $q_t(o) = \nu_t^o(\mathcal{H}_o)$.

Theorem 1 (Selected-set enrichment improves high-score support) *Assume that the scorer-selected target set is enriched with true high-score trajectories:*

$$q_t(o) \geq p_t(o) + \xi_t(o), \qquad \xi_t(o) > 0.$$

Assume that the generator update is conservative, i.e.,

$$\text{TV}\big(\mu_{t+1}^o, (1-\alpha_t)\mu_t^o + \alpha_t\nu_t^o\big) \leq \eta_t(o), \qquad \alpha_t \in [0, 1].$$

Then

$$p_{t+1}(o) \geq p_t(o) + \alpha_t \xi_t(o) - \eta_t(o).$$

Therefore, if $\alpha_t \xi_t(o) > \eta_t(o)$, Stage-2 refinement increases the probability mass assigned to high-score trajectories. Similarly, if the selected targets have higher true expected score, $\mathbb{E}_{\nu_t^o}[R^*] - \mathbb{E}_{\mu_t^o}[R^*] \geq \beta_t(o) > 0$, then

$$\mathbb{E}_{\mu_{t+1}^o}[R^*] \geq \mathbb{E}_{\mu_t^o}[R^*] + \alpha_t \beta_t(o) - \eta_t(o).$$

Implication for CLOVER. The theorem states a weak and testable condition: the scorer does not need to rank every trajectory correctly; it only needs to select target sets that are statistically enriched with true high-quality trajectories. This matches CLOVER's design. Stage 1 provides broad proposal support, the scorer phase calibrates the value estimator on generated proposals, and the generator phase conservatively distills scorer-selected top-k and Pareto targets rather than directly maximizing a learned scalar reward.

Empirical support. We empirically validate the two premises behind Theorem 1: the Stage-1 proposal pool must contain high-quality modes, and the scorer-selected targets must be statistically enriched under the true evaluator. On 12,146 NAVSIM Test scenes with 64 proposals per scene, the pooled proposal distribution already contains substantial high-score support, with 35.42% full-score proposals. More importantly, high-confidence scorer targets are much more likely to be truly high scoring: among proposals with predicted score $s \geq 0.95$, the true mean score is 0.9753, $P(R^* \geq 0.90) = 96.69\%$, and $P(R^* = 1) = 69.74\%$. Thus, for the full-score region $\mathcal{H}_o = \{\tau : R^*(o, \tau) = 1\}$, the empirical enrichment gap is approximately $69.74\% - 35.42\% = 34.32\%$, directly matching the condition $q_t(o) \geq p_t(o) + \xi_t(o)$. The scorer also separates high- and low-quality proposals reliably: for fixed groups with true score ≥ 0.95 versus ≤ 0.50, pairwise ranking accuracy reaches 94.72%, and the elite/reject margin condition $\gamma_o > 2\epsilon_{p90,o}$ holds in 82.37% of scenes. Finally, scorer-selected Top-8 and Top-16 target sets achieve mean best true scores of 0.9656 and 0.9730, respectively, supporting the expected-score version of the theorem. These results indicate that Stage 2 does not rely on a globally perfect scorer or on discovering high-quality modes from scratch; rather, it reallocates generator probability mass toward evaluator-enriched targets selected from an already broad proposal space. Additional empirical validation is provided in Appendix E.

5 Experiments

5.1 Experimental Setup

Our main experiments evaluate CLOVER on NAVSIM v1, NAVSIM v2 EPDMS, and NavHard, supplemented by nuScenes open-loop planning and Bench2Drive reactive closed-loop evaluation. Our main evaluation follows NAVSIM-style planning metrics, where trajectory quality is measured by rule-based safety, progress, and comfort components rather than only displacement error to the logged trajectory. NAVSIM v1 reports PDMS, while NAVSIM v2 reports EPDMS on the `navtest` split (Dauner et al., 2024; Cao et al., 2025). We further evaluate on the more challenging `navhard-two-stage` split using EPDMS. For supplementary open-loop validation on nuScenes (Caesar et al., 2020), we follow the ST-P3 and UniAD protocols and report L2 displacement error and collision rate (Hu et al., 2022b; 2023b). For Bench2Drive (Jia et al., 2024), we report Success Rate (SR) and Driving Score (DS); the adaptation and evaluation results are detailed in Appendix K. Exact PDMS/EPDMS definitions are provided in Appendix C.

Our main NAVSIM experiments use the same DrivoR-style generator–scorer architecture (Kirby et al., 2026). The model uses 4-camera images without LiDAR, a DINOv2 ViT-S visual encoder with LoRA fine-tuning (Oquab et al., 2024; Hu et al., 2022a), 64 trajectory proposals, and a 4-second horizon sampled every 0.5 seconds. Stage 1 is trained with pseudo-expert coverage and scorer pretraining, and Stage 2 is initialized from Stage 1 and trained with alternating scorer-fitting and generator-refinement phases. CLOVER retains the base DrivoR-style generator–scorer architecture; the optional NAVSIM-v2 anchor setting additionally runs a separate reference planner (Section 3.4). With $K = 64$ proposals, the deployed model runs at about 110 ms per scene on a single NVIDIA A100 GPU. Detailed architecture, hyperparameters, and training cost are provided in Appendix F.

Table 1: **NAVSIM v1 benchmark.** Higher is better. CLOVER achieves the best PDMS and ranks first or second across all sub-scores.

Method	NC	DAC	TTC	Comf.	EP	**PDMS**
Human driver (Dauner et al., 2024)	100	100	100	99.9	87.5	94.8
Ego-stat. MLP (Dauner et al., 2024)	93.0	77.3	83.6	100	62.8	65.6
UniVLA (Wang et al., 2025)	96.9	91.1	91.7	96.7	76.8	81.7
UniAD (Hu et al., 2023a)	97.8	91.9	92.9	100	78.8	83.4
DriveX-S (Shi et al., 2025)	97.5	94.0	93.0	100	79.7	84.5
World4Drive (Zheng et al., 2025)	97.4	94.3	92.8	100	79.9	85.1
DRAMA (Yuan et al., 2024)	98.0	93.1	94.8	100	80.1	85.5
VAD-v2 (Chen et al., 2024)	98.1	94.8	94.3	100	80.6	86.2
DiffusionDrive (Liao et al., 2025)	98.2	96.2	94.7	100	82.2	88.1
AutoVLA (Zhou et al., 2025b)	98.4	95.6	**98.0**	99.9	81.9	89.1
DriveVLA-W0 (Li et al., 2025b)	98.7	**99.1**	95.3	99.3	83.3	90.2
ReCogDrive (Li et al., 2025c)	97.9	97.3	94.9	100	87.3	90.8
Hydra-MDP++ (Li et al., 2025a)	98.6	98.6	95.1	100	85.7	91.0
iPad (Guo et al., 2025)	98.6	98.3	94.9	100	88.0	91.7
DriveSuprim (Yao et al., 2025)	98.6	98.6	95.5	100	91.3	93.5
DrivoR (Kirby et al., 2026)	99.0	98.9	96.7	100	90.0	93.7
CLOVER (ours)	**99.1**	99.0	96.9	**100**	**91.7**	**94.5**

Table 2: **Stage-wise proposal quality and diversity.** Stage 1 expands proposal support, while Stage 2 improves proposal quality and preserves higher diversity than the baseline.

Metric	Base	Stage 1	Stage 2
Selected ↑	0.9369	0.9413	**0.9448**
Oracle@64 ↑	0.9933	**0.9976**	0.9939
Gap ↓	0.0564	0.0563	**0.0491**
Mean@64 ↑	0.7972	0.7374	**0.8277**
Std@64	0.2152	**0.3013**	0.2235
Count>0.95 ↑	30.94	24.78	**31.97**
Count<0.50 ↓	9.05	12.21	**6.83**
Pairwise ADE ↑	1.80	**5.97**	5.20
Pairwise FDE ↑	4.47	**10.70**	8.25
Endpoint area ↑	5.78	**46.18**	25.56
Eff. rank ↑	1.14	1.24	**1.27**
Q-Clust.@2m ↑	6.02	**10.91**	8.71
Top-6 ADE ↑	0.91	**2.11**	1.75
Top-6 FDE ↑	2.20	**3.55**	2.59

Table 3: **NAVSIM v2 EPDMS on navtest.** Comparison under the Extended PDMS metric. EPDMS* denotes results computed with the original NAVSIM v2 evaluation code before applying the human-behavior filtering fix, while EPDMS denotes results computed with the updated official implementation. Higher is better for all columns.

Method	NC ↑	DAC ↑	DDC ↑	TLC ↑	EP ↑	TTC ↑	LK ↑	HC ↑	EC ↑	EPDMS* ↑	EPDMS ↑
Ego-MLP (Li et al., 2024b)	93.1	77.9	92.7	99.6	86.0	91.5	89.4	**98.3**	85.4	64.0	–
Transfuser (Chitta et al., 2022)	96.9	89.9	97.8	99.7	87.1	95.4	92.7	**98.3**	87.2	76.7	–
ARTEMIS (Feng et al., 2025b)	98.3	95.1	98.6	99.8	81.5	97.4	96.5	**98.3**	**98.3**	83.1	–
DiffusionDriveV2 (Zou et al., 2025)	97.7	96.6	99.2	99.8	88.9	97.2	96.0	97.8	91.0	85.5	87.5
Hydra-MDP++ (Li et al., 2025a)	98.4	98.0	99.4	99.8	87.5	97.7	95.3	**98.3**	77.4	85.1	–
DriveSuprim (Yao et al., 2025)	97.8	97.9	99.5	**99.9**	90.6	97.1	96.6	**98.3**	77.9	86.0	–
SparseDriveV2 (Sun et al., 2026)	98.1	98.1	**99.6**	99.8	**91.1**	97.3	**96.9**	98.2	78.4	86.7	90.1
CLOVER(ours)	**99.4**	**99.3**	99.5	99.8	86.9	**98.9**	95.2	98.2	75.5	**87.2**	**90.4**

5.2 Main Results

Tables 1, 3, 4, and 5 summarize results on NAVSIM v1, NAVSIM v2 `navtest`, NavHard, and supplementary nuScenes open-loop evaluation. CLOVER achieves **94.5 PDMS** on NAVSIM v1, improving over the strongest generator–scorer baseline DrivoR (Kirby et al., 2026) by 0.8 points and ranking first or second across all PDMS sub-scores. Under NAVSIM v2, it obtains **90.4 EPDMS** with the updated official implementation and **87.2 EPDMS*** with the original evaluation code. On the challenging `navhard-two-stage` split, CLOVER reaches **48.3 EPDMS**, matching the strongest reported result. It also achieves strong supplementary nuScenes open-loop results, including the lowest ST-P3 L2 error and best or tied-best collision rate.

On supplementary **Bench2Drive** reactive closed-loop evaluation, Stage 2 reaches **70.5% SR** and **89.2 DS**, compared with 68.6% SR and 86.8 DS after Stage 1 (+1.9 percentage points in SR and +2.4 DS); detailed results are provided in Appendix K. Three-seed repeatability results for NAVSIM are provided in Appendix H.

5.3 Proposal Quality and Diversity Analysis

A central goal of CLOVER is to improve the proposal set itself, rather than only rerank a fixed set of candidates. We therefore analyze the 64 generated proposals across three stages: the DrivoR baseline, Stage 1 pseudo-expert diversity pretraining, and Stage 2 self-distillation refinement. All statistics are computed on 12,146 common scenes, using true PDMS scores from the evaluator for analysis only. Following the proposal-diversity protocol of Curious-VLA (Chen et al., 2026), we report pairwise ADE/FDE and additional proposal-set quality metrics in Table 2.

Table 4: **NAVSIM v2 `navhard-two-stage`.** Comparison on the challenging NavHard split using EPDMS. Stage-1 and Stage-2 columns report sub-scores from the two-stage evaluation procedure, and EPDMS denotes the final aggregated score. Higher is better for all columns.

	Stage 1									Stage 2								
Method	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	EPDMS
RAP-DINO (Feng et al., 2025a)	97.1	94.4	98.8	99.8	83.9	96.9	94.7	96.4	66.2	83.2	83.9	87.4	98.0	86.9	80.4	52.3	95.2	52.4	39.6
GTRS-D (Li et al., 2025d)	98.9	96.2	99.4	99.3	72.9	98.9	95.1	96.9	39.1	91.2	89.4	94.4	98.8	69.5	90.0	54.3	94.0	48.7	45.0
GTRS-A (Li et al., 2025d)	98.9	95.1	99.1	99.6	76.2	99.1	94.9	97.6	54.2	88.1	88.8	89.3	98.9	98.9	85.9	53.7	96.8	56.9	45.4
Drivor (Kirby et al., 2026)	98.8	95.1	98.9	100	72.6	98.7	94.0	97.6	73.3	90.2	88.4	91.9	98.6	70.0	88.0	50.1	98.5	76.2	48.3
CLOVER (ours)	99.4	98.0	99.9	100	75.7	98.9	94.4	97.1	56.0	89.8	88.2	93.2	99.0	74.2	87.7	52.6	97.0	57.3	48.3

Table 5: **nuScenes open-loop validation.** Lower is better.

Method	ST-P3		UniAD	
	L2 ↓	Col. ↓	L2 ↓	Col. ↓
ST-P3 (Hu et al., 2022b)	2.11	0.71	–	–
VAD (Jiang et al., 2023)	0.37	0.14	–	–
UniAD (Hu et al., 2023b)	0.69	0.12	1.03	0.31
EMMA (Hwang et al., 2024)	0.32	–	–	–
AutoVLA (Zhou et al., 2025b)	0.48	0.13	0.86	0.35
Impromptu VLA (Chi et al., 2025)	0.33	0.13	0.67	0.38
OpenDriveVLA (Zhou et al., 2025a)	0.33	**0.10**	0.67	**0.30**
CLOVER (ours)	**0.31**	**0.10**	**0.65**	**0.30**

Table 6: **Anchor-assisted soft reranking on NAVSIM v2 EPDMS.** *: without anchor.

