World4Scorer:

Outcome-Grounded World Modeling for Autonomous Driving

Anonymous authors
Paper under double-blind review

Abstract

Autonomous driving requires choosing a safe and efficient plan as surrounding traffic evolves. Generate-and-select planners propose multiple trajectories and score them for execution, and they have outperformed representative direct-prediction baselines on NAVSIM. Their scorer must compare plans that were never executed. Driving logs record the future of only the executed trajectory, so matching the logged future can leave predictions for the alternatives unconstrained; a simulator, in contrast, can label the outcome of every candidate. We introduce World4Scorer, which builds the scorer as a trajectory-conditioned JEPA-style predictor: it predicts a state for each candidate and reads the candidate's scores from that state. Simulator outcome labels supervise the states of all candidates, and the observed future of the executed trajectory anchors the predictor to real scene evolution. Because one predictor produces every candidate's state, the anchor can constrain shared parameters used to score unexecuted plans, while the future itself is needed only during training. Generated candidates mostly score well, so a scene-matched bank adds low-scoring plans to the outcome supervision; framewise choices can conflict, so inertial re-ranking keeps consecutive selections consistent. World4Scorer achieves state-of-the-art NAVSIM-v2 performance and a strong adapted-system result on closed-loop Bench2Drive. With the LeWM world model and planning budget fixed, outcome-based scoring also improves manipulation planning on the OGBench-Cube benchmark.

1 Introduction

Autonomous driving requires turning sensor observations into safe, efficient, and comfortable motion. Direct trajectory-prediction methods such as TransFuser (Chitta et al., 2023) and UniAD (Hu et al., 2023) map scene features to a driving plan. Recent generate-and-select planners have achieved stronger performance than representative direct-prediction baselines on NAVSIM (Liao et al., 2025; Li et al., 2025d; Kirby et al., 2026). They use a trajectory generator to propose alternatives and a learned scorer to choose one for execution (Figure 1a). To benefit from these alternatives, the scorer must identify which plan best balances safety, progress, and comfort.

Scoring a plan requires assessing its consequences as surrounding traffic evolves. World models provide predictive features for planning (Li et al., 2025a; Hong et al., 2026) and candidate evaluation (Li et al., 2025b) (Figure 1b). Their training faces a supervision gap: each driving log records the future of the executed trajectory, while alternative trajectories have no corresponding future observations. Matching the logged future can leave predictions for unobserved alternatives unconstrained, even though these predictions decide which plan is selected.

Candidate outcomes fill this gap. Simulation can label whether each alternative plan collides, leaves the drivable area, or makes progress, without an observed visual future for that plan. When a predictor produces the state from which each candidate is scored, these labels supervise the state of every candidate, including plans that were never executed. The observed future serves as an anchor: it supervises the state of the executed trajectory and ties the shared predictor to real scene evolution.

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.

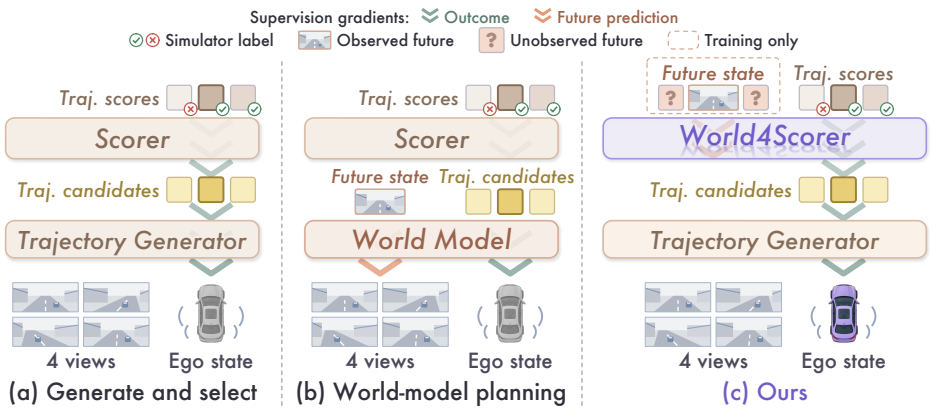

Figure 1: **Gradient routing in generator–scorer planning.** (a) Outcome labels train the scorer. (b) A world model adds future prediction, but only the executed trajectory has an observed future. (c) In World4Scorer, outcome labels supervise the state of every candidate, and the observed future anchors the predictor.

We introduce World4Scorer, an *outcome-grounded world modeling* framework in which a trajectory-conditioned predictor is the scorer (Figure 1c). Following the joint-embedding predictive architecture (JEPA) (Assran et al., 2023; Bardes et al., 2024), the predictor maps each candidate to a state; score heads read candidate outcomes from this state, and a visual readout predicts the future features of the executed trajectory. The core is this shared predictor, trained by both candidate outcomes and the observed future. As support, a perturbed candidate bank widens outcome coverage, and training-free inertial re-ranking checks continuity with the previous plan.

- **Outcome-grounded world modeling.** We build the candidate scorer as a trajectory-conditioned predictor. Simulator outcomes supervise the predicted state of every generated and bank candidate, including plans that were never executed, while the observed future of the executed plan anchors the shared predictor to real scene evolution.
- **Complementary supervision.** Outcome labels reach every candidate; the observed future reaches one trajectory yet constrains the shared predictor. In controlled tests, an observed-future target improves planning but a current-frame target does not, mismatched outcome labels degrade ranking, and reassigning candidate states lowers PDMS in every trial.
- **Evidence across benchmarks and domains.** World4Scorer achieves state-of-the-art performance on NAVSIM-v2 and leads the compared methods on NAVSIM-v1. Cross-setting tests extend these results: an adapted system is strong in closed-loop Bench2Drive, and outcome-based scoring improves manipulation with a frozen world model.

2 Related Work

End-to-end autonomous driving. TransFuser (Chitta et al., 2023) and UniAD (Hu et al., 2023) regress a trajectory from scene features. Candidate-based planners compare alternatives from vocabularies or generators. VADv2 (Jiang et al., 2026) predicts probabilities over a fixed vocabulary. DiffusionDrive (Liao et al., 2025) refines anchors and selects by learned confidence. GTRS (Li et al., 2025d) scores vocabulary and generated candidates, and DriveSuprim (Yao et al., 2026a) filters candidates from coarse to fine. CLOVER (Ang et al., 2026) trains its generator on evaluator-filtered pseudo-experts and its scorer on evaluator sub-scores. DrivoR (Kirby et al., 2026) learns proposals and simulator-supervised scores with separate decoders. We follow its generator–scorer framework and make the scorer a predictor trained by simulator labels, anchored by observed futures.

Predictive representations for planning. Future prediction supports both trajectory generation (Li et al., 2025a; Xia et al., 2026; Hong et al., 2026) and candidate evaluation (Li et al., 2025b; Kim et al., 2026; Zheng et al., 2025). Drive-JEPA (Wang et al., 2026) uses video-JEPA pretraining to provide predictive features for planning. We focus on the supervision available during candidate comparison: observed futures cover executed trajectories, whereas simulator outcomes cover every candidate; World4Scorer trains one predictor with both.

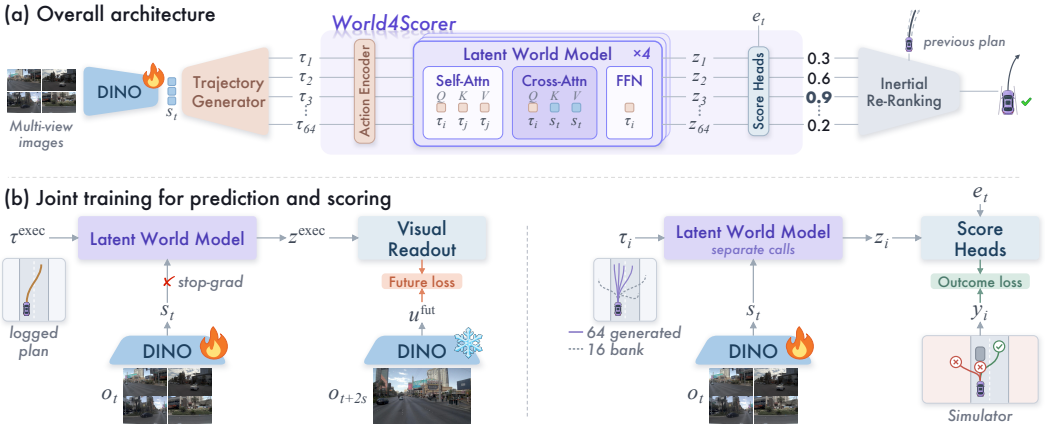

Figure 2: **Overview of World4Scorer.** (a) The trajectory generator proposes 64 candidates, scored by World4Scorer before inertial re-ranking. (b) The visual readout and score heads train the shared predictor: future observations supervise visual prediction for the executed trajectory; outcome labels supervise scoring for all generated candidates and 16 bank trajectories.

3 World4Scorer

World4Scorer follows a generator–scorer framework: it proposes trajectories, predicts a state for each candidate, and scores these states before inertial re-ranking (Figure 2). We use "world model" in a decision-focused sense: latent states carry candidate outcomes, not full futures.

3.1 Overview

At inference (Figure 2a), DINOv2 (Oquab et al., 2024) with LoRA (Hu et al., 2022) encodes four camera views into scene tokens s_t. A transformer generates $N = 64$ trajectories, $\mathcal{C}_t = \{\tau_{t,i}\}_{i=1}^{N}$. The predictor maps each trajectory query to a state using the scene tokens. Score heads read each state together with the ego state. Inertial re-ranking then adds compatibility with the previous plan.

At training time (Figure 2b), winner-take-all imitation supervises the generator. Outcome labels supervise all 64 generated candidates and 16 trajectories sampled from a scene-matched bank. A separate query for the logged trajectory predicts its observed visual future. These three query sets use the same predictor, allowing visual prediction and outcome scoring to train shared parameters.