Inference setting	EC ↑	EPDMS ↑
Single-mode anchor	77.6	88.7
clover*	35.0	89.3
clover	75.5	**90.4**

Table 2 shows a clear stage-wise behavior. Stage 1 greatly expands the proposal space: compared with the baseline, it increases Oracle@64 from 0.9933 to 0.9976, Pairwise ADE from 1.80 to 5.97, Pairwise FDE from 4.47 to 10.70, and Endpoint Area from 5.78 to 46.18. This confirms that pseudo-expert coverage supervision raises the oracle upper bound, although it also introduces a heavier low-score tail.

Stage 2 then refines this expanded distribution. It achieves the best scorer-selected PDMS, improving from 0.9369 for the baseline and 0.9413 for Stage 1 to 0.9448. It also obtains the highest mean proposal score, 0.8277, and reduces the number of low-score proposals with PDMS<0.50 to 6.83. At the same time, it preserves substantially more diversity than the baseline, with Pairwise ADE 5.20 versus 1.80 and qualified cluster count 8.71 versus 6.02.

These results support the intended division of labor: Stage 1 expands proposal support, while Stage 2 converts the expanded space into a higher-quality and better-ranked proposal distribution. Additional qualitative comparisons and scorer-only diagnostic experiments are provided in Appendices I.1 and G.

5.4 Ablation Studies

We ablate the main components of CLOVER on NAVSIM v1 PDMS and NAVSIM v2 EPDMS. The ablations examine four design choices: pseudo-expert proposal coverage, alternating closed-loop refinement, vector-Pareto teacher-set construction, and anchor-assisted reranking.

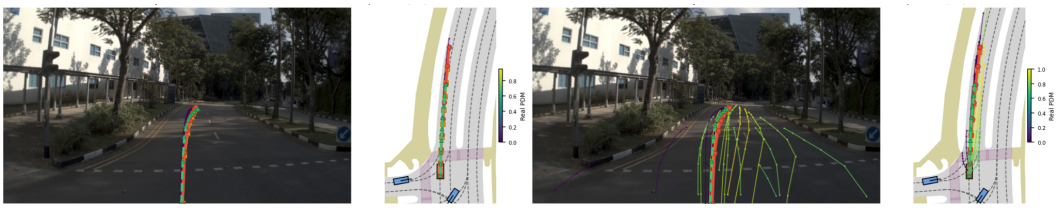

Figure 3: **Qualitative proposal diversity.** Compared with the DrivoR baseline, which concentrates candidates around a narrow mode, CLOVER covers a wider set of feasible trajectory branches. Many of these diverse candidates remain high-scoring under the evaluator, showing that the additional diversity is not merely caused by low-quality outliers.

Table 7: **Main component ablation.**

Setting	Div.	Ref.	PDMS ↑
Single-expert	–	–	93.7
Diversity only	✓	–	94.1
Closed-loop only	–	✓	93.8
CLOVER full	✓	✓	**94.5**

Table 8: **Stage-2 refinement on NAVSIM v1.**

Stage-2 variant	PDMS ↑
Non-iterative generator/scorer update	< 10
Self-distillation only, 64/32/16/8 traj.	94.0/93.8/93.8/93.9
Full alternating refinement	**94.5**

Table 9: **Ablation of teacher-set construction.**

Teacher target construction	PDMS ↑	Q-Cluster@2m ↑	Top-6 ADE ↑	Top-6 FDE ↑
Real-PDMS top-k	93.9	6.45	1.04	2.31
Real-PDMS top-k + distance suppression	94.3	7.49	1.66	2.47
Vector-Pareto targets (ours)	**94.5**	**8.71**	**1.75**	**2.59**

Main components. Table 7 shows that the two stages are complementary. The single-expert baseline obtains 93.7 PDMS. Adding pseudo-expert diversity supervision improves the score to 94.1, while applying closed-loop refinement without the diversity stage yields only 93.8. The full model reaches 94.5, suggesting that scorer-guided refinement is most effective when Stage 1 has already expanded the proposal support.

Stage-2 refinement design. Table 8 shows that non-iterative generator/scorer updating collapses (< 10 PDMS), while self-distillation alone remains stable but saturates at 93.8–94.0. Full alternating refinement reaches 94.5, supporting repeated scorer refitting before generator updates.

Teacher-set construction. Table 9 compares Stage-2 teacher targets. Scalar real-PDMS top-k yields 93.9 PDMS and relatively low proposal diversity; distance suppression improves both, but vector-Pareto targets reach 94.5 with the highest high-quality diversity, supporting the preservation of safety–progress–comfort trade-offs. On 12,146 held-out NAVSIM-v1 scenes, scalar Top-8 leaves 23.26% of retained Pareto targets uncovered under the same eight-target budget, with at least one additional target in 80.92% of scenes (Appendix J).

Anchor-assisted reranking. Table 6 evaluates the optional anchor-assisted soft reranking used for NAVSIM v2 EPDMS. The anchor uses our *Closed-loop only* checkpoint (Table 7): Stage 2 without Stage 1, scoring 93.8 PDMS and 88.7 EPDMS. It differs from official DrivoR (87.4 EPDMS).[1] CLOVER without an anchor already reaches 89.3, a 1.9-point gain over that DrivoR evaluation. Anchoring raises EC from 35.0 to 75.5 and EPDMS to 90.4, while reducing EP from 92.4 to 86.9 (Table 13). Appendix D.1 examines alternative anchors, baseline controls, and temporal selection.

Number of proposals. Using $K = 32$, 64, and 96 proposals yields 93.9, 94.5, and 94.4 PDMS, respectively; $K = 64$ provides the best performance and is our default.

6 Conclusion

We presented **CLOVER**, which addresses the training–evaluation mismatch through evaluator-filtered proposal coverage and conservative closed-loop self-distillation. A scorer fitted to evaluator sub-scores guides generator refinement through top-k and vector-Pareto targets with stability regularization. Our analysis establishes sufficient conditions for imperfect-scorer guidance to increase high-quality proposal support: local scorer accuracy, conservative updates, and selected-set enrichment. Experiments on NAVSIM v1, NAVSIM v2 EPDMS, NavHard, and nuScenes show strong planning performance and improvements in both final selection and the underlying candidate distribution. Supplementary Bench2Drive experiments also show gains from Stage-2 refinement in reactive closed-loop driving.

[1] We obtain 87.4 by inference with the official weights, close to PriorEye's 87.2 (Yeon et al., 2026). DrivoR reports no official `navtest` EPDMS, so this result is excluded from Table 3.

AI USE STATEMENT

We used generative AI tools to polish the language of the initial manuscript, and the authors reviewed the revised text for accuracy and consistency with the underlying work. Generative AI was also used to create an early version of Figure 2; the authors subsequently iterated on the design and redrew the entire figure as vector graphics. The authors reviewed the AI-assisted text and the final redrawn figure. We take responsibility for the final content of this work, including all text and visual materials developed with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We will publicly release the complete code for CLOVER. Appendix E provides the mathematical assumptions and full proofs supporting our theoretical analysis. Appendix B details pseudo-expert construction, and Appendix D specifies anchor-assisted reranking and reports the parameter-search results. Appendix F, including Tables 15–19, documents the architecture, training hyperparameters, pseudo-expert configuration, hardware, and training time. Appendix K describes the supplementary Bench2Drive adaptation, local supervision, and trajectory-to-control conversion. These materials, together with the evaluation definitions in Appendix C, support reproduction of the method and experiments.

REFERENCES

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

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.

Canyu Chen, Yuguang Yang, Zhewen Tan, Yizhi Wang, Ruiyi Zhan, Haiyan Liu, Xuanyao Mao, Jason Bao, Xinyue Tang, Linlin Yang, Bingchuan Sun, Yan Wang, and Baochang Zhang. Devil is in narrow policy: Unleashing exploration in driving vla models, 2026. URL https://arxiv.org/abs/2603.06049.

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

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

Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):12878–12895, 2022.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, et al. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *NeurIPS*, 2024.

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

Renju Feng, Ning Xi, Duanfeng Chu, Rukang Wang, Zejian Deng, Anzheng Wang, Liping Lu, Jinxiang Wang, and Yanjun Huang. Artemis: Autoregressive end-to-end trajectory planning with mixture of experts for autonomous driving. *arXiv preprint arXiv:2504.19580*, 2025b.

Ke Guo, Haochen Liu, Xiaojun Wu, Jia Pan, and Chen Lv. ipad: Iterative proposal-centric end-to-end autonomous driving. *arXiv preprint arXiv:2505.15111*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022a.

Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *European Conference on Computer Vision*, pp. 533–549. Springer, 2022b.

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*, 2023a.

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 *CVPR*, pp. 17853–17862, 2023b.

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

Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. *Advances in Neural Information Processing Systems*, 37:819–844, 2024.

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.

Siwen Jiao, Kangan Qian, Hao Ye, Yang Zhong, Ziang Luo, Sicong Jiang, Zilin Huang, Yangyi Fang, Jinyu Miao, Zheng Fu, et al. Evadrive: Evolutionary adversarial policy optimization for end-to-end autonomous driving. *arXiv preprint arXiv:2508.09158*, 2025.

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

Kailin Li, Zhenxin Li, Shiyi Lan, Yuan Xie, Zhizhong Zhang, Jiayi Liu, Zuxuan Wu, Zhiding Yu, and Jose M Alvarez. Hydra-mdp++: Advancing end-to-end driving via expert-guided hydra-distillation. *arXiv e-prints*, pp. arXiv–2503, 2025a.

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, et al. Drivevla-w0: World models amplify data scaling law in autonomous driving. *arXiv preprint arXiv:2510.12796*, 2025b.

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

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

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. *arXiv preprint arXiv:2506.06664*, 2025d.

Zhiqi Li, Zhiding Yu, Shiyi Lan, Jiahan Li, Jan Kautz, Tong Lu, and Jose M Alvarez. Is ego status all you need for open-loop end-to-end autonomous driving? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14864–14873, 2024b.

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 *CVPR*, pp. 12037–12047, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *TMLR*, 2024.

Chen Shi, Shaoshuai Shi, Kehua Sheng, Bo Zhang, and Li Jiang. Drivex: Omni scene modeling for learning generalizable world knowledge in autonomous driving. In *ICCV*, 2025.

Wenchao Sun, Xuewu Lin, Keyu Chen, Zixiang Pei, Xiang Li, Yining Shi, and Sifa Zheng. Sparsedrivev2: Scoring is all you need for end-to-end autonomous driving, 2026. URL https://arxiv.org/abs/2603.29163.

Linhan Wang, Zichong Yang, Chen Bai, Guoxiang Zhang, Xiaotong Liu, Xiaoyin Zheng, Xiao-Xiao Long, Chang-Tien Lu, and Cheng Lu. Drive-jepa: Video jepa meets multimodal trajectory distillation for end-to-end driving. *arXiv preprint arXiv:2601.22032*, 2026.

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

Zebin Xing, Xingyu Zhang, Yang Hu, Bo Jiang, Tong He, Qian Zhang, Xiaoxiao Long, and Wei Yin. Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving. In *CVPR*, 2025.

Wenhao Yao, Zhenxin Li, Shiyi Lan, Zi Wang, Xinglong Sun, Jose M Alvarez, and Zuxuan Wu. Drivesuprim: Towards precise trajectory selection for end-to-end planning. *arXiv preprint arXiv:2506.06659*, 2025.

Kyuhwan Yeon, Benjamin Ramtoula, and Daniele De Martini. Prioreye: Geospatial visual priors for end-to-end autonomous driving. In *European Conference on Computer Vision*, pp. 670–688. Springer, 2026.

Chengran Yuan, Zhanqi Zhang, Jiawei Sun, Shuo Sun, Zefan Huang, Christina Dao Wen Lee, Dongen Li, Yuhang Han, Anthony Wong, Keng Peng Tee, et al. Drama: An efficient end-to-end motion planner for autonomous driving with mamba. *arXiv preprint arXiv:2408.03601*, 2024.

Yupeng Zheng, Pengxuan Yang, Zebin Xing, Qichao Zhang, Yuhang Zheng, Yinfeng Gao, Pengfei Li, Teng Zhang, Zhongpu Xia, Peng Jia, et al. World4drive: End-to-end autonomous driving via intention-aware physical latent world model. In *ICCV*, 2025.

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. *arXiv preprint arXiv:2503.23463*, 2025a.

Zewei Zhou, Tianhui Cai, Seth Z Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *arXiv preprint arXiv:2506.13757*, 2025b.

Jialv Zou, Shaoyu Chen, Bencheng Liao, Zhiyu Zheng, Yuehao Song, Lefei Zhang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Diffusiondrivev2: Reinforcement learning-constrained truncated diffusion modeling in end-to-end autonomous driving. *arXiv preprint arXiv:2512.07745*, 2025.

A Extended Related Work

A.1 End-to-end Autonomous Driving Planning

End-to-end autonomous driving planning aims to predict future ego trajectories directly from sensor observations. Early methods typically mapped front-view images to low-level control commands, while recent approaches use multi-view encoders, Transformer decoders, BEV representations, trajectory queries, and vision pretraining to unify scene understanding and planning (Hu et al., 2023a; Jiang et al., 2023; Chitta et al., 2022; Guo et al., 2025; Liao et al., 2025; Li et al., 2025c; Chen et al., 2024). These methods have significantly improved planning performance, especially under modern evaluation protocols such as NAVSIM (Dauner et al., 2024; Cao et al., 2025). However, most end-to-end planners are still trained primarily by imitating logged expert trajectories. Such supervision is stable and data-efficient, but it is not fully aligned with rule-based planning metrics that evaluate safety, feasibility, progress, and comfort. CLOVER differs by explicitly using evaluator-provided planning feedback to train a trajectory-level value estimator and to refine the candidate proposal distribution, rather than relying only on single-trajectory imitation.

A.2 Multimodal Trajectory Generation and Proposal-selection Planning

Driving behavior is inherently multi-modal: in the same scene, multiple future trajectories, such as maintaining speed, yielding, braking, or choosing a slightly different interaction strategy, may all be valid. Proposal-selection planners address this by generating a set of candidate trajectories and selecting one for execution. One line of work uses discretized trajectory vocabularies or anchor sets, such as Hydra-MDP (Li et al., 2024a) and VADv2 (Chen et al., 2024), which provide a structured candidate space but are constrained by the coverage of the predefined vocabulary. Another line uses generative models, such as DiffusionDrive (Liao et al., 2025) and GoalFlow (Xing et al., 2025), to produce diverse trajectories with greater flexibility, often at higher computational cost. Proposal-centric planners such as iPad (Guo et al., 2025) refine candidate trajectories online and offer an efficient alternative. Despite these advances, proposal generation is often still supervised mainly by the single logged trajectory, which can limit candidate-set coverage when multiple high-quality behaviors are possible. CLOVER addresses this limitation by constructing evaluator-filtered pseudo-expert trajectories and training the generator with set-level coverage supervision, thereby improving both proposal diversity and candidate-set quality.

A.3 Learned Trajectory Scoring

A key component of proposal-selection planning is trajectory scoring: given a candidate set, the planner must identify the safest and most effective trajectory. Learned cost functions, ranking modules, and trajectory scorers have been used to improve selection quality, especially when paired with rich proposal sets (Yao et al., 2025; Kirby et al., 2026; Li et al., 2025d). However, trajectory scoring in autonomous driving is challenging because the true planning score depends on map constraints, future obstacle states, interaction rules, and non-differentiable metric computations. A learned scorer must estimate this quality from the current observation and candidate trajectory, and directly maximizing such an imperfect score can exploit scorer errors or reduce diversity. CLOVER treats the scorer as a closed-loop value estimator trained on evaluator-provided sub-scores, and uses it conservatively: instead of unrestricted score maximization, the generator is refined through teacher-selected top-k and vector-Pareto proposal targets with stability regularization.

A.4 Relation to Reinforcement Learning

CLOVER shares a superficial similarity with actor–critic reinforcement learning: a generator proposes trajectories and a scorer evaluates them. However, CLOVER is not a reinforcement learning method. Classic reinforcement learning relies on environment interaction, trial-and-error exploration, and reward optimization over visited states. In contrast, CLOVER operates offline on logged driving scenes; all feedback is computed from generated candidate trajectories using a rule-based planning evaluator, and no online environment interaction or exploration is performed. Recent works such as Evadrive (Jiao et al., 2025) explore RL-style planning for autonomous driving, whereas CLOVER focuses on closed-loop value estimation and conservative self-distillation for proposal-selection

planning. Thus, CLOVER is better viewed as an offline evaluator-guided ranking and proposal-refinement framework rather than an RL-based planner.

B Pseudo-Expert Trajectory Generation

CLOVER uses pseudo-expert trajectories to provide set-level multi-modal supervision beyond the single logged trajectory. The pseudo-expert pipeline is used only during training. It constructs a diverse trajectory pool using privileged metric-cache information, scores and filters the candidates with the NAVSIM evaluator, and converts the resulting pool into compact per-scene supervision targets during Stage-1 training. The overall data flow is:

$$\begin{aligned}\text{metric cache} &\rightarrow \text{candidate family generation}\\ &\rightarrow \text{drivable-area / occupancy pre-check}\\ &\rightarrow \text{PDM scoring}\\ &\rightarrow \text{coverage-aware selection}\\ &\rightarrow \text{post-processing}\\ &\rightarrow \text{training-time pseudo-expert sampling}\end{aligned}$$

Training-time privileged information. The pseudo-expert generator uses information that is available in the training metric cache but not available to the deployed planner. This includes the current ego state, route centerline, drivable-area map, future obstacle occupancy, and the NAVSIM PDM evaluator. Therefore, pseudo-experts should be understood as privileged training supervision rather than an online planner. At inference time, CLOVER only uses the learned generator and scorer.

Trajectory parameterization. Each candidate trajectory is generated in a centerline-aligned coordinate system. Let d_0 be the current ego progress along the route centerline and let lat_0 be the current lateral offset. A candidate is defined by a longitudinal progress curve d_t and a lateral offset curve lat_t, and then mapped back to Cartesian coordinates:

$$d_t = d_0 + \sum_{k \le t} v_k \Delta t, \tag{4}$$

$$x_t = x_{\mathrm{cl}}(d_t) - \mathrm{lat}_t \sin h_{\mathrm{cl}}(d_t), \quad y_t = y_{\mathrm{cl}}(d_t) + \mathrm{lat}_t \cos h_{\mathrm{cl}}(d_t), \tag{5}$$

$$\theta_t = h_{\mathrm{cl}}(d_t), \tag{6}$$

where $(x_{\mathrm{cl}}, y_{\mathrm{cl}}, h_{\mathrm{cl}})$ denotes the route centerline pose. For lateral transitions, we use a quintic smooth step:

$$s(r) = 6r^5 - 15r^4 + 10r^3, \tag{7}$$

which has zero first- and second-order derivatives at both endpoints. This produces smoother lateral shifts than linear interpolation and enables different comfort profiles even when the initial and final lateral offsets are similar.

Candidate action families. For each scene, we generate a structured pool of candidates from six interpretable action families, summarized in Table 10. These families are designed to cover both longitudinal and lateral variations that affect PDM sub-scores, including progress, time-to-collision, comfort, and drivable-area compliance. The goal is not to keep only high-score candidates, but to expose the model to a broad range of feasible, near-feasible, and informative low-score behaviors.

Table 10: **Pseudo-expert candidate families.** Each scene first generates a structured pool of candidate trajectories from interpretable longitudinal and lateral action families.

Family	Count	Purpose
Constant-speed lateral transitions	200	Cover smooth and aggressive lateral shifts under different speeds
Off-road boundary cases	12	Populate low-score drivable-area violation regions
Acceleration profiles	18	Vary longitudinal progress and speed profiles
Stop-go behaviors	9	Cover safe but low-progress behaviors
Approach-brake behaviors	10	Create TTC / comfort trade-offs
Overshoot lateral motions	12	Add aggressive lateral dynamics and comfort variation
Total	261	Structured candidate pool per scene

Pre-check before PDM scoring. Running the full PDM evaluator on every raw candidate is expensive. We therefore first perform lightweight geometric pre-checks. For drivable-area compliance, we compute ego-vehicle corners at each future pose and query whether they lie within valid drivable polygons. For collision pre-checking, we compare the future ego footprint with future obstacle occupancy from the metric cache. Each trajectory is assigned one of three coarse labels: feasible, near-feasible, or infeasible. We prioritize feasible and near-feasible candidates for full PDM scoring, while retaining a small subset of infeasible candidates to preserve score coverage. In our implementation, at most 180 candidates per scene are sent to the full evaluator.

PDM scoring and sub-score recording. Each selected candidate is evaluated by the NAVSIM PDM evaluator. Besides the final scalar PDM score, we also store its sub-scores, including no-at-fault collisions, drivable-area compliance, ego progress, time-to-collision, comfort, and driving-direction compliance. These sub-scores are used both for scorer supervision and for coverage-aware pseudo-expert selection.

Coverage-aware selection. A naive top-score selection would often keep many similar high-score trajectories, causing the pseudo-expert set to collapse back to a narrow mode. Instead, we select a compact per-scene set by covering different PDM sub-score patterns and score ranges. For each scored candidate, we define a coverage key:

$$\text{key} = (\text{DAC}, \text{Collision}, \text{TTC}, \text{Comfort}, \text{ProgressBin}) , \tag{8}$$

where binary or discrete PDM components are rounded to their categorical values, and ego progress is discretized into progress bins. We then greedily select candidates by minimizing:

$$\text{cost}_i = 10 \cdot \text{cov_count}(\text{key}_i) + \text{bin_count}(\text{score_bin}_i), \tag{9}$$

where cov_count counts how many already selected candidates share the same sub-score key, and bin_count counts how many candidates have been selected from the same total-score bin. This encourages coverage over both sub-score combinations and scalar score ranges. We keep up to 50 trajectories per scene in the pseudo-expert dataset before post-processing.

Post-processing and boundary interpolation. The generated pseudo-expert dataset is further cleaned before training. We remove candidates with wrong driving direction, i.e., trajectories whose driving-direction compliance is less than 1.0. We then identify score-boundary regions where small changes in lateral offset cause a large PDM drop. Specifically, within each family and speed group, adjacent lateral targets are compared; if the PDM score drop exceeds a threshold, we interpolate an additional candidate between the high-score and low-score sides and evaluate it again. These boundary-interpolation samples provide fine-grained supervision near metric discontinuities, where small geometric changes can lead to large score differences.

Training-time pseudo-expert sampling. The stored pseudo-expert pool is not used directly in full during training. For each scene, we first keep trajectories with PDM score above a threshold, then sort them by score and apply greedy farthest-point sampling in trajectory space. This produces at most $M = 8$ pseudo-experts per scene. If the logged human trajectory is not sufficiently covered by the selected pseudo-expert set, it is added to preserve the human driving prior. The selected trajectories are returned to the model as:

$$\texttt{pseudo_experts} \in \mathbb{R}^{M \times T \times 3}, \quad \texttt{pseudo_expert_mask} \in \{0, 1\}^M.$$

If no valid pseudo-expert is available for a scene, the system falls back to the logged human trajectory. This ensures that the training schema remains consistent while retaining standard imitation supervision as a safe fallback.

Key hyperparameters. Table 11 summarizes the main hyperparameters used in pseudo-expert generation and training-time sampling.

Qualitative examples. Figure 4 visualizes pseudo-expert trajectory pools on representative scenes. The candidates cover a wide range of lateral offsets, progress profiles, and safety/comfort trade-offs. The visualization also illustrates why pseudo-expert supervision is more informative than local perturbation around the logged trajectory: the generated pool contains both high-score feasible alternatives and lower-score boundary cases that expose the model to metric discontinuities.

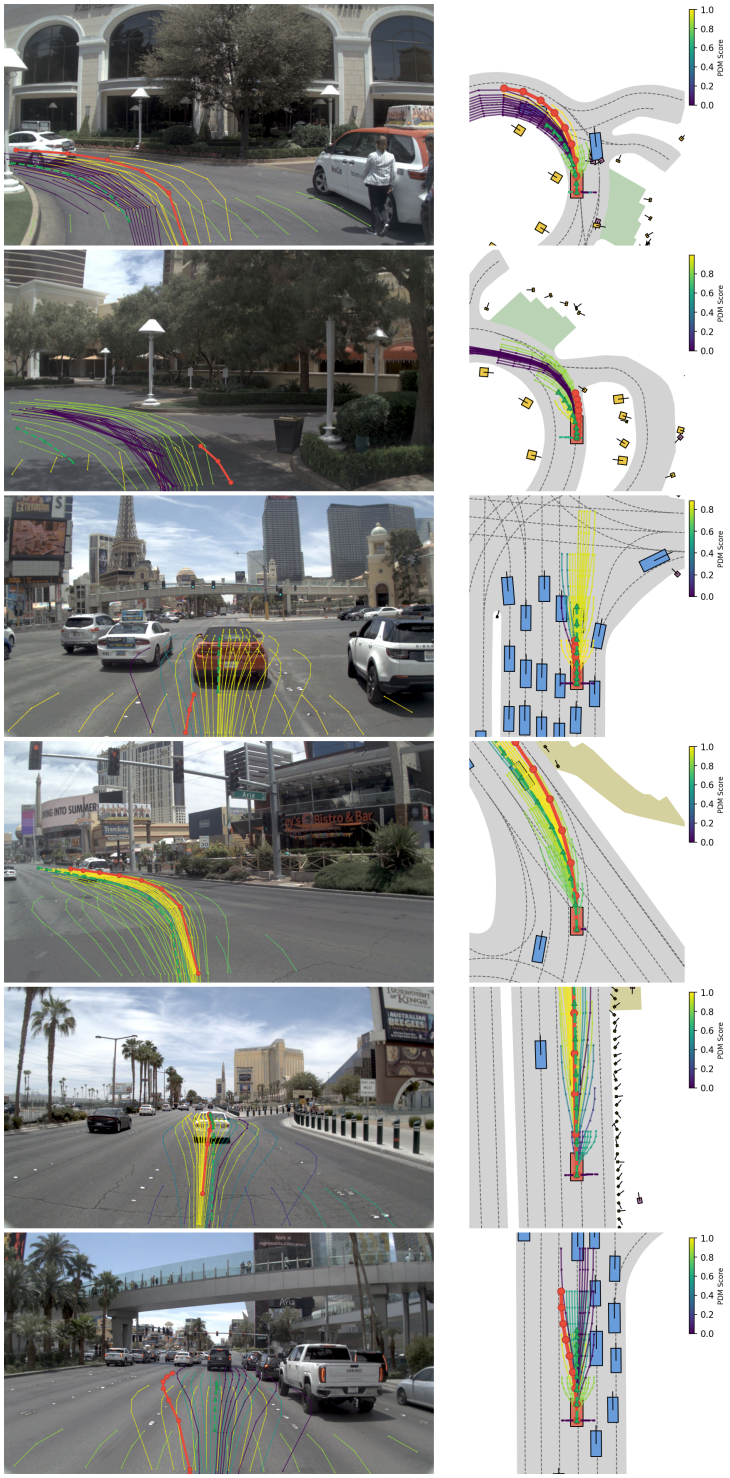

Figure 4: **Visualization of pseudo-expert trajectory candidates.** We show representative pseudo-expert trajectory pools in camera view and bird's-eye view. Candidate trajectories are colored by their true PDM scores. The pseudo-expert generator creates diverse trajectories covering different lateral offsets, progress regimes, and boundary cases. High-score candidates provide feasible multi-modal supervision, while lower-score and boundary candidates help expose the generator and scorer to drivable-area, collision, progress, and comfort variations. These pseudo-experts are used only during Stage-1 training and are not required at inference time.

Table 11: **Key hyperparameters for pseudo-expert generation and sampling.** The candidate pool is generated from structured action families, scored by the evaluator, filtered and selected for coverage, and finally sampled into a compact training-time pseudo-expert set.