3.2 Shared Prediction and Scoring

An action encoder ϕ maps each candidate trajectory to a query token. The shared predictor P (Latent World Model in Figure 2) processes all candidate queries together with the scene tokens:

$$q_{t,i} = \phi(\mathrm{sg}[\tau_{t,i}]), \quad Q_t = [q_{t,1}, \ldots, q_{t,N}], \quad Z_t = P(Q_t, s_t). \tag{1}$$

Its four transformer blocks exchange information among candidates through self-attention and incorporate scene information through cross-attention, followed by output normalization. Row $z_{t,i}$ of Z_t is the predicted state for candidate i, and sg stops gradients.

The score heads H read each candidate state after adding the encoded ego state e_t: $\boldsymbol{\ell}_{t,i} = H(z_{t,i} + e_t)$. They predict collision avoidance (NC), drivable-area compliance (DAC), driving direction (DDC), time-to-collision (TTC), progress (EP), and comfort (C). Let $p_{t,i,m} = \sigma(\ell_{t,i,m})$, $\mathcal{G} = \{\mathrm{NC}, \mathrm{DAC}, \mathrm{DDC}\}$, and $\mathcal{A} = \{\mathrm{TTC}, \mathrm{EP}, \mathrm{C}\}$. The ranking score is

$$\hat{J}_{t,i} = \sum_{m \in \mathcal{G}} \alpha_m \log p_{t,i,m} + \log\left(\sum_{m \in \mathcal{A}} \beta_m p_{t,i,m} \right). \tag{2}$$

Here α_m and β_m are per-benchmark weights (Appendix A.2). The highest-scoring candidate defines the framewise choice; inertial re-ranking further incorporates previous-plan compatibility.

Detaching candidate coordinates prevents scoring gradients from reaching the generator through its trajectory outputs. The scoring loss still trains ϕ, P, H, and the scene features.

3.3 Outcome-Grounded Predictive Learning

Only the executed trajectory has an observed future; generated and bank candidates lack future observations but have outcome labels. Outcome labels supervise every candidate state, and the observed future anchors the predictor through the executed trajectory (Figure 2b).

Trajectory-conditioned JEPA. Following JEPA (Assran et al., 2023; Bardes et al., 2024), the predictor takes scene tokens and the logged trajectory query. Its target is the frozen DINOv2 embedding of the observed front view two seconds after the current frame, mean-pooled to 384 dimensions:

$$z_t^{\text{exec}} = P([\phi(\tau_t^{\text{exec}})], \text{sg}[s_t]), \qquad u_t^{\text{fut}} = f_{\text{DINO}}(o_{t+2\text{s}}). \tag{3}$$

The linear visual readout g predicts this feature with a cosine loss:

$$\mathcal{L}_{\text{fut}} = \frac{1}{M} \sum_{n=1}^{M} \left[1 - \text{sim}_{\cos}\big(g(z_n^{\text{exec}}), u_n^{\text{fut}}\big)\right]. \tag{4}$$

Here n indexes M scenes. With scene tokens detached, this loss updates ϕ, P, and g; at deployment, the predictor needs only current observations and candidate trajectories. Because every query passes through the same P, this single target also constrains the parameters that produce each candidate's state. The self-attention query and key projections receive no gradient from this single-query loss and are trained by the outcome losses alone (Appendix A.2).

Learning from candidate outcomes. Outcome labels y_i supervise $H(z_i + e)$:

$$\mathcal{L}_{\text{out}} = \mathcal{L}_{\text{score}} + \lambda_{\text{bank}} \mathcal{L}_{\text{bank}} = \sum_i w_i \, \ell_{\text{out}}\big(H(z_i + e), y_i\big), \tag{5}$$

where i covers generated and bank candidates, ℓ_{out} uses each set's loss components, and w_i includes averaging and bank weighting (Section 3.4). Outcome supervision thus reaches every candidate state, including those without visual targets, and requires distinctions in safety, progress, and comfort. A scene supplies outcome labels for up to 80 candidates and a future target for one.

Why future supervision still matters. Models that fit the available outcome labels may still disagree on other candidates (Grimm et al., 2020), including which one to select (Mandi et al., 2022). We bound this variation with a local generalized Gauss–Newton model around a fitted θ_0, keeping each predictor call in its own context.

Proposition 1 (Future supervision and decision stability) *Let $\mu, \lambda_{\text{fut}}, \epsilon > 0$, and let the executed-query readout be differentiable and nonzero at θ_0. Let G_{out} (with a trust-region term μI) and G_{fut} be the generalized Gauss–Newton curvatures of the BCE outcome and cosine future losses at θ_0, and let $\rho_{ij}(G)$ be the largest first-order change $|c_{ij}^\top \delta|$ over $\delta^\top G \delta \le \epsilon^2$. Then*

$$\rho_{ij}^{\text{out+fut}} = \rho_{ij}(G_{\text{out}} + \lambda_{\text{fut}} G_{\text{fut}}) \le \rho_{ij}(G_{\text{out}}) = \rho_{ij}^{\text{out}}, \tag{6}$$

strictly if and only if $J_{\text{fut}} G_{\text{out}}^{-1} c_{ij} \neq 0$, where $c_{ij} = \nabla_\theta(\hat{J}_i - \hat{J}_j)$ and J_{fut} is the Jacobian of the normalized readout. If $\hat{J}_i - \hat{J}_j > \rho_{ij}^{\text{out+fut}}$, the linearized ordering $i \succ j$ holds throughout.

Figure 3 shows how future constraints can remove local disagreement on candidate choice. A smaller variation range makes a candidate's lead harder to reverse within the allowed neighborhood. The future loss does not constrain every unexecuted candidate uniformly: it narrows a comparison only along directions that shared parameters couple to the executed query's readout. Appendix C.4 proves this local result; Section 4.3 evaluates the learning effects predicted by this mechanism.

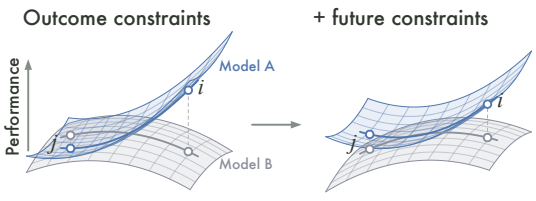

Figure 3: **Complementary supervision.** Two score surfaces fit the outcome labels but favor different candidates. In this example, future constraints remove that disagreement while allowing different predictions elsewhere.

3.4 Outcome Supervision across Candidates

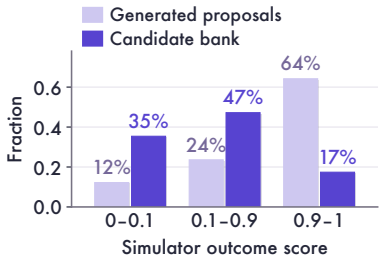

Figure 4: **Candidate bank.** The bank adds low-scoring plans.

Imitation training yields mostly expert-like, high-scoring candidates (light bars in Figure 4), so adverse outcomes such as collisions and drivable-area violations form a small share of the training examples.
Following CLOVER (Ang et al., 2026), we use a scene-matched bank of perturbed trajectories with precomputed simulator labels. The bank covers a broader score range, including unsafe plans (dark bars). We sample 16 bank trajectories per step to complement the generated candidates. Both sets use separate calls to the same ϕ, P, and H. For generated candidates, $\mathcal{L}_{\text{score}}$ sums binary cross-entropy over six components, using soft targets for progress. For bank candidates, $\mathcal{L}_{\text{bank}}$ averages binary cross-entropy over five non-progress components and adds 0.1 times the progress L1 loss. Each component loss is averaged over scenes and candidates in its set.

Winner-take-all L1 imitation, $\mathcal{L}_{\text{WTA}}$, supervises the candidate closest to the logged trajectory. Joint training combines all four losses:

$$\mathcal{L} = \mathcal{L}_{\text{WTA}} + \mathcal{L}_{\text{score}} + \lambda_{\text{fut}}\mathcal{L}_{\text{fut}} + \lambda_{\text{bank}}\mathcal{L}_{\text{bank}}, \tag{7}$$

with $\lambda_{\text{fut}} = 1$ and $\lambda_{\text{bank}} = 0.5$. We down-weight the bank loss to limit its influence on shared features while retaining supervision from trajectories outside the generated candidate set.

3.5 Inertial Re-Ranking

We observe abrupt changes between plans selected in adjacent frames. Framewise scoring can favor a high-scoring trajectory that is incompatible with the previous choice; similar candidate scores can also make selection sensitive to small changes. We therefore check each candidate against the previous plan π_{t-1} and use this compatibility to re-rank the current candidates.

For NAVSIM-v2, we instantiate the check with the benchmark's two-frame extended-comfort (EC) criterion (Cao et al., 2025). A deterministic linear-quadratic-regulator (LQR) controller tracks the previous plan and each current candidate with a kinematic bicycle model, producing ego-state rollouts. After aligning the two rollouts on their common horizon, we compare their acceleration, jerk, yaw-rate, and yaw-acceleration profiles. The candidate is marked compatible when all four RMS differences fall below their respective benchmark thresholds, giving the binary flag

$$c_{t,i} = c_{\text{EC}}(\pi_{t-1}, \tau_{t,i}) \in \{0, 1\}.$$

Logged human plans pass this check on 90.1% of adjacent frame pairs (Appendix E.1).

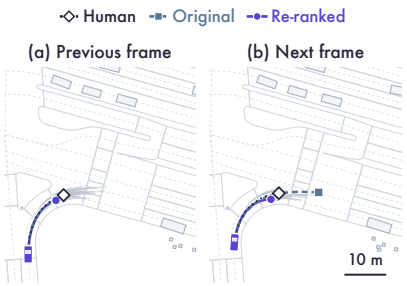

Figure 5: **Consecutive plans through a right turn.** Frames are 0.5 s apart. In (b), re-ranking selects a plan closer to the preceding choice and the human trajectory. Both views share map coordinates.

The check uses only the previous plan, current candidates, and controller rollouts, so it requires no future observations or labels at deployment. Within each log, we process frames chronologically and carry the selected plan forward. The first frame uses the learned score alone.

We add the compatibility result as a log-domain penalty to the learned utility:

$$\Psi_{t,i} := \log(0.01 + 0.99\, c_{t,i}), \qquad i_t^\star \in \arg\max_{1 \le i \le N} \left[\hat{J}_{t,i} + \lambda_{\text{state}}\Psi_{t,i}\right]. \tag{8}$$

With $\lambda_{\text{state}} = 1$, compatible candidates receive no penalty and retain their learned-score ordering; incompatible candidates receive $\Psi_{t,i} = -\log 100$. Re-ranking adds no learned parameters or training loss. We report both framewise and re-ranked results on NAVSIM-v2.

Method	Venue	Input	NC^2	DAC^2	TLC	DDC	EP^1	EP^2	TTC^1	TTC^2	C	LK	HC	EC	PDMS	EPDMS
End-to-end																
TransFuser	TPAMI'23	C+L	97.8	92.1	99.8	99.2	79.2	87.5	92.8	96.6	100.0	95.7	98.3	88.7	84.0	83.9
DiffusionDrive	CVPR'25	C+L	98.2	96.2	99.8	99.5	82.2	87.4	94.7	97.3	100.0	97.0	98.3	87.4	88.1	88.2
DiffusionDrive-V2	arXiv'25	C	97.7	96.6	99.8	99.2	87.5	88.9	94.8	97.2	99.9	96.0	97.8	91.0	91.2	87.5
DAWN	arXiv'26	C	97.3	92.0	99.7	99.1	84.3	87.4	96.0	96.6	100.0	96.0	98.3	85.5	89.1	83.2
LWDrive	arXiv'26	C	98.8	98.4	99.7	99.0	87.3	90.3	96.2	98.6	99.8	96.3	97.9	73.3	92.0	89.6
UniTeD	ECCV'26	C	99.1	97.2	99.9	99.6	84.1	86.9	96.6	98.5	100.0	98.4	99.9	87.3	90.2	90.1
DrivoR	CVPR'26	C	99.3	99.2	99.8	99.7	89.9	87.4	96.7	98.8	100.0	95.8	98.0	75.5	<u>93.7</u>	90.7
Vision-language-action																
ReCogDrive	ICLR'26	C	98.3	97.7	99.8	98.4	87.3	89.9	94.9	97.4	100.0	90.9	97.8	27.6	90.8	82.7
DriveVLA-W0	ICLR'26	C	98.5	99.1	99.7	98.0	83.3	86.4	95.3	98.1	99.3	93.2	97.9	58.9	90.2	86.1
DriveWorld-VLA	ICML'26	C	98.6	99.1	99.8	99.6	85.9	87.4	96.1	97.9	100.0	97.0	97.8	78.6	91.3	86.8
ExploreVLA	ECCV'26	C	98.8	96.2	99.8	99.6	83.5	87.1	96.5	98.2	99.9	97.8	98.3	86.8	90.4	88.8
WAM-Flow	CVPR'26	C	99.5	98.4	99.9	99.4	82.3	84.6	97.0	99.3	99.7	96.9	96.9	33.8	90.3	84.9
World-model augmented																
WoTE	ICCV'25	C+L	98.5	96.8	99.8	98.8	81.9	86.1	94.9	97.9	99.9	95.5	98.3	82.9	88.3	87.6
WorldRFT	AAAI'26	C	97.8	96.5	99.8	99.5	81.7	88.5	94.0	97.0	100.0	97.4	98.1	69.1	87.8	86.7
GraphWorld	arXiv'26	C	98.4	98.8	99.1	99.1	83.2	85.9	95.5	97.9	100.0	96.0	97.8	74.6	90.1	89.5
Discrete-WAM	arXiv'26	C	98.5	98.2	99.8	99.7	88.7	90.5	95.3	97.9	100.0	97.2	98.3	78.1	92.2	90.4
DriveFuture	arXiv'26	C	98.8	99.1	99.9	99.6	84.2	86.6	95.4	98.4	100.0	96.4	98.3	74.8	90.7	89.9
SafeDrive	CVPR'26	C+L	99.5	99.0	99.9	99.7	84.3	88.6	97.2	98.9	100.0	97.5	98.2	81.9	91.6	87.5
Drive-JEPA	arXiv'26	C	98.4	98.6	99.8	99.7	91.4	91.4	96.2	97.8	99.9	97.6	97.9	76.9	<u>93.7</u>	90.8
Ours	This work	C	99.3	99.2	99.6	99.7	91.3	90.0	96.0	98.8	100.0	96.1	97.8	77.0	**94.0**	<u>91.4</u>
+ inertial re-ranking		C	99.3	99.2	99.6	99.7	–	89.8	–	98.7	–	96.2	97.8	92.8	–	**93.0**

Table 1: **Planning performance on NAVSIM-v1 and NAVSIM-v2 (%).** Input C: cameras, L: LiDAR; superscripts 1 and 2 denote native NAVSIM-v1 and NAVSIM-v2. Best and second-best results are bold and underlined. We evaluate with the latest official NAVSIM devkits.

(a) Auxiliary target

Aux.	Realized	PDMS
✗	✗	93.01
✓	✗	92.70$_{-0.31}$
✓	✓	93.96$_{+0.95}$

(b) Outcome-label pairing

Permuted	Sel.	Order
none	0.728	64.2%
candidate bank	0.647	55.6%
all	0.461	51.5%

(c) Inertial re-ranking

Score	off	on
EC	77.0	92.8$_{+15.8}$
EP	90.0	89.8$_{-0.2}$
EPDMS	91.4	93.0$_{+1.6}$

Table 2: **Ablations of future targets, outcome supervision, and re-ranking.** (a) Native-V1 PDMS with no, current-frame, or realized-future targets. (b) Outcome-label permutation during training, tested on 511 held-out scenes. *Sel.*: selected PDM (0–1; random pick: 0.524). *Order*: pairwise ordering accuracy. (c) Native-V2 re-ranking with fixed weights and candidates. EC: extended comfort; EP: ego progress. Subscripts give changes from row 1 in (a) and off in (c).

4 Experiments

We organize the experiments around four research questions:

- **RQ1.** *Does World4Scorer improve driving performance on NAVSIM and Bench2Drive?*
- **RQ2.** *What do future targets, candidate outcomes, and inertial re-ranking contribute?*
- **RQ3.** *Do candidate-specific predictive states guide the model's trajectory selection?*
- **RQ4.** *Can outcome-based scoring improve manipulation with a frozen world model?*

4.1 Setup

We evaluate on all 12,146 NAVSIM test scenes using the NAVSIM-v1 PDM score (PDMS) (Dauner et al., 2024) and NAVSIM-v2 extended PDMS (EPDMS) (Cao et al., 2025), defined in Appendix B.1. Both metrics score the selected trajectory in the official simulator; the learned utility only ranks candidates. World4Scorer uses DrivoR's (Kirby et al., 2026) training split, LoRA-adapted DINOv2 encoder, generator, and candidate count; we train for 25 epochs with AdamW, a peak learning rate of 2×10^{-4}, and a global batch size of 64. Linear warmup covers the first 10% of training, then cosine decay. We select the main checkpoint on the validation split.

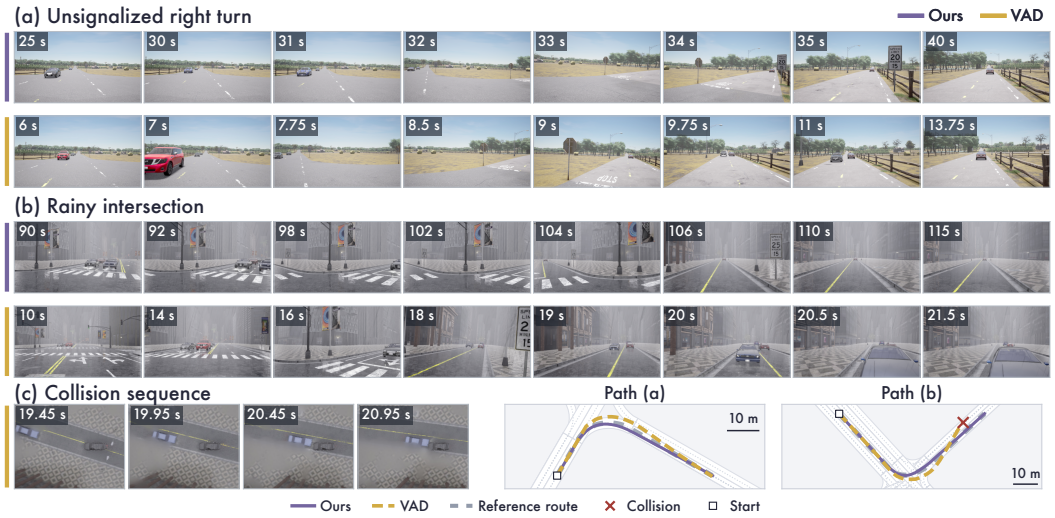

Figure 6: **Closed-loop trajectories on Bench2Drive.** Purple denotes Ours; yellow denotes VAD (Jiang et al., 2023). Eight frames per method show (a) a right turn where VAD leaves its route lane and (b) a rainy intersection where VAD collides with an oncoming vehicle. Ours completes both routes. Panel (c) enlarges the collision sequence; maps show the full executed paths and recorded collision location.

4.2 • Driving Performance (RQ1)

Planning on NAVSIM. World4Scorer achieves state-of-the-art NAVSIM-v2 performance with 93.0 EPDMS and obtains 94.0 PDMS on NAVSIM-v1, the best results in Table 1. Without re-ranking, it already reaches 91.4 EPDMS, 0.7 points above DrivoR's 90.7 on the same evaluation. Re-ranking adds 1.6 EPDMS points through previous-plan compatibility. The comparison evaluates complete planners with their own generators; Section 4.4 tests scoring with fixed candidate pools.