Hyperparameter	Value
Number of future poses	8
Sampling interval	0.5 s
Speed candidates	$(0, 2, 4, 6, 8, 10, 12, 15)$ m/s
Regular lateral offsets	$(-3.5, -2.0, -1.0, -0.5, 0, 0.5, 1.0, 2.0, 3.5)$ m
Off-road lateral offsets	$(-7.0, -5.5, 5.5, 7.0)$ m
Transition portions	$(0.35, 0.6, 1.0)$
Acceleration rates	$(-2.0, -1.0, -0.5, 0.5, 1.0, 2.0)$ m/s^2
Raw candidates per scene	261
Maximum PDM-scored candidates per scene	180
Retained pseudo-expert pool per scene	50
Score-drop threshold for boundary interpolation	0.25
Maximum score boundaries per scene	3
Boundary interpolation samples per boundary	1
Training-time score threshold	0.8
Training-time pseudo-expert top-K	8

Table 12: EPDMS composition in NAVSIM v2 (new metrics relative to v1 are highlighted).

Metric	Weight	Range
No at-fault Collisions (NC)	multiplier	$\{0, \frac{1}{2}, 1\}$
Drivable Area Compliance (DAC)	multiplier	$\{0, 1\}$
Driving Direction Compliance (DDC)	multiplier	$\{0, \frac{1}{2}, 1\}$
Traffic Light Compliance (TLC)	multiplier	$\{0, 1\}$
Ego Progress (EP)	5	$[0, 1]$
Time to Collision (TTC) within bound	5	$\{0, 1\}$
Lane Keeping (LK)	2	$\{0, 1\}$
History Comfort (HC)	2	$\{0, 1\}$
Extended Comfort (EC)	2	$\{0, 1\}$

C TRAJECTORY-QUALITY SCORE IN NAVSIM (PDMS V1 AND EPDMS V2)

We use the planning-oriented NAVSIM benchmark and adopt the official Predictive Driver Model Score (PDMS) from NAVSIM v1 as our primary offline trajectory-quality score $s(\cdot, \cdot)$; higher is better. PDMS is a pseudo closed-loop metric that holistically assesses safety, comfort, and progress via multiplicative penalties and a weighted average:

$$\text{PDMS} \;=\; \text{NC} \times \text{DAC} \times \left(\frac{5 \cdot \text{EP} + 5 \cdot \text{TTC} + 2 \cdot \text{C}}{12} \right), \tag{10}$$

where NC denotes no at-fault collisions, DAC drivable-area compliance, EP ego progress, TTC time-to-collision within bound, and C comfort.

NAVSIM v2: Extended PDMS (EPDMS). NAVSIM v2 extends PDMS to improve coverage and fairness of open-loop planning evaluation. Compared to NAVSIM v1, EPDMS introduces additional weighted subscores (lane keeping and extended comfort variants), additional multiplier penalties (driving direction compliance and traffic light compliance), and a false-positive penalty filtering scheme.

False-positive penalty filtering. To reduce false-positive penalties, NAVSIM v2 disables a penalty when the human agent is also responsible for the corresponding violation. Formally, for a metric m, define

$$\text{filter}_m(\text{agent}, \text{human}) = \begin{cases} 1.0, & \text{if } m(\text{human}) = 0, \\ m(\text{agent}), & \text{otherwise.} \end{cases} \tag{11}$$

Intuitively, if the human baseline also triggers the violation, the metric is neutralized (set to 1.0) rather than penalizing the planner.

EPDMS definition. With the above filtering, EPDMS is defined as

$$\text{EPDMS} = \left(\prod_{m \in \{\text{NC,DAC,DDC,TLC}\}} \text{filter}_m(\text{agent}, \text{human}) \right) \tag{12}$$

$$\cdot \left(\frac{\sum_{m \in \{\text{TTC,EP,HC,LK,EC}\}} w_m \cdot \text{filter}_m(\text{agent}, \text{human})}{\sum_{m \in \{\text{TTC,EP,HC,LK,EC}\}} w_m} \right), \tag{13}$$

where w_m are the weights listed in Table 12.

Pseudo closed-loop aggregation in NAVSIM v2. NAVSIM v1 computes metrics after a 4-second non-reactive simulation rollout (background actors follow recorded futures; ego follows the planned trajectory via a controller). NAVSIM v2 uses a two-stage aggregation to better approximate closed-loop behavior while remaining open-loop: (i) a first-stage score is computed on an initial 4-second scene; (ii) multiple follow-up scenes (precomputed rollouts starting from the same initial scene but with different end states) are also scored, and then aggregated with weights given by a Gaussian kernel based on how close each follow-up scene's start state is to the submitted planner's first-stage end state. Finally, the first-stage score and the aggregated second-stage score are multiplied to obtain the final aggregated EPDMS score.

D Anchor-assisted Soft Reranking

This appendix describes the optional anchor-assisted soft reranking used in our NAVSIM v2 EPDMS submission. The motivation is that diverse proposals can improve oracle coverage while independent per-frame ranking can switch between near-tied high-quality modes. Such switching can reduce extended comfort (EC), which assesses consistency across adjacent planning decisions. The anchor provides a stable reference computed independently for each scene, without future information, evaluator scores, or previously selected trajectories at inference time. It is a practical mitigation for temporal inconsistency, not a complete solution to temporally coherent multimodal planning.

Anchor model. The anchor trajectory is produced by a single-mode planner with the same observation inputs and the same base architecture as CLOVER. We use the *Closed-loop only* ablation checkpoint from Table 7: Stage 1 is omitted and only Stage-2 refinement is applied. This checkpoint achieves 93.8 PDMS on NAVSIM v1 and is distinct from the official DrivoR weights. It provides a stable reference trajectory, but is not used as an evaluator and does not filter candidates. Its standalone EPDMS is 88.7 (Table 6).

Soft reranking formula. For each candidate trajectory τ_k, let

$$\mathbf{p}_k(t) = (x_k(t), y_k(t), \psi_k(t)), \quad t = 1, \ldots, T,$$

and let the anchor trajectory be

$$\mathbf{a}(t) = (x_a(t), y_a(t), \psi_a(t)).$$

We define the full-trajectory position RMS:

$$\text{XYRMS}_k = \sqrt{\frac{1}{T} \sum_{t=1}^{T} [(x_k(t) - x_a(t))^2 + (y_k(t) - y_a(t))^2]}, \tag{14}$$

and the full-trajectory heading RMS:

$$\text{HeadRMS}_k = \sqrt{\frac{1}{T} \sum_{t=1}^{T} \text{wrap}\left(\psi_k(t) - \psi_a(t)\right)^2}. \tag{15}$$

Table 13: **Coarse sweep for anchor-assisted soft reranking.** We report representative grid-search results over scorer weight λ_s, position-anchor weight λ_{xy}, and heading-anchor weight λ_ψ. Scores are normalized EPDMS values. The no-anchor baseline obtains 0.89322 EPDMS, while the displayed anchor configurations improve the aggregate score at the cost of EP. The main result uses $\lambda_s = 2.00$, $\lambda_{xy} = 0.20$, and $\lambda_\psi = 0.50$.

λ_s	λ_{xy}	λ_ψ	Score	EP	EC	HC
–	–	–	0.89322	**0.92408**	0.3500	0.96459
2.00	0.20	0.50	**0.90356**	0.8689	0.7550	0.9823
1.75	0.20	0.75	0.90213	0.8613	0.7641	0.9825
2.00	0.20	0.75	0.90292	0.8652	0.7594	0.9825
2.25	0.20	0.75	0.90222	0.8689	0.7509	0.9823
1.75	0.20	1.00	0.90212	0.8596	0.7667	0.9825
2.00	0.20	1.00	0.90212	0.8630	0.7605	0.9825
2.25	0.20	1.00	0.90168	0.8663	0.7539	0.9824
2.00	0.25	0.50	0.90285	0.8633	0.7610	0.9825
2.00	0.25	0.75	0.90194	0.8601	0.7640	0.9825
1.75	0.35	0.75	0.90086	0.8502	0.7698	0.9828
2.00	0.35	0.75	0.90177	0.8534	0.7679	0.9826
2.25	0.35	0.75	0.90216	0.8564	0.7660	0.9825
1.75	0.35	1.00	0.90021	0.8492	**0.7718**	**0.9828**
2.00	0.35	1.00	0.90118	0.8521	0.7699	0.9826
2.25	0.35	1.00	0.90180	0.8549	0.7679	0.9825

The anchor penalty is:

$$\text{Penalty}_k = \lambda_{xy} \frac{\text{XYRMS}_k}{S_{\text{pos}}} + \lambda_\psi \frac{\text{HeadRMS}_k}{S_\psi}, \tag{16}$$

where $S_{\text{pos}} = 5.0$ and $S_\psi = 0.35$. Given the learned scorer score s_k, the final soft reranking score is:

$$Q_k = \lambda_s s_k - \text{Penalty}_k. \tag{17}$$

The selected candidate is:

$$k^* = \arg\max_k Q_k. \tag{18}$$

This is a soft scheme: no hard threshold is used to discard candidates.

Parameter search. We sweep the scorer weight λ_s, position-anchor weight λ_{xy}, and heading-anchor weight λ_ψ on the validation split. The main reported result uses:

$$\lambda_s = 2.00, \quad \lambda_{xy} = 0.20, \quad \lambda_\psi = 0.50.$$

This configuration achieves the best score in our search, with a final score of 0.90356.

The search shows a consistent trade-off between final score and extended comfort. A light position anchor, around $\lambda_{xy} = 0.20$, retains more progress than stronger anchoring and achieves the best total score in the displayed search. Relative to no anchoring, the selected configuration increases EC from 0.3500 to 0.7550 but decreases EP from 0.92408 to 0.8689; aggregate EPDMS rises from 0.89322 to 0.90356. Thus, the gain reflects a trade-off between consistency and progress rather than a uniform improvement in every sub-score.

D.1 Anchor Choice, Baseline Controls, and Temporal Consistency

Distinguishing the anchor from DrivoR. The 88.7 EPDMS anchor result refers to the Stage-2-only checkpoint described above, whose NAVSIM-v1 PDMS is 93.8, not to the released DrivoR weights. Evaluating the official DrivoR checkpoint on NAVSIM-v2 `navtest` yields 87.4 EPDMS in our evaluation, close to the 87.2 reported for DrivoR in PriorEye (Yeon et al., 2026). The DrivoR release does not provide an official result for this `navtest` EPDMS setting; we therefore report our evaluation here separately from the published results in Table 3. Even without anchoring, CLOVER reaches 89.3 EPDMS, improving over our DrivoR evaluation by 1.9 points. With anchoring, it reaches 90.4. These comparisons separate the improvement of the underlying planner from the additional benefit of the consistency reference.

Alternative anchor models. The reranking rule does not require a particular anchor architecture; we use our own checkpoint to avoid introducing a different model architecture. We additionally tested trajectories from DiffusionDrive (Liao et al., 2025) and RecogDrive (Li et al., 2025c) as anchors. With modest adjustments to the reranking weights, these alternatives yielded EPDMS scores in the range 90.2–90.5, comparable to our default anchor's 90.4. These tests suggest that the benefit is not specific to our checkpoint: the anchor serves as a stability reference to mitigate the loss of extended comfort caused by inconsistent decisions across adjacent frames.

Applying the same reranking to SparseDriveV2. SparseDriveV2 (Sun et al., 2026) already has higher EC than anchored CLOVER in Table 3 (78.4 vs. 75.5). We applied the same anchor-assisted reranking scheme to SparseDriveV2 and swept the weight combinations in Table 13. The resulting EPDMS ranged from 88.9 to 90.1; even the best configuration did not improve on its 90.1 baseline at the reported precision. In this control, anchoring did not improve EC and reduced EP, limiting the aggregate score. This indicates that the benefit depends on the planner's existing consistency–progress balance; anchoring is not a general score boost for every planner.

Diversity versus consistency of the selected trajectory. The NAVSIM-v1 analysis measures proposal coverage and quality under PDMS. NAVSIM v2 additionally evaluates cross-frame consistency through EC. When several behaviorally distinct candidates receive similarly high or saturated scores, a per-frame scorer has little incentive to prefer a temporally consistent mode. Small ranking changes can then switch the selected trajectory between individually valid alternatives and sharply reduce EC. The issue is temporal switching in the selected sequence, rather than diversity of the candidate set itself. Our anchor mitigates this mismatch by restricting selection toward a consistent reference, but sacrifices some progress and flexibility, as quantified in Table 13.

A stateful alternative. Drive-JEPA (Wang et al., 2026) uses momentum-aware selection that compares current candidates with the trajectory selected at the preceding time step. In an additional CLOVER experiment, this type of selection reaches 90.6 EPDMS without the single-mode anchor, compared with 90.4 for our reported anchor-assisted configuration. Using previous decisions is reasonable in online driving, but creates a temporal dependency in final selection: scene predictions can no longer be selected entirely independently in parallel. This changes the inference execution protocol, rather than the definition of EPDMS. We retain the anchor-assisted configuration for the main results to preserve independent scene processing and report the stateful result separately, since it uses a different information dependency at inference.

E Theoretical Analysis

This appendix provides full proofs for the theoretical results in Section 4. All scores are normalized to $[0, 1]$. For simplicity, we write expectations over $o \sim \mathcal{D}$ implicitly when no ambiguity arises.