Closed-loop driving on Bench2Drive. After training on the official 1000-clip subset, our adapted planner achieves 73.27 Driving Score and 46.36% success over 220 Bench2Drive (Jia et al., 2024) routes (Table 3). The adaptation adds a route-point input and uses simulator ego state, retained turn commands, and route-based re-ranking (Appendix E.5). The table compares this complete system with published reference results; their training data and input interfaces are not matched to ours. Figure 6 compares two recorded maneuvers with VAD (Jiang et al., 2023). In (a), VAD completes the right turn but leaves its route lane, scoring 77.77. In (b), it enters the opposing lane after the turn, collides with an oncoming vehicle, and becomes blocked, scoring 43.93. Ours completes both routes without lane departures or collisions, achieving a Driving Score of 100 in each case.

Method	Score ↑
VAD (Jiang et al., 2023)	42.35
WoTE (Li et al., 2025b)	61.71
ThinkTwice (Jia et al., 2023b)	62.44
DriveAdapter (Jia et al., 2023a)	64.22
Drive-JEPA (Wang et al., 2026)	64.52
SafeDrive (Kim et al., 2026)	<u>66.77</u>
Ours	**73.27**

Table 3: **Bench2Drive results.** Driving Score.

4.3 ■ Supervision and Re-Ranking (RQ2)

Realized-future targets. Anchoring the executed state to its observed future raises PDMS from 93.01 to 93.96, whereas the same objective with a current-frame target gives 92.70 (Table 2a). The three runs share the architecture, seed, bank supervision, and last-epoch evaluation; this single-seed gap is consistent with, but does not by itself isolate, the targets' temporal content.

Candidate–outcome pairing. Shuffling bank outcome labels reduces ordering accuracy from 64.2% to 55.6%; shuffling all labels gives 51.5% (Table 2b). Selected PDM falls from 0.728 to 0.647 and 0.461. Each permutation preserves the scene's trajectories and label distribution but breaks their correspondence. All three conditions train the same architecture on 512 scenes for 60 epochs and evaluate on 511 scenes from disjoint logs. These controls show that ranking requires the correct correspondence between trajectories and outcomes.



Figure 7: **Candidate-specific states guide trajectory selection.** Twelve selected scenes where *Ours* attains pool-best native-V1 PDMS. Purple and amber show Ours and the logged human trajectory; gray marks state interventions. Faint lines show the same 64-candidate pool used in all three conditions.

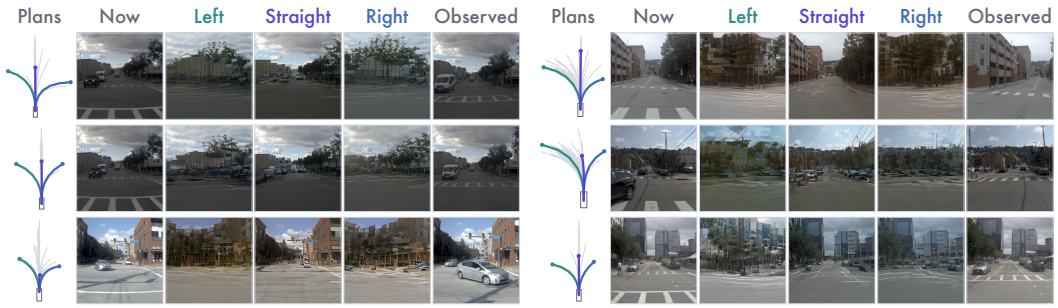

Figure 8: **Visual readouts for alternative trajectory queries.** Colors match left-turn, near-straight (selected), and right-turn candidates to their +2 s readouts from a separately trained network. Only the logged trajectory has an observed future; query images are qualitative readouts, not tests of counterfactual accuracy.

Inertial re-ranking. With weights and candidates fixed, inertial re-ranking raises extended comfort from 77.0 to 92.8 while progress changes only from 90.0 to 89.8, yielding 1.6 EPDMS points (Table 2c). In the right turn of Figure 5, it favors trajectories compatible with the preceding choice.

4.4 ♦ Trajectory-Selection Analysis (RQ3)

Matching states to candidates. We test whether selection depends on the state predicted for each trajectory. The model, scene features, and 64 candidates per scene stay fixed; we change only the assignment of states to candidates. Every evaluation covers all 12,146 NAVSIM-v1 test scenes.

Assigning each trajectory another candidate's state reduces PDMS from 94.19 to 79.73, averaged over 100 random shuffles (Table 4). Replacing all states with their scene mean yields 71.10. Equal scores then leave the choice to a fixed candidate index.

Candidate state	PDMS ↑	Drop ↓
Own state	**94.19**	–
Shuffled states	79.73	14.47
Scene-mean state	71.10	23.09

Table 4: **Candidate states affect selection.** All 12,146 test scenes with DrivoR's 64 candidates, not the Table 1 pool. Shuffled PDMS averages 100 runs; drops are relative to Own state.

Every one of the 100 shuffles lowers PDMS. Shuffling preserves each scene's score distribution but breaks the match between trajectories and states, changing which candidate receives each score.

Figure 7 illustrates twelve scenes where original states select pool-best plans and reassignment changes the choice. Successful plans can differ from the logged driver because the evaluator scores safety, progress, and comfort along each candidate.

Visualizing candidate states. Figure 8 compares left-turn, near-straight, and right-turn queries in each scene. With the driving model frozen, we train a separate image reconstruction network on

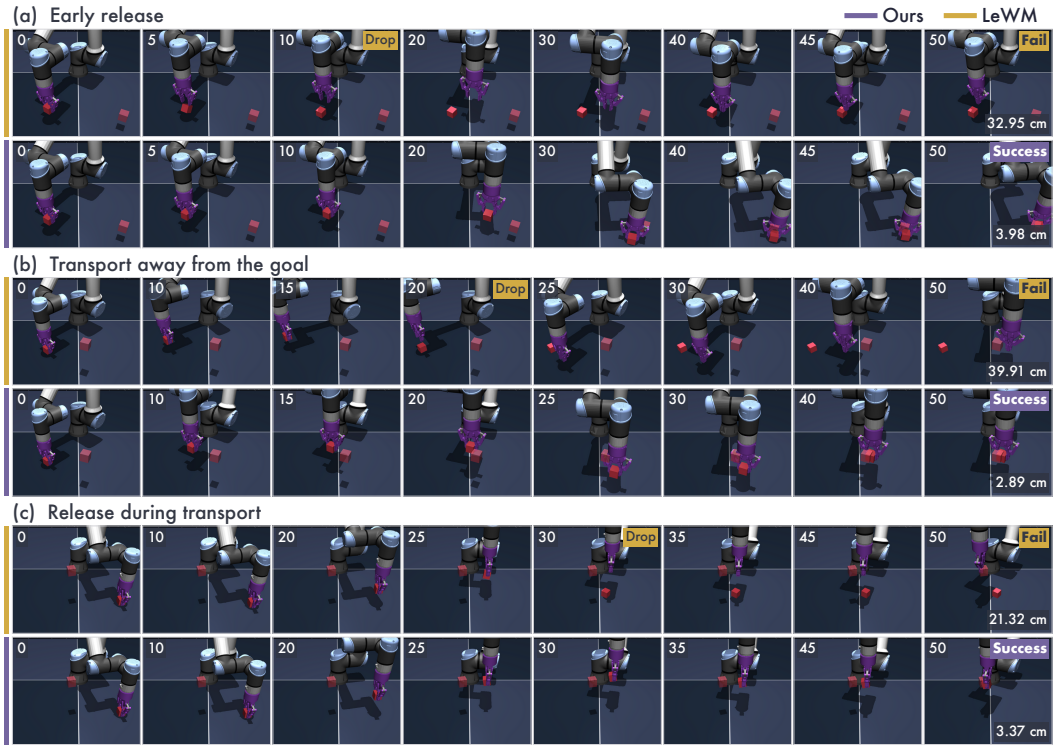

Figure 9: **Outcome-based scoring corrects unsuccessful LeWM plans.** For each case, eight frames trace each method from the same start toward the same goal. Numbers mark steps; *Drop* marks release. Badges give outcomes and final distances to the translucent goal cube (success: within 4 cm).

6,000 logged current/future pairs. It combines predicted pooled features with the current spatial grid and a frozen RAE (Zheng et al., 2026) decoder. Query changes alter the decoded viewpoint. These images visualize query-conditioned representations; only the logged query has an observed future.

4.5 ▲ Transfer to Manipulation (RQ4)

On OGBench-Cube (Park et al., 2025), adding outcome-based scoring to LeWM (Maes et al., 2026) raises success from 68.91% to 73.64% over 11 evaluation seeds (Table 5; paired gain +4.73 points, 95% CI [+2.18, +7.27]). We keep LeWM's official implementation, world model, and planning budget fixed; our scorer evaluates action sequences from predicted states and goals and adds its learned outcome score to LeWM's planning score.

Each seed runs 50 paired episodes sharing initial states and goals (Appendix D.2). At LeWM's released evaluation seed 42, success rises from 72% to 80%. In Figure 9, LeWM ends 21–40 cm from the goal and ours succeeds.

Method	Success (%)
GCIQL (Kostrikov et al., 2022)	64
GCIVL (Park et al., 2025)	56
PLDM (Sobal et al., 2025)	65
LeWM (Maes et al., 2026)	68.91
Ours	**73.64**

Table 5: **OGBench-Cube results.** Upper rows are reported by LeWM; lower rows average our 11 seeds of 50 paired episodes.

5 Conclusion

World4Scorer scores candidates with a trajectory-conditioned predictor trained by simulator outcomes and anchored by the observed future. Controlled experiments show the value of correctly paired outcome labels and realized-future targets, and state interventions show that selection depends on candidate-specific states. The method achieves state-of-the-art NAVSIM-v2 performance and strong closed-loop driving results, and outcome-based scoring also improves manipulation with a frozen world model. These findings support grounding predictive states in decision outcomes.

Reproducibility Statement

All experiments use public benchmarks and their official evaluators (NAVSIM-v1 and NAVSIM-v2, Bench2Drive, and OGBench-Cube). Appendix A.2 and Table 7 give the full training configuration, including the data split, losses and weights, optimizer, schedule, batch size, and checkpoint selection; Appendix A.3 reports model size, compute, and latency. Appendix B.1 specifies how the simulator labels used for training are computed, and Section 3.5 with Appendix A.2 specifies inertial re-ranking, including the controller and thresholds. The fixed-pool interventions (Appendix D.1) state their permutation seeds and bootstrap procedures, and Appendices D.2 and E.5 describe the OGBench-Cube and Bench2Drive protocols. We will release the code upon publication.

Use of AI

Large language models assisted with prose editing, LaTeX, figure code, analysis scripts, and cluster tooling. The authors determined the scientific claims, experimental design, controls, and interpretation of the results. Reported results come from experiments and cited studies. The authors verified the manuscript and take full responsibility for its content.

References

Sining Ang, Yuguang Yang, Canyu Chen, and Yan Wang. CLOVER: Closed-loop value estimation and ranking for end-to-end autonomous driving planning. *arXiv preprint arXiv:2605.15120*, 2026.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024.

Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-simulation for autonomous driving. In *Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 4709–4722, 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, 2023. doi: 10.1109/TPAMI.2022.3200245.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations*, 2024.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 37, 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. In *International Conference on Learning Representations*, 2026.

Tanner Fiez, Lalit Jain, Kevin Jamieson, and Lillian Ratliff. Sequential experimental design for transductive linear bandits. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Christopher Grimm, André Barreto, Satinder P. Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17853–17862, 2023.

Feiyang Jia, Lin Liu, Ziying Song, Caiyan Jia, Hangjun Ye, Xiaoshuai Hao, and Long Chen. DriveWorld-VLA: Unified latent-space world modeling with vision-language-action for autonomous driving. In *International Conference on Machine Learning*, 2026.

Xiaosong Jia, Yulu Gao, Li Chen, Junchi Yan, Patrick Langechuan Liu, and Hongyang Li. DriveAdapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *IEEE/CVF International Conference on Computer Vision*, 2023a.

Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023b.

Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2Drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 37, pp. 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 *IEEE/CVF International Conference on Computer Vision*, pp. 8340–8350, 2023.

Bo Jiang, Shaoyu Chen, Hao Gao, Bencheng Liao, Qian Zhang, Wenyu Liu, and Xinggang Wang. VADv2: End-to-end vectorized autonomous driving via probabilistic planning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2402.13243.

Jungho Kim, Jiyong Oh, Seunghoon Yu, Hongjae Shin, Donghyuk Kwak, and Jun Won Choi. Safedrive: Fine-grained safety reasoning for end-to-end driving in a sparse world. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24854–24864, 2026. arXiv:2602.18887.

Ellington Kirby, Alexandre Boulch, Yihong Xu, Yuan Yin, Gilles Puy, Éloi Zablocki, Andrei Bursuc, Spyros Gidaris, Renaud Marlet, Florent Bartoccioni, Anh-Quan Cao, Nermin Samet, Tuan-Hung Vu, and Matthieu Cord. Driving on registers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 32058–32069, 2026. arXiv:2601.05083.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

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

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

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, Lu Hou, Lue Fan, and Zhaoxiang Zhang. Drivevla-w0: World models amplify data scaling law in autonomous driving. In *International Conference on Learning Representations*, 2026a. arXiv:2510.12796.

Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, Kun Ma, Guang Chen, Hangjun Ye, Wenyu Liu, and Xinggang Wang. ReCogDrive: A reinforced cognitive framework for end-to-end autonomous driving. In *International Conference on Learning Representations*, 2026b. arXiv:2506.08052.

Zhenxin Li, Wenhao Yao, Zi Wang, Xinglong Sun, Jingde Chen, Nadine Chang, Maying Shen, Jingyu Song, Zuxuan Wu, Shiyi Lan, and Jose M. Alvarez. ZTRS: Zero-imitation end-to-end autonomous driving with trajectory scoring. *arXiv preprint arXiv:2510.24108v1*, 2025c.

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.

Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and Xinggang Wang. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12037–12047, 2025. doi: 10.1109/CVPR52734.2025.01124.

Hongbo Lu, Liang Yao, Chenghao He, Haoyu Wang, Xiang Gu, Xianfei Li, Wenlong Liao, Tao He, and Pai Peng. The DAWN of world-action interactive models. *arXiv preprint arXiv:2605.11550*, 2026.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorld-Model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Jayanta Mandi, Víctor Bucarey, Maxime Mulamba Ke Tchomba, and Tias Guns. Decision-focused learning: Through the lens of learning to rank. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14935–14947, 2022.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud 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. *Transactions on Machine Learning Research*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025.

Zihao Sheng, Xin Ye, Jingru Luo, Sikai Chen, and Liu Ren. ExploreVLA: Dense world modeling and exploration for end-to-end autonomous driving. In *European Conference on Computer Vision*, 2026.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems*, 2025.

Ziying Song, Caiyan Jia, Lin Liu, Lei Yang, Shengkai Zhang, Feiyang Jia, Fengda Zhao, Peiliang Wu, Shaoqing Xu, Chen Lv, and Yadan Luo. GraphWorld: Long-horizon planning with world models for end-to-end autonomous driving. *arXiv preprint arXiv:2606.16274*, 2026.

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.

Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *Advances in Neural Information Processing Systems*, volume 35, pp. 6119–6132, 2022.

Tianze Xia, Yongkang Li, Lijun Zhou, Jingfeng Yao, Kaixin Xiong, Haiyang Sun, Bing Wang, Kun Ma, Guang Chen, Hangjun Ye, Wenyu Liu, and Xinggang Wang. Drivelaw: Unifying planning and video generation in a latent driving world. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 39701–39712, 2026. arXiv:2512.23421.

Yifang Xu, Jiahao Cui, Zhihao Zhu, Hanlin Shang, Shan Luan, Mingwang Xu, Feipeng Cai, Neng Zhang, Yaoyi Li, Jia Cai, and Siyu Zhu. WAM-flow: Parallel coarse-to-fine motion planning via discrete flow matching for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24918–24928, 2026. arXiv:2512.06112.

Chen Yang, Yuhao Wei, Ze Xu, Ziheng Zou, Shuang Liang, Delin Ouyang, Lingfeng Qi, Jie Li, and Guofa Li. Lwdrive: Layer-wise world-model-guided vision-language model planning for autonomous driving. *arXiv preprint arXiv:2606.29879*, 2026a.

Pengxuan Yang, Ben Lu, Zhongpu Xia, Chao Han, Yinfeng Gao, Teng Zhang, Kun Zhan, Xianpeng Lang, Yupeng Zheng, and Qichao Zhang. WorldRFT: Latent world model planning with reinforcement fine-tuning for autonomous driving. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(14):11649–11657, 2026b. doi: 10.1609/aaai.v40i14.38149.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(14):11910–11918, 2026a. doi: 10.1609/aaai.v40i14.38178.

Ziyang Yao, Haochen Liu, Yuncheng Jiang, Zeyu Zhu, Zibin Guo, Jingru Wang, Tianle Liu, Jianwei Cui, Kuiyuan Yang, Hongwei Xie, Jingwei Zhao, Guang Chen, and Hangjun Ye. DiscreteWAM: Unified discrete vision-action token editing for world-policy learning. *arXiv preprint arXiv:2606.05645*, 2026b.

Bo Zhao, Xinting Zhao, Naifan Li, Erkang Cheng, and Haibin Ling. UniTeD: Unified temporal diffusion for joint perception and planning in autonomous driving. In *European Conference on Computer Vision*, 2026.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, 2026.

Yupeng Zheng, Pengxuan Yang, Zebin Xing, Qichao Zhang, Yuhang Zheng, Yinfeng Gao, Pengfei Li, Teng Zhang, Zhongpu Xia, Peng Jia, Xianpeng Lang, and Dongbin Zhao. World4drive: End-to-end autonomous driving via intention-aware physical latent world model. In *IEEE/CVF International Conference on Computer Vision*, pp. 28632–28642, 2025.

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.

APPENDIX

This appendix provides implementation details and further analyses of prediction and planning.

- Appendix A: notation, training objectives, implementation, and computational cost.
- Appendix B: simulator labels and differences between NAVSIM scoring rules.
- Appendix C: visual targets, image readouts, and candidate-dependent features.
- Appendix D: state reassignment and transfer to manipulation.
- Appendix E: consecutive selections, history controls, and Bench2Drive evaluation.

A SETUP AND REFERENCE

We define the notation and document World4Scorer's training setup and computational cost.