E.1 Notation

Let $o \sim \mathcal{D}$ be a logged driving scene and let $\tau \in \mathcal{X}_o$ be a candidate ego trajectory. The true non-differentiable trajectory-quality score is

$$R^*(o, \tau) \in [0, 1]. \tag{19}$$

The learned scalar scorer at iteration t is

$$s_t(o, \tau) = \Gamma_{\mathrm{PDMS}}(S_t(o, \tau)) \in [0, 1], \tag{20}$$

where $S_t(o, \tau)$ predicts PDMS-related sub-scores. The generator induces a trajectory distribution $\pi_t(\tau \mid o)$. The true expected score and scorer-surrogate expected score are

$$J^*(\pi) = \mathbb{E}_{o\sim\mathcal{D},\, \tau\sim\pi(\cdot|o)}[R^*(o, \tau)], \tag{21}$$

$$J_{s_t}(\pi) = \mathbb{E}_{o\sim\mathcal{D},\, \tau\sim\pi(\cdot|o)}[s_t(o, \tau)]. \tag{22}$$

For a K-proposal set, we also use the empirical distribution

$$\mu_t^o = \frac{1}{K} \sum_{i=1}^{K} \delta_{\tau_i^t(o)}. \tag{23}$$

E.2 Auxiliary Result: Approximate Monotonicity

We first restate the assumptions. Let the trust region around the current generator be

$$\mathcal{B}_t(\rho) = \{\pi : d(\pi, \pi_t) \le \rho\}. \tag{24}$$

Assume local scorer accuracy:

$$|J_{s_t}(\pi) - J^*(\pi)| \le \epsilon_t(\rho), \qquad \forall \pi \in \mathcal{B}_t(\rho). \tag{25}$$

Define

$$e_t(\pi) = |J_{s_t}(\pi) - J^*(\pi)|, \tag{26}$$

and the surrogate improvement

$$m_t = J_{s_t}(\pi_{t+1}) - J_{s_t}(\pi_t). \tag{27}$$

By the definition of $e_t(\pi)$,

$$J^*(\pi_{t+1}) \ge J_{s_t}(\pi_{t+1}) - e_t(\pi_{t+1}), \tag{28}$$

and

$$J^*(\pi_t) \le J_{s_t}(\pi_t) + e_t(\pi_t). \tag{29}$$

Subtracting the second inequality from the first gives

$$J^*(\pi_{t+1}) - J^*(\pi_t) \ge J_{s_t}(\pi_{t+1}) - J_{s_t}(\pi_t) - e_t(\pi_{t+1}) - e_t(\pi_t) \tag{30}$$

$$= m_t - e_t(\pi_{t+1}) - e_t(\pi_t). \tag{31}$$

This proves the general form. Therefore, if

$$m_t \ge e_t(\pi_{t+1}) + e_t(\pi_t), \tag{32}$$

then

$$J^*(\pi_{t+1}) \ge J^*(\pi_t). \tag{33}$$

Now assume the generator update is conservative and remains in $\mathcal{B}_t(\rho)$. If the generator phase improves the regularized surrogate up to residual $\alpha_t \ge 0$:

$$J_{s_t}(\pi_{t+1}) - \lambda D(\pi_{t+1}, \pi_t) \ge J_{s_t}(\pi_t) - \alpha_t, \tag{34}$$

with $D(\pi_{t+1}, \pi_t) \ge 0$, then

$$J_{s_t}(\pi_{t+1}) \ge J_{s_t}(\pi_t) - \alpha_t, \tag{35}$$

so $m_t \ge -\alpha_t$. Since $\pi_t, \pi_{t+1} \in \mathcal{B}_t(\rho)$,

$$e_t(\pi_t) \le \epsilon_t(\rho), \qquad e_t(\pi_{t+1}) \le \epsilon_t(\rho). \tag{36}$$

Substituting into the general bound yields

$$J^*(\pi_{t+1}) \ge J^*(\pi_t) - 2\epsilon_t(\rho) - \alpha_t. \tag{37}$$

This completes the proof.

E.3 Auxiliary Result: One-step Scorer Extrapolation

Define the scorer error function

$$g_t(o, \tau) = |s_t(o, \tau) - R^*(o, \tau)|. \tag{38}$$

Since both s_t and R^* lie in $[0, 1]$, we have $0 \le g_t \le 1$. Assume that the scorer phase fits the current generator distribution:

$$\mathbb{E}_{\tau \sim \pi_t}[g_t(o, \tau)] \le \epsilon_t. \tag{39}$$

Let the one-step drift be

$$\Delta_t = \mathrm{TV}(\pi_{t+1}, \pi_t). \tag{40}$$

For any function $f \in [0, 1]$,

$$|\mathbb{E}_P[f] - \mathbb{E}_Q[f]| \le \mathrm{TV}(P, Q). \tag{41}$$

Thus,

$$\mathbb{E}_{\tau\sim\pi_{t+1}}[g_t(o,\tau)] \le \mathbb{E}_{\tau\sim\pi_t}[g_t(o,\tau)] + \mathrm{TV}(\pi_{t+1},\pi_t) \tag{42}$$

$$\le \epsilon_t + \Delta_t. \tag{43}$$

Summing over $t = 0, \dots, T-1$ gives

$$\sum_{t=0}^{T-1} \mathbb{E}_{\pi_{t+1}}[|s_t - R^*|] \le \sum_{t=0}^{T-1} \epsilon_t + \sum_{t=0}^{T-1} \Delta_t. \tag{44}$$

If $\epsilon_t \le \epsilon$ and $\Delta_t \le \rho$, the bound becomes

$$\sum_{t=0}^{T-1} \mathbb{E}_{\pi_{t+1}}[|s_t - R^*|] \le T\epsilon + T\rho. \tag{45}$$

We now compare this with using a fixed initial scorer s_0. If

$$\mathbb{E}_{\pi_0}[|s_0 - R^*|] \le \epsilon_0, \tag{46}$$

then

$$\mathbb{E}_{\pi_{t+1}}[|s_0 - R^*|] \le \epsilon_0 + \mathrm{TV}(\pi_{t+1},\pi_0). \tag{47}$$

By the triangle inequality,

$$\mathrm{TV}(\pi_{t+1},\pi_0) \le \sum_{k=0}^{t} \Delta_k. \tag{48}$$

Therefore,

$$\mathbb{E}_{\pi_{t+1}}[|s_0 - R^*|] \le \epsilon_0 + \sum_{k=0}^{t} \Delta_k. \tag{49}$$

Summing over t yields

$$\sum_{t=0}^{T-1} \mathbb{E}_{\pi_{t+1}}[|s_0 - R^*|] \le T\epsilon_0 + \sum_{t=0}^{T-1}\sum_{k=0}^{t} \Delta_k \tag{50}$$

$$= T\epsilon_0 + \sum_{k=0}^{T-1} (T-k)\Delta_k. \tag{51}$$

If $\Delta_k \le \rho$, this is

$$T\epsilon_0 + \frac{T(T+1)}{2}\rho. \tag{52}$$

Thus, refitting the scorer reduces the drift dependence from quadratic to linear in the number of refinement steps.

E.4 Auxiliary Result: Approximate Pareto Consistency

Let the true vector score be

$$\mathbf{R}^*(o,\tau) = (R_1^*(o,\tau), \dots, R_M^*(o,\tau)), \tag{53}$$

and the predicted vector score be

$$\mathbf{s}_t(o,\tau) = (s_{t,1}(o,\tau), \dots, s_{t,M}(o,\tau)). \tag{54}$$

Assume uniform vector error on the candidate pool:

$$\|\mathbf{s}_t(o,\tau) - \mathbf{R}^*(o,\tau)\|_\infty \le \delta_t, \qquad \forall \tau \in \mathcal{C}_t(o). \tag{55}$$

We define the true ϵ-Pareto set as

$$\mathcal{P}_\epsilon^*(o) = \{\tau \in \mathcal{C}_t(o) : \nexists \tau' \in \mathcal{C}_t(o) \text{ such that } R_m^*(o,\tau') \ge R_m^*(o,\tau) + \epsilon,\ \forall m\}. \tag{56}$$

The scorer-induced Pareto set is

$$\mathcal{P}_{s_t}(o) = \big\{\tau \in \mathcal{C}_t(o) : \nexists \tau' \in \mathcal{C}_t(o) \text{ such that } s_{t,m}(o,\tau') \geq s_{t,m}(o,\tau),\ \forall m, \text{ and } \exists m :\ s_{t,m}(o,\tau') > s_{t,m}(o,\tau)\big\}. \tag{57}$$

We prove that for any $\kappa > 0$,

$$\mathcal{P}_{s_t}(o) \subseteq \mathcal{P}^*_{2\delta_t+\kappa}(o). \tag{58}$$

Suppose, for contradiction, that there exists $\tau \in \mathcal{P}_{s_t}(o)$ but $\tau \notin \mathcal{P}^*_{2\delta_t+\kappa}(o)$. Then there exists $\tau' \in \mathcal{C}_t(o)$ such that

$$R^*_m(o,\tau') \geq R^*_m(o,\tau) + 2\delta_t + \kappa, \qquad \forall m. \tag{59}$$

By the error bound,

$$s_{t,m}(o,\tau') \geq R^*_m(o,\tau') - \delta_t, \tag{60}$$

and

$$s_{t,m}(o,\tau) \leq R^*_m(o,\tau) + \delta_t. \tag{61}$$

Combining these inequalities gives

$$\begin{aligned} s_{t,m}(o,\tau') &\geq R^*_m(o,\tau) + 2\delta_t + \kappa - \delta_t && (62)\\ &= R^*_m(o,\tau) + \delta_t + \kappa && (63)\\ &\geq s_{t,m}(o,\tau) + \kappa. && (64) \end{aligned}$$

Thus,

$$s_{t,m}(o,\tau') > s_{t,m}(o,\tau), \qquad \forall m. \tag{65}$$

Therefore τ' strictly dominates τ in scorer space, contradicting $\tau \in \mathcal{P}_{s_t}(o)$. This proves the result.

The additional $\kappa > 0$ only excludes the degenerate boundary case where the true margin is exactly $2\delta_t$ and the scorer errors simultaneously attain their worst-case values in every objective. Since κ can be arbitrarily small, the result can be interpreted as containment in the right-limit $(2\delta_t)^+$-approximate Pareto region.

Coverage-error extension. Assume that the student generator covers each predicted Pareto target within distance η_t: for any $\tau \in \mathcal{P}_{s_t}(o)$, there exists a student proposal $\tilde{\tau}$ such that

$$d_{\mathcal{X}}(\tilde{\tau}, \tau) \leq \eta_t. \tag{66}$$

Assume also that $\mathbf{R}^*$ is L-Lipschitz on the valid proposal manifold:

$$\|\mathbf{R}^*(o,\tau) - \mathbf{R}^*(o,\tau')\|_\infty \leq L d_{\mathcal{X}}(\tau,\tau'). \tag{67}$$

Then the covered student proposal lies in the true $(2\delta_t + \kappa + L\eta_t)$-approximate Pareto region. Indeed, if $\tilde{\tau}$ were dominated by some τ' with margin $2\delta_t + \kappa + L\eta_t$, then

$$R^*_m(o,\tau') \geq R^*_m(o,\tilde{\tau}) + 2\delta_t + \kappa + L\eta_t. \tag{68}$$

Since

$$R^*_m(o,\tilde{\tau}) \geq R^*_m(o,\tau) - L\eta_t, \tag{69}$$

we obtain

$$R^*_m(o,\tau') \geq R^*_m(o,\tau) + 2\delta_t + \kappa, \qquad \forall m, \tag{70}$$

which contradicts $\tau \in \mathcal{P}^*_{2\delta_t+\kappa}(o)$.

E.5 Proof of Theorem 1

For a scene o, define the true high-score region

$$\mathcal{H}_o = \{\tau : R^*(o,\tau) \geq r_{\text{high}}\}. \tag{71}$$

Let the current proposal distribution be

$$\mu^o_t = \frac{1}{K}\sum_{i=1}^{K} \delta_{\tau^t_i(o)}. \tag{72}$$

Let $A_t(o)$ denote the scorer-selected top-k or vector-Pareto target set, and let

$$\nu_t^o = \frac{1}{|A_t(o)|} \sum_{\tau \in A_t(o)} \delta_\tau \tag{73}$$

be the empirical distribution on this target set. Define

$$p_t(o) = \mu_t^o(\mathcal{H}_o), \qquad q_t(o) = \nu_t^o(\mathcal{H}_o). \tag{74}$$

The selected-set enrichment assumption states that

$$q_t(o) \geq p_t(o) + \xi_t(o), \qquad \xi_t(o) > 0. \tag{75}$$

The conservative refinement assumption states that the actual student distribution is close to the ideal mixture

$$\bar{\mu}_{t+1}^o = (1-\alpha_t)\mu_t^o + \alpha_t \nu_t^o, \qquad \alpha_t \in [0,1], \tag{76}$$

in total variation:

$$\mathrm{TV}\left(\mu_{t+1}^o, \bar{\mu}_{t+1}^o\right) \leq \eta_t(o). \tag{77}$$

Then

$$\begin{aligned}
\bar{\mu}_{t+1}^o(\mathcal{H}_o) &= (1-\alpha_t)\mu_t^o(\mathcal{H}_o) + \alpha_t \nu_t^o(\mathcal{H}_o) && (78)\\
&= (1-\alpha_t)p_t(o) + \alpha_t q_t(o) && (79)\\
&\geq (1-\alpha_t)p_t(o) + \alpha_t[p_t(o) + \xi_t(o)] && (80)\\
&= p_t(o) + \alpha_t \xi_t(o). && (81)
\end{aligned}$$

For any event A, total variation implies

$$\mu_{t+1}^o(A) \geq \bar{\mu}_{t+1}^o(A) - \eta_t(o). \tag{82}$$

Taking $A = \mathcal{H}_o$ yields

$$p_{t+1}(o) = \mu_{t+1}^o(\mathcal{H}_o) \geq p_t(o) + \alpha_t \xi_t(o) - \eta_t(o). \tag{83}$$

This proves the high-score support statement.