A.1 NOTATION REFERENCE

Symbol	Meaning
t, i, j, N	Decision time, candidate indices, and candidate count ($N = 64$).
n, M	Training-scene index and scene count in the future loss.
o_t, $o_{t+2\mathrm{s}}$	Current observation and the observed future two seconds later.
s_t, e_t	Camera scene tokens and the encoded ego state.
$\tau_{t,i}$, $\mathcal{C}_t$, τ_t^{exec}	Candidate trajectory, candidate set, and logged executed trajectory.
ϕ, P, g, H	Action encoder, shared predictor, visual readout, and score heads.
$q_{t,i}$, Q_t	Candidate query and the collection of candidate queries.
$z_{t,i}$, Z_t, z_t^{exec}	Candidate state, candidate-state matrix, and executed-query state.
f_{DINO}, u_t^{fut}	Frozen visual encoder and its observed future embedding.
sg, σ	Stop-gradient operator and sigmoid function.
$\ell_{t,i,m}$, $p_{t,i,m}$, y_m	Component logit, sigmoid prediction, and outcome target.
$\mathcal{G}$, $\mathcal{A}$, $\mathcal{M}$	Multiplicative and additive score-component sets; $\mathcal{M} = \mathcal{G} \cup \mathcal{A}$.
$\hat{J}_{t,i}$, α_m, β_m	Log-domain ranking score and its fixed component weights.
$\mathcal{L}_{\mathrm{WTA}}$, $\mathcal{L}_{\mathrm{score}}$	Imitation loss and generated-candidate outcome loss.
$\mathcal{L}_{\mathrm{fut}}$, $\mathcal{L}_{\mathrm{bank}}$	Observed-future loss and candidate-bank outcome loss.
$\mathcal{L}_{\mathrm{out}}$, ℓ_{out}, w_i	Outcome objective, per-candidate loss, and averaging/weighting factor.
λ_{fut}, λ_{bank}	Future and bank loss weights (1 and 0.5).
π_{t-1}, $c_{t,i}$, c_{EC}	Previous selected plan, binary compatibility result, and the EC check.
$\Psi_{t,i}$, λ_{state}, $i_t^\star$	Re-ranking penalty, its weight (1), and the selected index.
θ_0, δ	Fitted parameters of ϕ, P, g, H and a local perturbation.
A_F, A_O	Weighted Jacobians of normalized visual and outcome predictions.
G_O, G	Local constraint matrices before and after adding visual constraints.
$\hat{\Delta}_{ij}$, c_{ij}	Candidate score difference and its parameter gradient at θ_0.
μ, λ, ϵ	Analysis scale, visual-constraint weight, and perturbation budget.
$\rho_{ij}(G)$	Maximal first-order score-difference change; $\rho_{ij}^{\mathrm{out}} = \rho_{ij}(G_O)$, $\rho_{ij}^{\mathrm{out+fut}} = \rho_{ij}(G)$.
$\widehat{u}_{t,i}$, X_t, $\widehat{X}_{t+2\mathrm{s},i}$	Predicted visual feature, current patch grid, and reconstructed future grid.
λ_{cube}	Weight of learned utility in the LeWM planner.
PDMS, EPDMS	Native NAVSIM-v1 and NAVSIM-v2 aggregate planning scores.

Table 6: **Notation used in the method and analyses.** Time indices are omitted when a scene is fixed. Section 3.2 defines the six predicted score components.

A.2 TRAINING AND STATE INTERVENTIONS

Trajectory and outcome losses. Generated, executed, and bank queries share the predictor P. The heads H score candidate states, while g reads the executed state for visual prediction. For generated candidates, $\mathcal{L}_{\mathrm{WTA}}$ sums absolute errors over coordinates, averages over poses, minimizes

over candidates, and then averages over scenes. We sum this loss for two targets: the logged four-second trajectory and the same trajectory time-warped to five seconds by cubic-spline resampling.

The outcome losses average over scene–candidate pairs, written $\langle\cdot\rangle_{\text{gen}}$ and $\langle\cdot\rangle_{\text{bank}}$:

$$\begin{aligned}\mathcal{L}_{\text{score}} &= \sum_{m\in\mathcal{M}} \langle b(\ell_m, y_m)\rangle_{\text{gen}},\\ \mathcal{L}_{\text{bank}} &= \frac{1}{5}\sum_{m\in\mathcal{M}\setminus\{\text{EP}\}} \langle b(\ell_m, y_m)\rangle_{\text{bank}} + 0.1\langle|\sigma(\ell_{\text{EP}}) - y_{\text{EP}}|\rangle_{\text{bank}}.\end{aligned} \tag{9}$$

Here $\mathcal{M} = \mathcal{G}\cup\mathcal{A} = \{\text{NC}, \text{DAC}, \text{DDC}, \text{TTC}, \text{EP}, \text{C}\}$, y_m is the outcome target for component m, and $b(\ell, y) = \log(1+\exp\ell) - y\ell$ is binary cross-entropy on a logit. Generated progress uses soft-label cross-entropy, whereas bank progress uses L1. NC and DDC targets equal to 0.5 are mapped to zero before these losses. The averages use the available labels and masks described below. Candidate coordinates are detached; the scoring losses update ϕ, P, H, and the scene features.

Visual supervision. Once per scene, the executed query predicts the frozen front-view embedding two seconds later. We minimize cosine distance with weight $\lambda_{\text{fut}} = 1$. Gradients update ϕ, P, g; current-scene tokens are detached on this auxiliary path.

With one query, self-attention has only one key and its softmax is constant. For a singleton input x and head dimension d,

$$\text{Attn}(x) = \text{softmax}\left(\frac{(W_Q x)^\top (W_K x)}{\sqrt{d}}\right) W_V x = W_V x, \qquad \nabla_{W_Q, W_K}\mathcal{L}_{\text{fut}} = 0. \tag{10}$$

Thus the self-attention query and key projections receive no visual-loss gradient: 524,288 of the 4,504,320 shared parameters (11.6%). They are trained by the multi-candidate outcome losses. Cross-attention over scene tokens is unaffected by this singleton property.

Candidate-bank supervision. Each step samples 16 extra trajectories from the matching scene, with bank loss weight $\lambda_{\text{bank}} = 0.5$. Each scene's bank holds 54.5 valid entries on average, so one step sees on average 30.8% of a scene's stored pool. There are no bank entries for 310 scenes; 10.7% of scenes lack enough trajectories in the required safety categories to fill the quota.

Validity masks exclude missing outcome labels from the losses. Bank time-to-collision labels are used without masking; generated-candidate labels use 2.0 as the invalid-target marker. All available components supervise the shared predictor used for candidate scoring at deployment.

Training configuration. Table 7 lists training settings. Caches store front-view embeddings two seconds ahead and scene-matched trajectories, labels, and validity masks.

Setting	Value
Data and batches	103,288 examples; four devices, 16 samples per device (64 total).
Optimization	AdamW, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01.
Schedule	25 epochs; peak learning rate 2×10^{-4}, linear warmup over the first 10% of steps, then cosine decay to zero.
Image input	672×1148 (height $\times$ width), ImageNet normalization; deterministic preprocessing without crop, flip, or color jitter.
Backbone	DINOv2 ViT-S/14 with rank-32 LoRA.
Predictor	Four decoder layers; feature width 256, feed-forward width 1024.
Random seed	3 for Python, NumPy, PyTorch, the distributed sampler, and data workers.

Table 7: **Training configuration.** All runs use the public DrivoR software stack.

Validation runs every epoch. The reported NAVSIM checkpoint is epoch 23, selected by validation score; the three target controls in Table 2a use the last epoch. Appendix E.5 gives the Bench2Drive setup. The test split is excluded from training, validation, and checkpoint selection.

Candidate-state interventions. We fix the epoch-23 checkpoint, scene features, candidate pool, score heads, and score aggregation. Scene-mean replacement assigns the mean state to every candidate in a scene. State shuffling uses a fixed-point-free permutation, so each candidate receives another candidate's state while all inputs remain fixed.

Inertial re-ranking and evaluation. Frames are processed chronologically within each log, and the selected trajectory is carried to the next frame. At the nominal 0.5 s frame interval, the 0.1 s LQR rollouts of the previous plan and each candidate are compared over their shared 3.5 s window. The first frame uses framewise selection. Ties are resolved by selecting the lowest candidate index.

Each selected trajectory is evaluated with the official benchmark aggregator; selection uses the learned utility in Equation 2. The ordered component weights $(\alpha_{\mathrm{NC}}, \alpha_{\mathrm{DAC}}, \alpha_{\mathrm{DDC}}, \beta_{\mathrm{TTC}}, \beta_{\mathrm{EP}}, \beta_{\mathrm{C}})$ are $(1, 1, 0, 5, 5, 2)$ for NAVSIM-v1 and $(10, 13, 6, 14, 15, 2.1)$ for NAVSIM-v2. Section 4.2 evaluates the epoch 23 checkpoint under native benchmark scoring. Section 4.3 toggles inertial re-ranking while holding this checkpoint and its candidate pool fixed.

Compute. The 25-epoch training run uses four 80 GB A800/A100 GPUs for roughly 40 hours. On one 24 GB A30, inference at batch 32 runs at 2.6 frames per second with 14 GiB peak memory and processes the full test split in about 1.3 hours. Software follows the public DrivoR stack.

A.3 Compute and Parameter Scale

World4Scorer uses the DrivoR sensor stack. Table 8 compares the parameter counts, computation, and latency of planners that score multiple candidates.

Compute convention. We follow the FVCore convention used by DrivoR (Kirby et al., 2026), counting one multiply–accumulate as one operation. Fused scaled dot-product attention is excluded from this count. World4Scorer uses 349.9 billion operations: 349.1 for the encoder and 0.74 for generation and scoring. Including attention raises the total to 927.1 billion; attention over 15,744 encoder tokens from four views accounts for almost all of the 577.2 billion added operations.

Scale. World4Scorer has 35.52M parameters, compared with 321M and 888M for the ViT-L and ViT-H systems. Visual encoding dominates runtime. Generation and scoring take 1.4 of 106.5 ms per pass and 0.08% of compute including attention.

Precision. The latency column uses fp32 with TF32 disabled. On the same card and input, the same graph runs in 75.6 ms with TF32 and 23.0 ms with fp16 (encoder: 20.2 ms). The $4.6\times$ speedup from fp32 to fp16 reflects precision alone. All reported planning scores use fp32.

Method	Image size	Cams	Encoder	Params	GFLOPs	Latency
RAP-Dino (Feng et al., 2026)	448×768	4	ViT-H	888M	4760	690
GTRS-Dense (Li et al., 2025d)	512×2048	1	ViT-L	321M	1730	400
ZTRS (Li et al., 2025c)	512×2048	2	V2-99	81M	840	193
GTRS-Aug (Li et al., 2025d)	512×2048	1	V2-99	171M	439	243
GTRS-Dense (Li et al., 2025d)	512×2048	1	V2-99	81M	404	96
DrivoR (Kirby et al., 2026)	672×1148	4	ViT-S	41M	351	110
Ours	672×1148	4	ViT-S	**35.52M**	**349.9**	**106.5**
attention counted					927.1	

Table 8: **Compute and parameter scale.** GFLOPs use the FVCore convention of one operation per multiply–accumulate and exclude fused attention; the *attention counted* row includes it. Latency is in milliseconds for one fp32 forward pass at batch one on an A100. World4Scorer's latency excludes inertial re-ranking (125.4 ms on CPU for all 64 candidates).