Expected-score version. Assume instead that the selected target distribution has higher true expected score:

$$\mathbb{E}_{\nu_t^o}[R^*] - \mathbb{E}_{\mu_t^o}[R^*] \geq \beta_t(o) > 0. \tag{84}$$

Then

$$\begin{aligned}
\mathbb{E}_{\bar{\mu}_{t+1}^o}[R^*] &= (1-\alpha_t)\mathbb{E}_{\mu_t^o}[R^*] + \alpha_t \mathbb{E}_{\nu_t^o}[R^*] && (85)\\
&= \mathbb{E}_{\mu_t^o}[R^*] + \alpha_t \left(\mathbb{E}_{\nu_t^o}[R^*] - \mathbb{E}_{\mu_t^o}[R^*]\right) && (86)\\
&\geq \mathbb{E}_{\mu_t^o}[R^*] + \alpha_t \beta_t(o). && (87)
\end{aligned}$$

Because $R^* \in [0,1]$ and $\mathrm{TV}(\mu_{t+1}^o, \bar{\mu}_{t+1}^o) \leq \eta_t(o)$,

$$\mathbb{E}_{\mu_{t+1}^o}[R^*] \geq \mathbb{E}_{\bar{\mu}_{t+1}^o}[R^*] - \eta_t(o). \tag{88}$$

Thus,

$$\mathbb{E}_{\mu_{t+1}^o}[R^*] \geq \mathbb{E}_{\mu_t^o}[R^*] + \alpha_t \beta_t(o) - \eta_t(o). \tag{89}$$

Multi-round extension. Summing over $t = 0, \ldots, T-1$ gives

$$\mathbb{E}_{\mu_T^o}[R^*] \geq \mathbb{E}_{\mu_0^o}[R^*] + \sum_{t=0}^{T-1} \alpha_t \beta_t(o) - \sum_{t=0}^{T-1} \eta_t(o). \tag{90}$$

Therefore, Stage-2 refinement improves the true expected proposal quality whenever

$$\sum_{t=0}^{T-1} \alpha_t \beta_t(o) > \sum_{t=0}^{T-1} \eta_t(o). \tag{91}$$

From high-score support to Oracle@K. If K proposals are interpreted as samples from μ_t^o, the probability that at least one proposal falls in the high-score region is

$$A_K(p_t) = 1 - (1 - p_t)^K. \tag{92}$$

If all trajectories in $\mathcal{H}_o$ have score at least r_{high} and all candidates have score at least $r_{\min}$, then the oracle score satisfies

$$O_t(o) \geq r_{\min} + (r_{\text{high}} - r_{\min}) A_K(p_t). \tag{93}$$

Since $A_K(p)$ is monotone in p, increasing $p_t(o)$ increases this lower bound on Oracle@K.

From Oracle@K to deployed top-1. Let

$$O_t(o) = \max_{\tau \in T_t(o)} R^*(o, \tau) \tag{94}$$

be the true oracle in the proposal set, and let

$$V_t(o) = R^* \left(o, \arg \max_{\tau \in T_t(o)} s_t(o, \tau) \right) \tag{95}$$

be the true score of the deployed scorer-selected top-1. Define ranking regret

$$G_t(o) = O_t(o) - V_t(o). \tag{96}$$

Then

$$V_T(o) - V_0(o) = [O_T(o) - O_0(o)] + [G_0(o) - G_T(o)]. \tag{97}$$

Thus, deployed top-1 improvement comes from two sources: improving the proposal oracle and reducing ranking regret.

E.6 Uniform Margin as a Sufficient Condition

The selected-set enrichment assumption in Theorem 1 is weaker than worst-case scorer calibration. A clean sufficient condition is the standard margin condition.

Assume there are high-score and low-score regions $\mathcal{H}_o$ and $\mathcal{L}_o$ such that

$$R^*(o, \tau_h) \geq r_{\text{high}}, \quad \forall \tau_h \in \mathcal{H}_o, \tag{98}$$

$$R^*(o, \tau_l) \leq r_{\text{low}}, \quad \forall \tau_l \in \mathcal{L}_o, \tag{99}$$

with margin

$$\gamma = r_{\text{high}} - r_{\text{low}} > 0. \tag{100}$$

If the scorer satisfies the uniform error bound

$$|s_t(o, \tau) - R^*(o, \tau)| \leq \epsilon, \qquad \forall \tau \in \mathcal{H}_o \cup \mathcal{L}_o, \tag{101}$$

and $\gamma > 2\epsilon$, then for any $\tau_h \in \mathcal{H}_o$ and $\tau_l \in \mathcal{L}_o$,

$$s_t(o, \tau_h) \geq R^*(o, \tau_h) - \epsilon \geq r_{\text{high}} - \epsilon, \tag{102}$$

$$s_t(o, \tau_l) \leq R^*(o, \tau_l) + \epsilon \leq r_{\text{low}} + \epsilon. \tag{103}$$

Therefore,

$$s_t(o, \tau_h) - s_t(o, \tau_l) \geq \gamma - 2\epsilon > 0. \tag{104}$$

Thus high-score proposals are ranked above low-score proposals.

This condition is sufficient but not necessary. In practice, the worst-case error $\epsilon_{\max}$ can be overly conservative. Stage-2 distillation only requires the weaker statistical condition that scorer-selected top-k or Pareto targets are enriched with high true evaluator scores.

Table 14: **Empirical validation of selected-set enrichment.** Worst-case scorer calibration is too conservative, while high-probability separability, high-score precision, and scorer Top-k coverage support the enrichment condition required by Stage-2 distillation.

Validation target	Metric / protocol	Result	Interpretation
Stage-1 support	Pooled full-score proposals	35.42%	Stage 1 provides non-zero high-score support.
Worst-case margin	$P(\gamma_o > 2\epsilon_{\max,o})$, elite/reject $0.05/0.05$	10.64%	Worst-case calibration is too strict.
High-prob. margin	$P(\gamma_o > 2\epsilon_{p75,o})$, elite/reject $0.01/0.01$	92.50%	Quantile-level separation is widely satisfied.
High-prob. margin	$P(\gamma_o > 2\epsilon_{p90,o})$, elite/reject $0.01/0.01$	82.37%	Most scenes satisfy high-probability separation.
High-prob. margin	$P(\gamma_o > 2\epsilon_{p95,o})$, elite/reject $0.01/0.01$	71.56%	Strict quantile-level separation still often holds.
Pairwise separability	high ≥ 0.95, low ≤ 0.50	94.72%	Scorer usually ranks true high-score proposals above low-score ones.
High-score precision	$s \geq 0.90$	mean $R^* = 0.9529$, $P(R^* \geq 0.90) = 90.16\%$	Scorer-high proposals are reliable true high-score candidates.
High-score precision	$s \geq 0.95$	mean $R^* = 0.9753$, $P(R^* \geq 0.90) = 96.69\%$, $P(R^* = 1) = 69.74\%$	High-confidence scorer targets are strongly enriched.
Top-k quality	scorer Top-1	mean best $R^* = 0.9448$, oracle gap 0.0491	Single top selection is already high quality.
Top-k quality	scorer Top-8	mean best $R^* = 0.9656$, oracle gap 0.0283	Top-k targets improve high-score coverage.
Top-k quality	scorer Top-16	mean best $R^* = 0.9730$, oracle gap 0.0209	Larger target sets further reduce oracle gap.

E.7 Empirical Support for Selected-Set Enrichment

We empirically validate the selected-set enrichment assumption used in Theorem 1. The analysis is conducted on the NAVSIM evaluation PKL with 12,146 scenes and 64 proposals per scene.

Table 14 summarizes the empirical proxies. The worst-case margin condition $\gamma > 2\epsilon_{\max}$ is intentionally conservative and is not expected to hold for all proposals. Indeed, under the per-scene elite/reject $(0.05/0.05)$ protocol, only 10.64% of scenes satisfy $\gamma_o > 2\epsilon_{\max,o}$. However, the weaker statistical conditions needed by Stage-2 distillation are strongly supported. Under elite/reject $(0.01/0.01)$, 92.50%, 82.37%, and 71.56% of scenes satisfy $\gamma_o > 2\epsilon_{p75,o}$, $\gamma_o > 2\epsilon_{p90,o}$, and $\gamma_o > 2\epsilon_{p95,o}$, respectively. Moreover, fixed-threshold high/low groups are highly separable: for high ≥ 0.95 and low ≤ 0.50, pairwise ranking accuracy reaches 94.72%. Most importantly, scorer-high proposals are truly high scoring: when $s \geq 0.95$, the true mean score is 0.9753, $P(R^* \geq 0.90) = 96.69\%$, and $P(R^* = 1) = 69.74\%$, compared with a pooled full-score rate of 35.42%. These observations support the key premise of Theorem 1: Stage 2 improves over Stage 1 not by discovering high-quality modes from scratch, but by reallocating proposal mass toward scorer-selected targets that are statistically enriched with high true evaluator scores.

At the pooled level, let $\mathcal{H} = \{\tau : R^*(\tau) = 1\}$. The proposal pool has

$$p \approx 35.42\%. \tag{105}$$

Using $s \geq 0.95$ as a high-confidence scorer-selected proxy gives

$$q \approx 69.74\%. \tag{106}$$

Thus the enrichment gap is approximately

$$q - p \approx 69.74\% - 35.42\% = 34.32\%. \tag{107}$$

This directly supports the condition $q_t(o) \geq p_t(o) + \xi_t(o)$ in an aggregate sense.

F Implementation Details and Hyperparameters

This appendix summarizes the implementation details and hyperparameters for the main NAVSIM experiments. The Bench2Drive adaptation is described separately in Appendix K. Unless otherwise specified, all models and ablations use the same DrivoR-style base architecture shown in Table 15. Stage-1 diversity pretraining and its loss weights are summarized in Table 16. The pseudo-expert generation configuration used for evaluator-filtered trajectory coverage is provided in Table 17. Stage-2 alternating self-distillation and its teacher-set configuration are summarized in Table 18. Finally, Table 19 reports the compute resources and wall-clock training time.

F.1 Unified Scorer Loss

Both Stage-1 scorer pretraining and Stage-2 scorer fitting use BCE-with-logits for every predicted metric component, including ego progress (EP). Let a_m be the raw logit for component m, $\hat{r}_m = \sigma(a_m)$ its predicted sub-score, and $r_m \in [0, 1]$ the normalized evaluator target. The component loss is

$$\ell_m(a_m, r_m) = -r_m \log \sigma(a_m) - (1 - r_m) \log(1 - \sigma(a_m)), \tag{108}$$

Table 15: **Base DrivoR-style architecture used by CLOVER.** Unless otherwise specified, all models and ablations use this common architecture.

Hyperparameter	Value
Input cameras	front, left, right, back
LiDAR input	disabled
Backbone	DINOv2 ViT-S
Backbone identifier	`timm/vit_small_patch14_reg4_dinov2.lvd142m`
LoRA	enabled, rank 32
Scene tokens per camera	16
Prediction horizon	4.0 s
Sampling interval	0.5 s
Number of future poses	8
Number of proposals	64
Transformer hidden dimension	256
Transformer FFN dimension	1024
Trajectory representation	(x, y, θ)
One token per trajectory	true
NAVSIM v1 deployment weights	NC/DAC/DDC/TTC/EP/Comfort = 1/1/0/5/5/2

Table 16: **Stage-1 diversity pretraining hyperparameters.** Stage 1 trains the generator with logged-trajectory imitation and evaluator-filtered pseudo-expert coverage, while pretraining the scorer with evaluator-provided trajectory scores.

Hyperparameter	Value
Training epochs	25
Batch size	32
Number of GPUs	4
DataLoader workers	4
Optimizer	AdamW
Base learning rate	2×10^{-4}
Precision	16-mixed
Seed	2
Ray scoring	enabled
Ray threads per node	96
Pseudo-expert top-K	8
Pseudo-expert score threshold	0.8
Pseudo-expert loss weight	0.5
Pseudo-expert applied to all refinement layers	false
Trajectory imitation loss weight	1.0
Final score loss weight	1.0
Scorer sub-score loss (including EP)	BCE-with-logits
Prediction cross-entropy loss weight	1.0
Prediction L1 loss weight	0.1
Prediction area loss weight	2.0

evaluated using the numerically stable BCE-with-logits implementation. The scorer loss aggregates these component losses; sigmoid outputs, rather than raw logits, are used for metric composition and ranking. Continuous EP values and fractional discrete sub-scores remain soft targets and are not thresholded into binary labels. In our ablation, a separate regression loss for EP did not improve performance, so the released implementation uses the unified objective. A possible explanation is that EP targets need not be uniformly distributed over $[0, 1]$ and may concentrate near the endpoints, with low-progress or failed trajectories contributing near-zero outcomes. This is a hypothesis for the observed ablation result, not a claim that EP is intrinsically binary. The unified objective applies to scorer sub-score supervision; the trajectory-imitation and set-distillation losses retain their stated ℓ_1 form.

Table 17: **Pseudo-expert trajectory generation hyperparameters.** Pseudo-expert trajectories are generated from interpretable candidate families and filtered by evaluator-based trajectory quality.

Hyperparameter	Value
Number of poses	8
Sampling interval	0.5 s
Speed candidates	$(0, 2, 4, 6, 8, 10, 12, 15)$ m/s
Regular lateral offsets	$(-3.5, -2.0, -1.0, -0.5, 0, 0.5, 1.0, 2.0, 3.5)$ m
Off-road lateral offsets	$(-7.0, -5.5, 5.5, 7.0)$ m
Transition portions	$(0.35, 0.6, 1.0)$
Acceleration rates	$(-2.0, -1.0, -0.5, 0.5, 1.0, 2.0)$
Score bins	$(0.0, 0.2, 0.4, 0.6, 0.8, 1.01)$
Progress bins	$(0.0, 0.2, 0.5, 0.8, 1.01)$
Retained trajectories per scene	50
Maximum scored candidates per scene	180

Table 18: **Stage-2 alternating self-distillation hyperparameters.** Stage 2 alternates between scorer fitting and generator refinement. The visual backbone is frozen during Stage 2, while the scorer remains trainable.

Hyperparameter	Value
Initialization	Stage-1 checkpoint
Number of cycles	30
Scorer-fitting epochs per cycle	1
Generator-refinement epochs per cycle	1
Total epochs	60
Batch size	32
Number of GPUs	4
DataLoader workers	8
Base learning rate	3×10^{-5}
Starting phase	scorer fitting
Freeze visual backbone	true
Detach proposals in scorer	true
Sanitize proposals	true
Proposal XY limit	100.0
Ray scoring	enabled
Ray threads per node	96
Trajectory loss weight	0.1
Intermediate loss weight	0.02
Final score loss weight	1.0
Scorer sub-score loss (including EP)	BCE-with-logits
Pareto guidance weight	1.0
Teacher stability weight	0.05
Pareto set maximum size	8
Pareto set minimum size	2
Teacher reward threshold	0.4

G Fixed-Proposal Scorer Diagnostic

The proposal diversity analysis shows that CLOVER substantially improves both proposal quality and proposal diversity. However, it also reveals a new bottleneck: once many high-quality candidates are available, final performance depends heavily on whether the scorer can correctly rank near-tied but behaviorally different trajectories. A better scorer can improve the final selected trajectory directly, and in our framework it can also serve as a differentiable mediator for refining the generator during self-distillation. This motivates a standardized diagnostic protocol that focuses specifically on trajectory-level scorer design.

We introduce a fixed-proposal scorer diagnostic protocol under a shared CLOVER-generated candidate pool. For each scene in NAVSIM `navtrain` and `navtest`, we use the trained CLOVER

Table 19: **Compute resources and training time.** Training time includes real-time trajectory scoring with the NAVSIM evaluator, which depends on CPU throughput in addition to GPU training speed.

Item	Value
GPUs	4 NVIDIA A100
CPU	AMD EPYC 7742 64-Core Processor
Stage-1 training time	30–40 hours
Stage-2 training time	60–80 hours
Total training time	4–5 days

Table 20: **Fixed-proposal scorer diagnostic.** All scorers are trained and evaluated on the same CLOVER-generated 64-candidate pools with true PDMS sub-score labels. Train and test scores denote the true PDMS of the scorer-selected top-1 trajectory on `navtrain` and `navtest`, respectively. These diagnostic scores evaluate scorer ranking quality under fixed proposals and are not directly comparable to full-system CLOVER results.

Scorer backbone	Train score ↑	Test score ↑
DINOv2-Small	95.0	92.4
DINOv2-Base	95.3	92.6
DINOv3-Small	95.1	92.4
DINOv3-Base	95.1	92.7
Wan2.2-5B features	**97.7**	**92.8**

generator to produce 64 candidate trajectories. Each candidate is then evaluated by the true PDMS evaluator to obtain scalar scores and interpretable sub-score labels. This produces a scorer-only dataset:

$$\mathcal{D}_{\text{score}} = \left\{ \left(o, \{\tau_i\}_{i=1}^{64}, \{\mathbf{r}_i^{\text{true}}\}_{i=1}^{64} \right) \right\}. \tag{109}$$

A scorer architecture is trained to predict sub-scores for each fixed candidate and selects the trajectory with the highest composed score:

$$i_\phi^* = \arg\max_i \Gamma_{\text{PDMS}} \left(S_\phi(o, \tau_i) \right). \tag{110}$$

We report the true PDMS of the selected trajectory:

$$\text{Score} = \frac{1}{|\mathcal{D}|} \sum_{o \in \mathcal{D}} s_{i_\phi^*}^{\text{true}}. \tag{111}$$

This protocol isolates scorer quality from proposal generation, because all scorer architectures are evaluated under the same candidate trajectories and the same true labels. It is lightweight, reproducible, and directly targets the ranking component of proposal-selection planners. We deliberately treat it as a scorer diagnostic rather than a full planner benchmark: it does not include generator-scorer closed-loop training, and the fixed offline candidate pool differs from a fully deployed CLOVER system. Therefore, its scores should not be compared directly with full-system PDMS results.

Table 20 compares several scorer backbones under this protocol. DINOv2-Small is the backbone used in our main CLOVER model. Larger DINO backbones provide modest test improvements, while the Wan2.2-5B video-generation backbone achieves the highest diagnostic test score. For Wan2.2-5B, we LoRA fine-tune the video model on NAVSIM and use intermediate block-15 features for scoring. This result is consistent with the intuition that video or world-model features may better capture future-scene-dependent factors required by PDMS. However, Wan2.2-5B also shows a clear train-test gap and is far more expensive than the DINOv2-Small scorer. Similarly, DINO base variants increase training cost in our closed-loop refinement pipeline. For these reasons, we keep DINOv2-Small as the deployed scorer in CLOVER and use the larger backbones only as diagnostic references.

The diagnostic results suggest two observations. First, scorer design has measurable impact even when the candidate pool is fixed, confirming that trajectory ranking is a meaningful bottleneck after proposal diversity has been improved. Second, stronger visual or video features can improve scorer performance, but the gain must be balanced against computational cost and overfitting risk. The large

train-test gap of Wan2.2-5B indicates high representational capacity, but also suggests that using such a model as an online scorer inside the full CLOVER training loop would be impractical under our current training budget. We therefore release this scorer diagnostic protocol to support future studies on trajectory-level scoring, score calibration, and ranking architectures under a shared high-quality candidate pool.

H SEED STABILITY

The official NAVSIM evaluation is deterministic given a trained model and a fixed benchmark split. Therefore, the main source of stochastic variation in our reported results comes from training randomness, including parameter initialization, dataloader order, and stochastic optimization. To assess this effect, we repeat the main CLOVER training with three random seeds under the same architecture, data split, hyperparameters, and evaluation protocol.

Table 21 reports the seed-level variation. Across three runs, the final NAVSIM v1 PDMS differs by less than 0.02 points. This variation is below the one-decimal reporting precision used in the main benchmark tables and does not affect any conclusion or ranking trend. We therefore report the official single-run benchmark scores in the main paper and use this experiment only to verify run-to-run stability.

Table 21: **Seed stability of CLOVER.** All runs use the same architecture, training schedule, data split, and evaluation protocol. The observed variation is below the one-decimal reporting precision of the main benchmark tables.

Setting	Seed 1	Seed 2	Seed 3	Max–Min
NAVSIM v1 PDMS	94.488	94.502	94.492	0.014
NAVSIM v2 EPDMS	90.354	90.358	90.368	0.014
NAVSIM v2 EPDMS*	87.201	87.207	87.210	0.009

I ADDITIONAL PROPOSAL DIVERSITY METRICS

We provide additional proposal-set statistics comparing the DrivoR baseline, Stage 1 diversity pretraining, and Stage 2 self-distillation refinement. All metrics are averaged over 12,146 common scenes, with 64 proposals per scene. The true PDMS scores are used only for analysis. Reported differences and oracle-selection gaps are computed from the displayed values.

Quality metrics. Selected PDMS is the true PDMS score of the trajectory selected by the learned scorer. Oracle@64 is the best true PDMS score among all 64 proposals. Mean PDMS over 64 measures the average quality of the full proposal set. Gap is the difference between Oracle@64 and Selected PDMS. Count(PDMS>0.95), Count(PDMS>0.90), and Count(PDMS<0.50) summarize the proposal score distribution.

Diversity metrics. Pairwise ADE/FDE measure the average pairwise distance between generated proposals. Endpoint Std Radius and Endpoint Area measure the spread of final trajectory endpoints. Trajectory Effective Rank is computed by flattening each 8-step (x, y) trajectory into a vector, centering the proposal set, and computing the entropy-based effective rank of the singular-value spectrum. Endpoint Cluster Count@r greedily counts endpoint clusters under radius r.

Quality-aware diversity. To avoid overestimating diversity due to low-quality outliers, we also report diversity after filtering proposals with PDMS≥ 0.8. Qualified Cluster Count@2m measures the number of distinct endpoint clusters among qualified proposals. Qualified Pairwise ADE/FDE measures diversity within qualified proposals. Top-6 Real Pairwise ADE/FDE measures diversity among the six proposals with the highest true PDMS scores in each scene.

Interpretation. The three stages exhibit distinct behavior. Stage 1 is an expansion stage: it increases the oracle upper bound and raw geometric diversity, but also introduces more low-score

Table 22: **Top-K proposal quality under scorer ranking.** TopK-Oracle@K reports the best true PDMS score among the scorer-ranked top-K proposals. TopK-Mean@K reports the average true PDMS score within the scorer-ranked top-K proposals.

Metric	K	Baseline	Stage 1	Stage 2	Stage1 - Base	Stage2 - Stage1
TopK-Oracle@K	1	0.9369	0.9413	**0.9448**	+0.0044	+0.0035
TopK-Oracle@K	2	0.9461	0.9514	**0.9517**	+0.0053	+0.0003
TopK-Oracle@K	3	0.9512	**0.9570**	0.9554	+0.0058	-0.0016
TopK-Oracle@K	6	0.9603	**0.9660**	0.9625	+0.0057	-0.0035
TopK-Mean@K	1	0.9369	0.9413	**0.9448**	+0.0044	+0.0035
TopK-Mean@K	2	0.9353	0.9402	**0.9435**	+0.0049	+0.0033
TopK-Mean@K	3	0.9345	0.9383	**0.9430**	+0.0038	+0.0047
TopK-Mean@K	6	0.9320	0.9346	**0.9416**	+0.0026	+0.0070

Table 23: **Full proposal-set quality statistics.** Stage 1 achieves the highest oracle upper bound but also introduces a lower-quality tail. Stage 2 refines the expanded proposal distribution and achieves the best selected score, smallest oracle-selection gap, highest proposal mean, and fewest low-score proposals.

Metric	Baseline	Stage 1	Stage 2	Stage1 - Base	Stage2 - Stage1	Stage2 - Base
Selected PDMS ↑	0.9369	0.9413	**0.9448**	+0.0044	+0.0035	+0.0079
Oracle@64 PDMS ↑	0.9933	**0.9976**	0.9939	+0.0043	-0.0037	+0.0006
Gap ↓	0.0564	0.0563	**0.0491**	-0.0001	-0.0072	-0.0073
Mean PDMS over 64 ↑	0.7972	0.7374	**0.8277**	-0.0598	+0.0903	+0.0305
Std PDMS over 64	0.2152	**0.3013**	0.2235	+0.0861	-0.0778	+0.0083
Count(PDMS>0.95) ↑	30.94	24.78	**31.97**	-6.16	+7.19	+1.03
Count(PDMS>0.90) ↑	**40.24**	32.13	39.90	-8.11	+7.77	-0.34
Count(PDMS<0.50) ↓	9.05	12.21	**6.83**	+3.16	-5.38	-2.22

proposals. Stage 2 is a refinement stage: it preserves much of the expanded support while improving scorer-selected top-1, top-K mean quality, full-set mean quality, and low-score tail suppression. This supports the design of CLOVER as a two-stage process: first broaden the proposal space, then refine it toward high-value regions.

I.1 Additional Qualitative Comparisons of Proposal Diversity

Figure 5 presents additional qualitative comparisons between the DrivoR baseline and CLOVER on six more scenes, following the same visualization protocol as Figure 3 in the main text. For each scene, the left two panels show the DrivoR baseline and the right two panels show CLOVER after Stage-2 refinement. For each method, we visualize all 64 generated candidate trajectories in both the front-view image and bird's-eye view. Candidate trajectories are colored by their true PDMS scores for analysis, while the human trajectory, the scorer-selected trajectory, and the oracle trajectory with the highest true PDMS are highlighted explicitly.

These additional examples consistently support the same conclusion as in the main text. The DrivoR baseline tends to concentrate proposals around a narrow single mode, often with limited lateral or longitudinal variation. In contrast, CLOVER produces a broader set of feasible candidates, covering richer within-lane offsets, progress patterns, and alternative motion branches. Importantly, this increased diversity is not merely caused by low-quality outliers: many diverse candidates remain high-scoring under the evaluator, and the scorer-selected trajectory typically remains close to the oracle high-score mode. Together with the quantitative results in Section 5.3, these visualizations further confirm that CLOVER improves proposal diversity while preserving strong proposal quality.

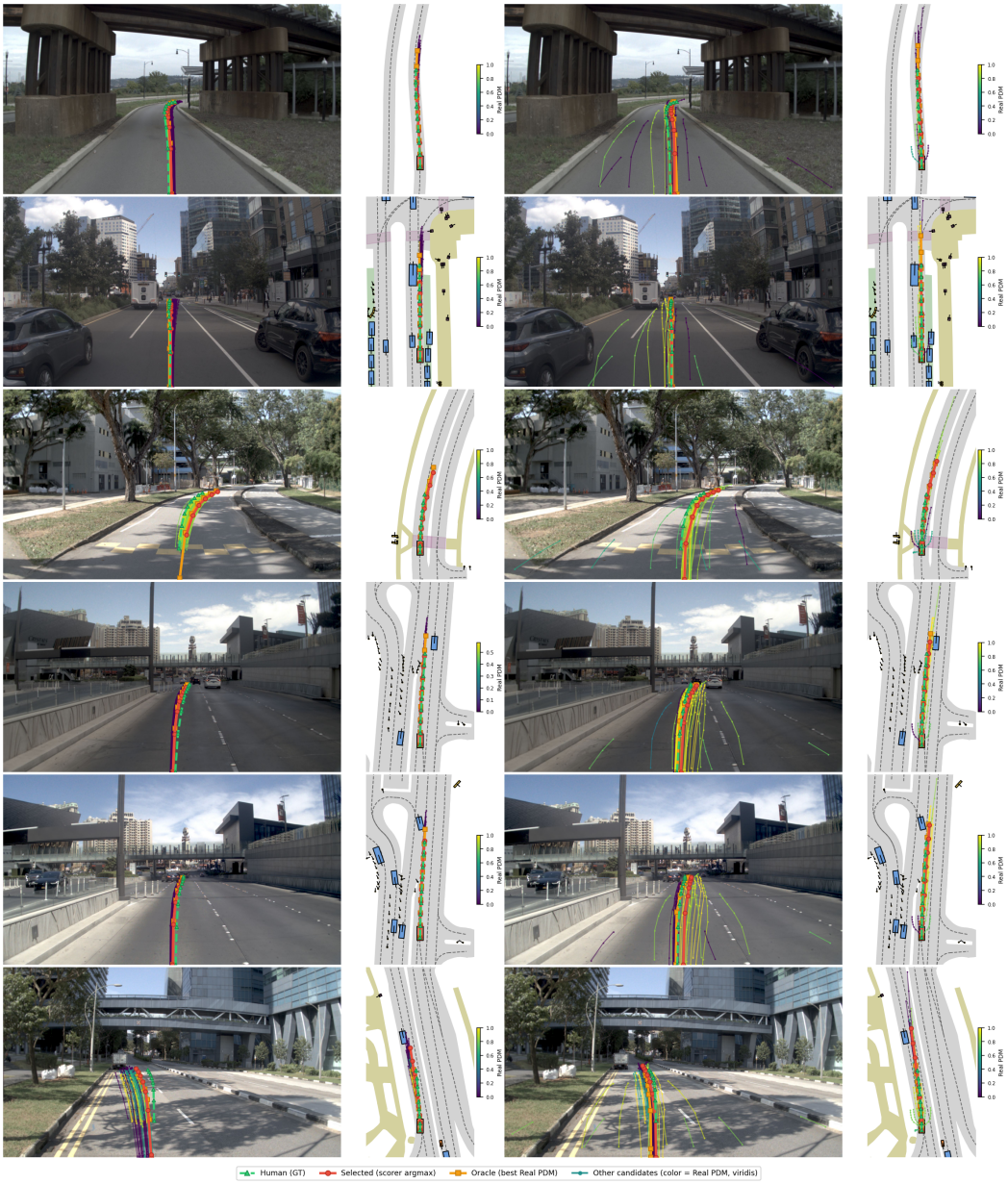

Figure 5: **Additional qualitative comparisons of proposal diversity on six scenes.** In each row, the left two panels show the DrivoR baseline and the right two panels show CLOVER after Stage-2 refinement. For each method, we visualize all 64 candidate trajectories in the front-view image and in bird's-eye view. Candidate trajectories are colored by their true PDMS scores for analysis. Green triangles denote the human trajectory, red circles denote the scorer-selected trajectory, orange squares denote the oracle trajectory with the highest true PDMS, and the remaining candidates are colored by real PDMS using the viridis colormap. Across all six scenes, the baseline proposals are concentrated around a narrow mode, whereas CLOVER consistently covers a wider set of feasible and often high-scoring candidates.