B Evaluation Protocols

We describe simulator labels and compare NAVSIM-v1 and NAVSIM-v2 objectives with the model and candidate pools fixed, isolating the scoring rule.

B.1 Simulator-Based Training Labels

We obtain candidate labels by tracking each trajectory in the simulator and measuring collision avoidance, progress, and other outcomes in three steps:

1. **Prepare the scene.** A model-independent cache stores the initial ego state, centerline, on-route lane IDs, drivable area, and the PDM planner's progress. It also stores 51 occupancy timesteps as STRtree-indexed polygons. Training and evaluation share this cache.
2. **Track the candidate.** Generated trajectories are detached before scoring, transformed to the global frame, and resampled to 40 poses at $0.1\,\mathrm{s}$. An LQR controller tracks each trajectory through a kinematic bicycle model, producing a 4-second ego-state sequence.
3. **Measure the outcomes.** Collision avoidance, drivable-area compliance, and driving direction use polygon queries against the cached maps. The tracked ego-state sequence supplies time-to-collision, centerline progress, and comfort labels.

In the simulator's aggregate score, the components combine in two groups. Collision avoidance and drivable-area compliance multiply together, and their product both scales raw progress and multiplies the aggregate score. For the v1 training-label scorer, progress, time-to-collision, and comfort enter a weighted average whose weights match the deployed NAVSIM-v1 tuple $(1, 1, 0, 5, 5, 2)$; driving-direction compliance is computed but has zero v1 weight.

Benchmark metrics. NAVSIM-v1 reports the PDM score (Dauner et al., 2024),

$$\mathrm{PDMS} = \mathrm{NC} \cdot \mathrm{DAC} \cdot \frac{5\,\mathrm{TTC} + 5\,\mathrm{EP} + 2\,\mathrm{C}}{12}.$$

NAVSIM-v2 reports the extended PDM score (Cao et al., 2025),

$$\mathrm{EPDMS} = \mathrm{NC} \cdot \mathrm{DAC} \cdot \mathrm{DDC} \cdot \mathrm{TLC} \cdot \frac{5\,\mathrm{TTC} + 5\,\mathrm{EP} + 2\,\mathrm{HC} + 2\,\mathrm{LK} + 2\,\mathrm{EC}}{16},$$

which adds DDC and traffic-light compliance (TLC) as multipliers and lane keeping (LK) and extended comfort (EC) to the weighted average; history comfort (HC) replaces comfort (C). EC compares plans of consecutive frames. A term is not penalized where the logged human trajectory also violates it. Both metrics use the navtest scenes with non-reactive background agents.

Progress normalization. Progress is normalized by a reference distance:

- **Training:** the maximum of the candidate's raw progress and the cached planner progress.
- **Evaluation:** the official scorer normalizes progress over the proposals passed to it, which are the PDM reference and the selected trajectory in our evaluation.

A training label depends only on its trajectory and scene, not on other candidates.

Selection and runtime. At selection time, the predicted components are combined using Equation 2. Collision avoidance, drivable-area compliance, and driving direction contribute weighted log probabilities; the remaining components contribute the log of their weighted sum. The NAVSIM-v1 weights $(1, 1, 0, 5, 5, 2)$ assign zero weight to driving direction.

Generated candidates change at every step, requiring online labels for all N candidates per sample. Computing these labels on CPU with NumPy and Shapely dominates training time.

B.2 Differences Between NAVSIM Evaluation Objectives

NAVSIM-v1 and NAVSIM-v2 select different candidates in 43% of scenes when applied to the same frozen candidate pools of our planner. We use simulator sub-scores to isolate the evaluation

rule from prediction error. The two utilities have within-scene rank correlation 0.69. Removing extended comfort raises the correlation to 0.96 and reduces argmax disagreement to 3%. The V1-optimal candidate fails the EC check in 99.7% of disagreement scenes. Selecting the V2-optimal candidate lowers native NAVSIM-v1 PDMS by 1.3 points.

An exact additive decomposition attributes 98% of the utility gap between the V1 and V2 choices to EC, 3% to driving direction (weighted only in V2), and -1% to progress, time-to-collision, and comfort together (progress alone: -0.7%). The V2-optimal candidate gains 50.4 EC points and loses 2.7 progress points relative to the V1-optimal candidate.

Progress equals 1.0 for 43% of candidates, so pool-wide percentiles from the sixtieth upward coincide. In 38% of scenes (57% in the most dynamic quarter), no candidate is both top-quartile in progress and EC-passing. The most dynamic quarter holds the scenes with the largest change between the current and $+2$ s front-view DINOv2 features. This share becomes 30% at the median threshold and 23% at a within-scene quartile. The within-scene progress–EC correlation is +0.07.

The recomputed V1 selections differ from the benchmark results only on 39 fp32 ties, changing mean PDMS by 6×10^{-7}. Recomputed EC matches the benchmark on all 10,040 adjacent pairs. The benchmark omits this term for the remaining 2,106 frames, which lack a valid adjacent pair.

C Predictive Representations

We examine visual targets and predictions, illustrate the content retained by pooled features, and connect learned candidate states to score differences.

C.1 Geometry of the Latent Prediction Target

We compare current and future DINOv2 features over 103,288 training frames. The current frame and its $+2.0$ s future are highly similar (mean cosine 0.94, fifth percentile 0.80), indicating that persistent scene content dominates the target and that a measurable temporal change remains. Both feature sets have effective rank 49 of 384 dimensions.

Pairing each current frame with a future from another scene reduces mean cosine to 0.649, close to the random-pairing baseline of 0.649; this shuffle removes the pairing but keeps the target features. True and shuffled pairs are readily distinguished (AUC 0.981, Cohen's d 2.84). Same-log pairs are also less similar (0.76), so the true-pair advantage is scene-specific.

C.2 Visualizing Predicted Features

The visual readout predicts a 384-dimensional front-view feature 2.0 s ahead, mean-pooled from frozen DINOv2 (Oquab et al., 2024) ViT-S/14 with registers (Darcet et al., 2024). The logged trajectory supplies the only observed future target; alternative queries share the predictor.

To visualize these features, we train two image reconstruction networks, CondGrid and conditional latent diffusion, with the driving model frozen (Figure 10). Each network combines the predicted feature with the current front view's 16×16 DINOv2 patch grid, which supplies spatial detail. This reconstruction stage is separate from the visual readout g and is used only for visualization. The alternative-query images below are qualitative readouts; no observed future exists for these queries.

The RAE decoder (Zheng et al., 2026) renders images from patch grids. Write X_t for the current patch grid and $\widehat{u}_{t,i} = g(z_{t,i})$ for a candidate's predicted visual feature. CondGrid, a small conditional transformer with parameters η, predicts a residual grid with classifier-free guidance at scale 2:

$$\Delta_{\text{cfg}} = \Delta_{\varnothing}(X_t) + 2[\Delta_{\eta}(X_t, \widehat{u}_{t,i}) - \Delta_{\varnothing}(X_t)]\,,$$
$$\widehat{X}_{t+2\text{s},i} = X_t + \Delta_{\text{cfg}},$$

where $\Delta_{\varnothing}$ omits the visual feature. The frozen RAE ViT-XL decoder renders $\widehat{X}_{t+2\text{s},i}$. Conditional latent diffusion uses cosine-schedule DDPM v-prediction on noisy future grids. Inference uses 20-step DDIM (guidance 1, seed 17) and the RAE decoder.
Both reconstruction networks use the same 6,000 navtrain current/future pairs. Inputs are the observed current grid and future pooled vector; the target is the full future grid. At test time, the pooled vector is replaced by $\widehat{u}_{t,i}$, normalized to the global norm of the cached DINOv2 targets. The driving model, DINOv2, and the RAE decoder stay fixed.

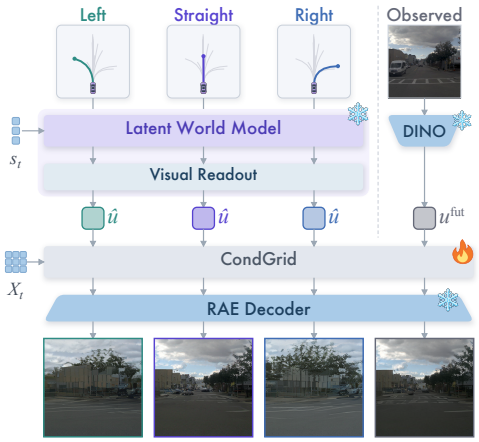

Figure 10: **From query to image.** Scene, ego state and candidates are fixed as the query changes. Each state z yields a visual feature $g(z)$, which the reconstruction network renders through a frozen decoder. Left, Straight (selected), and Right match Figure 8. The right column decodes the observed future's pooled feature.

CondGrid uses token-cosine, MSE, and weighted change-direction losses; diffusion uses a velocity loss. Both use scale perturbations, isotropic direction noise, and 10% vector dropout.

Changing the queried trajectory. Figure 8 fixes six junctions and compares left-turn, near-straight (model-selected), and right-turn candidates from each 64-candidate pool. The trajectory colors match the three decoded-image columns. The three candidates' predicted features and the current DINOv2 grid pass through the same CondGrid network and frozen RAE decoder. Changing the trajectory shifts the decoded viewpoint while preserving scene structure. These views show responses to alternative plans; without matching observations, their accuracy cannot be measured.

Removing the predicted vector preserves much of the image structure because the reconstruction network still receives the current grid. At guidance 2, the observed future's pooled vector improves regional patch cosine by +0.005 to +0.041 over this vector-free branch. For the model-selected query, whole-grid cosine is 0.681 at guidance 2 and 0.707 at guidance 1, versus 0.701 without the vector. The reconstruction therefore depends on both the pooled vector and the current spatial grid; at guidance 2 the predicted vector does not raise whole-grid similarity to the observed future.