Table 24: **Full proposal diversity statistics.** Stage 1 produces the widest raw proposal spread, while Stage 2 slightly contracts this spread but remains substantially more diverse than the baseline. Higher is better for all metrics.

Metric	Baseline	Stage 1	Stage 2	Stage1 - Base	Stage2 - Stage1	Stage2 - Base
Pairwise ADE	1.8004	**5.9724**	5.1956	+4.1720	-0.7768	+3.3952
Pairwise FDE	4.4660	**10.6973**	8.2536	+6.2313	-2.4437	+3.7876
Endpoint Std Radius	3.9033	**10.2638**	7.9477	+6.3605	-2.3161	+4.0444
Endpoint Area	5.7837	**46.1758**	25.5615	+40.3921	-20.6143	+19.7778
Trajectory Effective Rank	1.1401	1.2362	**1.2748**	+0.0961	+0.0386	+0.1347

Table 25: **Quality-aware proposal diversity.** Diversity is computed either after filtering proposals with PDMS≥ 0.8 or within the six highest-scoring proposals according to the true evaluator. This avoids attributing diversity gains only to low-quality outliers.

Metric	Baseline	Stage 1	Stage 2	Stage1 - Base	Stage2 - Stage1	Stage2 - Base
Qualified Count(PDMS≥ 0.8)	50.16	43.97	**50.44**	-6.19	+6.47	+0.28
Qualified Cluster Count@2m	6.02	**10.91**	8.71	+4.89	-2.20	+2.69
Qualified Pairwise ADE	1.5576	**3.7988**	3.6727	+2.2412	-0.1261	+2.1151
Qualified Pairwise FDE	3.8024	**6.8121**	5.6674	+3.0097	-1.1447	+1.8650
Top-6 Real Pairwise ADE	0.9110	**2.1111**	1.7507	+1.2001	-0.3604	+0.8397
Top-6 Real Pairwise FDE	2.1955	**3.5542**	2.5910	+1.3587	-0.9632	+0.3955

Table 26: **Endpoint cluster count under different radii.** Endpoint clusters are computed by greedy clustering over final trajectory endpoints. Stage 1 produces the most endpoint branches, while Stage 2 remains consistently more diverse than the baseline at all radii.

Radius	Baseline	Stage 1	Stage 2
1m	16.77	**32.77**	23.20
2m	8.07	**19.81**	13.81
3m	5.29	**14.65**	9.88
4m	3.97	**11.24**	8.07

J COMPLEMENTARITY OF VECTOR-PARETO AND SCALAR TOP-k TARGETS

Protocol. To quantify the additional supervision supplied by vector-Pareto selection, we reproduce the exact retained-set construction used during Stage-2 training on 12,146 NAVSIM-v1 `navtest` scenes, with 64 proposals per scene. For scene i, let P_i denote the retained vector-Pareto target set and T_i the scalar-score Top-8 set from the same proposal pool. Both selectors have an eight-target budget, but P_i may contain fewer than eight trajectories; the statistics concern the retained training targets rather than the entire untruncated Pareto front. There are 96,506 retained Pareto targets in total, or 7.95 per scene on average. Trajectory-level overlap rates are pooled across scenes, whereas scene-level rates count each scene equally.

Table 27: **Overlap between retained vector-Pareto targets and scalar Top-8.** Both sets are selected from the same 64 proposals in each of 12,146 NAVSIM-v1 `navtest` scenes. $N = 12{,}146$ and $\mathbf{1}[\cdot]$ denotes an indicator. Full Pareto coverage allows $|P_i| < 8$, whereas set equality requires mutual inclusion.

Statistic	Definition	Result
Pareto covered by Top-8	$\sum_i \lvert P_i \cap T_i \rvert / \sum_i \lvert P_i \rvert$	76.74%
Pareto-only targets	$\sum_i \lvert P_i \setminus T_i \rvert / \sum_i \lvert P_i \rvert$	23.26%
Top-8 targets in Pareto	$\sum_i \lvert P_i \cap T_i \rvert / \sum_i \lvert T_i \rvert$	76.21%
Top-8-only targets	$\sum_i \lvert T_i \setminus P_i \rvert / \sum_i \lvert T_i \rvert$	23.79%
Scenes with identical sets	$N^{-1} \sum_i \mathbf{1}[P_i = T_i]$	18.45%
Scenes with full Pareto coverage	$N^{-1} \sum_i \mathbf{1}[P_i \subseteq T_i]$	19.08%
Scenes with additional Pareto targets	$N^{-1} \sum_i \mathbf{1}[P_i \setminus T_i \neq \varnothing]$	80.92%

Additional targets beyond scalar ranking. Table 27 shows that scalar Top-8 covers 76.74% of the retained Pareto targets, leaving 23.26% (22,450 of 96,506) uncovered. Only 19.08% of scenes have their retained Pareto set fully covered by Top-8; thus, 80.92% contain at least one additional non-dominated supervision target. Conversely, 23.79% of scalar Top-8 targets are absent from the retained Pareto set, and the two sets are identical in only 18.45% of scenes. This bidirectional difference supports using the two target sets together: scalar ranking emphasizes aggregate score, while vector-Pareto selection retains additional trade-offs among metric components.

Comparison with geometric de-duplication. We also compare against a geometry-aware scalar Top-8 selector that imposes a minimum trajectory-ADE separation of 0.25 m. Its coverage of the retained Pareto targets, measured with the same pooled Pareto denominator, is 54.52%, compared with 76.74% for scalar Top-8 without this constraint. Thus, this geometric de-duplication baseline does not recover the additional objective-space targets selected by vector-Pareto construction. These held-out statistics measure target-set complementarity rather than the causal contribution of each target to a generator update; they complement the end-to-end teacher-set ablation in Table 9.

K Supplementary Reactive Closed-loop Evaluation on Bench2Drive

K.1 Results and Scope

Bench2Drive (Jia et al., 2024) evaluates driving through reactive closed-loop interactions in CARLA. We adapt CLOVER's two-stage training to this setting and report the route-level Success Rate (SR) and Driving Score (DS). Table 28 compares the two CLOVER stages and lists published reference results. Stage 2 improves SR from 68.6% to 70.5% and DS from 86.8 to 89.2, corresponding to gains of 1.9 percentage points and 2.4 points, respectively. These results provide supplementary evidence for the benefit of refinement beyond the NAVSIM setting. The adaptation uses a local scoring surrogate and approximate pseudo-expert construction, as detailed below; it does not directly optimize the full route-level DS.

Table 28: **Supplementary Bench2Drive results.** Published reference values are reproduced from Table 4 of Sun et al. (2026), rather than rerun by us. * denotes expert feature distillation. SR is the overall success rate. All scores are rounded to one decimal.

Method	SR (%) ↑	DS ↑
UniAD	16.4	45.8
VAD	15.0	42.4
SparseDrive	16.7	44.5
GenAD	15.9	44.8
DiFSD	21.0	52.0
DriveTransformer	35.0	63.5
Hydra-NeXt	50.0	73.9
SimLingo	67.3	86.0
HiP-AD	69.1	86.8
TCP-traj*	30.0	59.9
ThinkTwice*	31.2	62.4
DriveAdapter*	33.1	64.2
SparseDriveV2	70.0	89.2
CLOVER Stage 1	68.6	86.8
CLOVER Stage 2	**70.5**	**89.2**

K.2 Pseudo-expert Construction with Interaction Filtering

Constructing reliable pseudo-expert trajectories is harder in a reactive environment because other agents may respond to the ego plan. For training scenes, we first generate a candidate pseudo-expert pool using the trajectory construction procedure in Appendix B, with other agents temporarily following their recorded trajectories. We then introduce scene interactions in a second filtering step and discard trajectories with major failures or insufficient travel progress. The retained trajectories supply Stage-1 pseudo-expert supervision. This procedure is a practical approximation: initializing the pool from logged behavior and subsequently filtering it does not exhaustively account for the counterfactual responses of other agents to every possible ego trajectory.

K.3 Local Scorer Supervision

The official route-level DS combines final route completion with accumulated infractions (Jia et al., 2024). It cannot be faithfully assigned to a single 3-second trajectory window without accounting for the rest of the route. We therefore train the scorer to estimate the local quality of current-frame candidates and their relative ordering within the same scene.

Offline label construction. For each training frame, we generate future trajectories with a 3-second horizon, replacing the 4-second NAVSIM horizon. Trajectories represented at 0.5-second intervals are interpolated to 10 Hz for label computation. Offline labels use the map, global route, logged ego future, future agent boxes, traffic-light states, and stop-sign regions. Logged future information is used

to construct training supervision, not supplied to the deployed scorer. The supervision components are:

- **No collision (NOC):** separate collision labels for pedestrians, vehicles, and static objects distinguish the corresponding hazards.
- **Drivable-area compliance (DAC):** the fraction of candidate travel distance inside the legal drivable area.
- **Time to collision (TTC):** a continuous soft label based on minimum TTC, assigning higher scores to larger safety margins.
- **Ego progress (EP):** longitudinal progress along the global route graph, normalized relative to the logged expert's progress over the same 3-second interval. When a red light, pedestrian, or leading vehicle requires stopping, progress beyond the expert is not rewarded; candidates are instead scored by whether they stop near a safe stopping location.
- **Driving-direction compliance (DDC):** a combination of wrong-way distance, monotonicity of route projection, and alignment with the legal lane direction, while allowing legal lane changes and obstacle avoidance.
- **Traffic-rule compliance (Rule):** local compliance with traffic-light and stop-sign requirements.
- **Comfort:** soft compliance scores for longitudinal and lateral acceleration, jerk, yaw rate, and yaw acceleration, using the official Bench2Drive smoothness thresholds as reference bounds.

Candidate score. We use a NAVSIM-like scoring structure with an additional multiplicative traffic-rule compliance term (Rule):

$$\begin{aligned} Q_{\text{local}}(\tau) = \text{NOC} \cdot \text{DAC} \cdot \text{DDC} \cdot \text{Rule} \\ \times \big(5\,\text{TTC} + 5\,\text{EP} + 2\,\text{Comfort}\big). \end{aligned} \tag{112}$$

No collision, drivable-area compliance, driving-direction compliance, and traffic-rule compliance form the multiplicative gates, while progress, TTC, and comfort form the additive local utility. For collision and traffic-rule supervision, NOC and Rule are binary: a collision or rule violation sets the corresponding gate to zero, and hence makes the local teacher score zero; otherwise that gate is one. No graded infraction-penalty coefficients are applied. Equation 112 is therefore distinct from the official route-level DS formula. The additive weights are 5, 5, and 2; their common normalization would not change candidate ordering.

Training objective. Training follows the NAVSIM scorer-fitting and teacher-set distillation scheme, with evaluator targets replaced by the Bench2Drive local metrics. All supervised metric components use BCE-with-logits, retaining continuous values as soft targets in $[0, 1]$. Equation 112 composes the offline labels into local teacher scores and the predicted sub-scores into candidate scores. During generator refinement, the predicted scores and sub-score vectors select top-k and vector-Pareto teacher targets within each scene, followed by the same set-distillation and stability losses as in Section 3.3. This trains candidate selection and refinement without requiring the composed local score to have a globally calibrated absolute meaning across different scenes. At inference, the selected candidate maximizes the composed predicted local score.

K.4 Trajectory-to-Control Conversion

After selecting a trajectory, we convert it into steering, throttle, and brake commands using the path–velocity control scheme described by Sun et al. (2026). The selected trajectory is decomposed into a geometric path and a velocity profile. For lateral control, the preview distance is

$$d = (0.5\,\text{s})\, v_{\text{ego}} + 2.5\,\text{m}, \tag{113}$$

where v_{ego} is measured in m/s. The path point whose cumulative distance is closest to d determines the steering preview target. For longitudinal control, the first speed in the velocity profile is the target used to compute throttle and brake. This controller is adopted as an existing implementation choice; we do not perform a separate controller ablation.

K.5 Limitations of the Adaptation

The present Bench2Drive adaptation remains approximate, particularly in pseudo-expert construction and the rollout information used for scorer supervision. Interaction filtering improves the logged-trajectory candidate pool but does not provide comprehensive reactive rollouts for every candidate. Likewise, offline labels based on recorded future agent boxes cannot fully capture how other agents would respond to a different ego action. The 3-second surrogate also omits longer-term consequences and accumulated route-level infractions. Thus, the reported closed-loop improvements support this practical adaptation, while leaving more faithful interaction-aware pseudo-expert generation and scorer supervision as future work.