Current-view semantic masks partition the grids into road, dynamic objects, and other static regions. Table 9 pools patch-wise cosine over the three query pairs, the patches in each region, and six scenes. The resulting similarities measure how much the reconstructed regions respond to different queries in feature space, before the decoder renders an image.

Region	Grid cosine
Road	0.750
Dynamic objects	0.639
Other static	0.606

Table 9: **Cross-query readout consistency.** Higher cosine means less variation between query-conditioned grids.

Additional road scenes. Figure 11 adds six locations absent from Figure 8, covering broad junctions, urban streets, and curved roads. Changing the query shifts the road opening and visible building surfaces, with subtler differences between nearby plans. All eighteen images are rendered with the same fixed CondGrid network and RAE decoder.



Figure 11: **Additional visual readouts under alternative plans.** Each row shows the queried trajectories, current photograph, three visual readouts, and observed future (+2 s). Left and Right denote relative lateral alternatives, including variants of the same turn. The center query is model-selected in (a) and logged in (b–f); observed futures always follow the logged path in that scene. Alternative-plan images are qualitative readouts from the separately trained reconstruction network.

Reading the full gallery. Figure 12 shows all 30 pre-specified navtest scenes, ordered by decreasing change between current and future DINOv2 features. The columns show the current photograph, the logged-query reconstruction of the model's predicted feature, a decoder ceiling obtained by feeding the observed future's full patch grid directly to the same frozen RAE decoder, and the observed future. The ceiling isolates decoder quality; the reconstruction also depends on how much spatial information survives pooling and is captured by future prediction.

Low-change scenes are legible because the current grid supplies their spatial structure. With larger viewpoint changes, the pooled feature leaves several possible spatial layouts unresolved.

CondGrid is deterministic; diffusion samples from the same fixed feature. On the same 30 gallery scenes, diffusion yields sharper images but lower feature and pixel fidelity. Sampling occurs in the reconstruction network; JEPA emits one vector per query. The fixed-pool interventions in Appendix D.1 assess how candidate states affect trajectory ordering and selection performance.

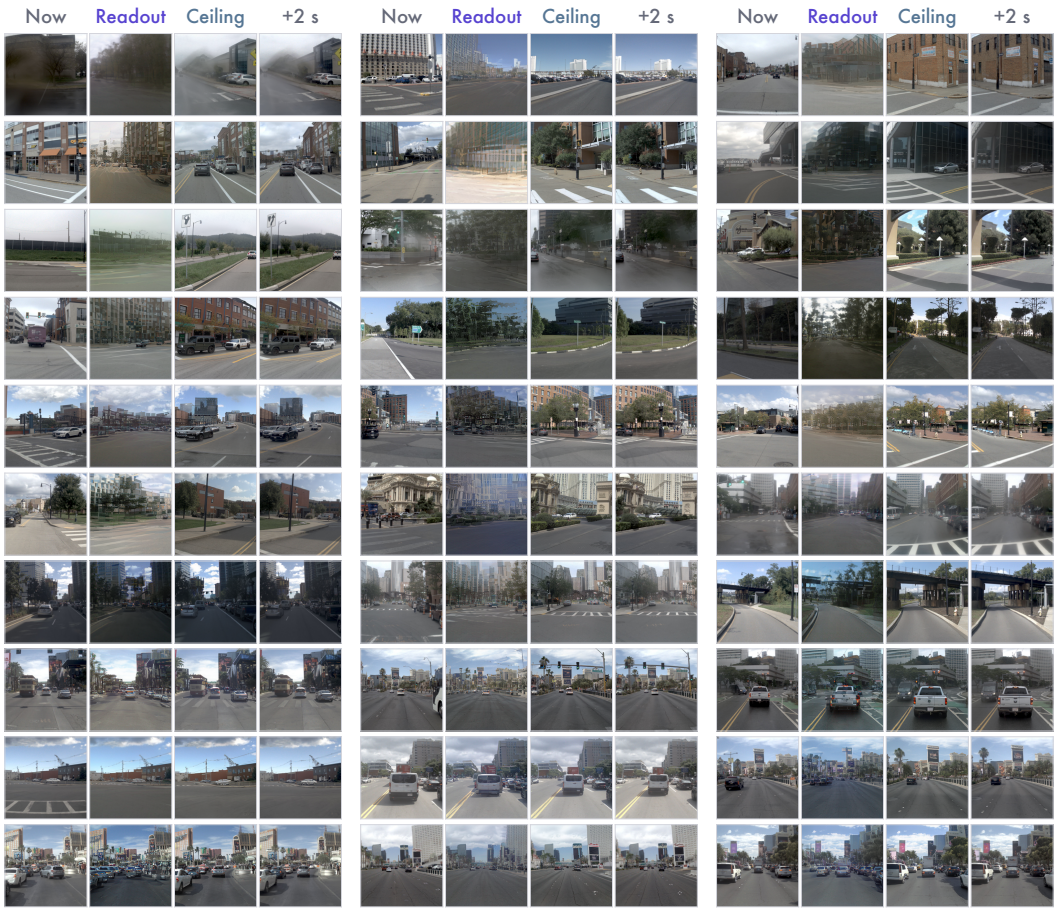

Figure 12: **Complete post-hoc decoding gallery.** Thirty pre-specified navtest scenes, three per row, sorted by decreasing current-to-future DINOv2 feature change. Each group shows the current frame (Now), executed-query readout, full-grid decoder ceiling, and observed future (+2 s). Logged-query reconstructions combine predicted features with the current spatial grid through the separately trained reconstruction network. The ordering is independent of decoding quality.

C.3 Candidate-State Geometry

We examine the candidate states used by the scorer. Across 128 navtrain scenes with 64 bank candidates each, the 256-dimensional states z have effective rank 17.9. Within-scene candidate differences account for 75.8% of total state variance.

The scorer compares candidates within a scene, so we examine the states of each scene separately. The 128 scenes come from 16 navtrain logs, eight per log. We group candidates by simulator

outcome: no collision and no drivable-area violation (NC = DAC = 1), collision (NC < 1), or drivable-area violation (DAC = 0). On average, 80.2% of a candidate's five nearest states in its scene share its outcome group, compared with 76.8% for the five nearest trajectories and 52.0% after shuffling outcome labels within each scene. The state agreement exceeds the shuffled value in all 128 scenes and the trajectory value in 98 of them. Figure 13 shows four scenes.

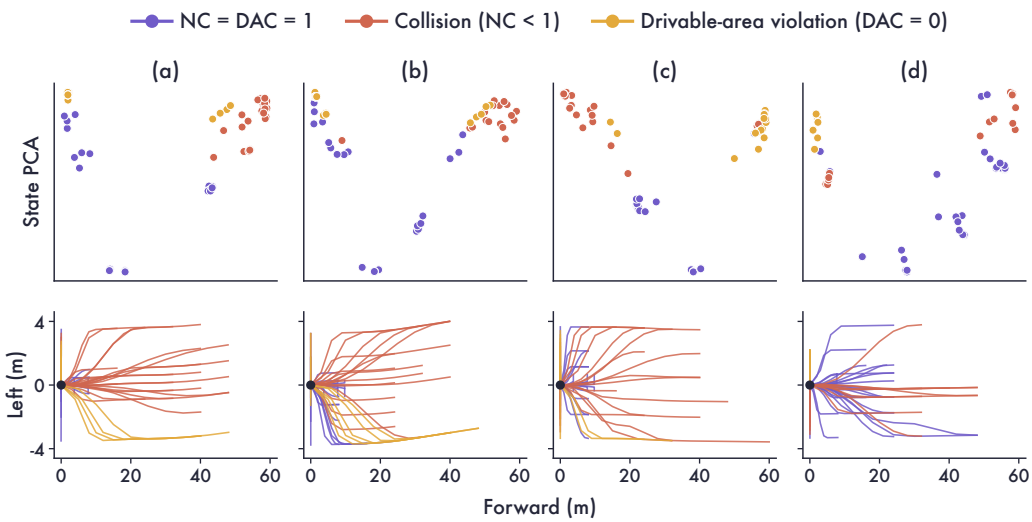

Figure 13: **Candidate states within four scenes.** Top: the first two principal components of the 64 candidate states in each scene, which explain 64–75% of their variance. Bottom: the same candidates' bank trajectories in the ego frame, with axes not to scale. Candidates with both a collision and a drivable-area violation are colored as drivable-area violations. The four scenes come from different logs and were chosen for balanced outcome groups (at least 15 candidates each). Vertical segments at the origin are bank trajectories that pair zero forward speed with a lateral transition.

C.4 Supervision Geometry and Candidate Comparisons

We expand the local model of Section 3.3 and prove Proposition 1. The analysis follows the principle that supervision should constrain differences relevant to candidate selection, as in transductive experimental design (Fiez et al., 2019). Here the observation maps are the network's visual and outcome readouts, whose parameters are themselves learned.

Observation maps and candidate context. Fix scene tokens, encoded ego states, trajectory coordinates, and the candidate set used by each predictor call. Let $\theta \in \mathbb{R}^p$ collect the parameters of ϕ, P, g, H. For an executed query, define the normalized visual prediction

$$v_n(\theta) = \frac{g_\theta(z_n^{\text{exec}}(\theta))}{\|g_\theta(z_n^{\text{exec}}(\theta))\|_2}.$$

We work around a fitted θ_0 at which these predictions are nonzero and the readouts are differentiable; we do not assume a stationary training loss. Stack the executed-query predictions into $F(\theta)$ and the supervised outcome logits into $O(\theta)$, with Jacobians J_{fut} and J_{out} at θ_0. Normalization follows the cosine loss (Equation 4); only the latent state's readout $g(z)$ is compared with the observed future.

Each state retains the context of its actual predictor call: an executed query is evaluated alone, generated queries jointly, and bank queries in a separate set. Shared weights couple these calls. For a candidate pair (i, j) in a fixed scene and pool, define $\hat{\Delta}_{ij}(\theta) = \hat{J}_i(\theta) - \hat{J}_j(\theta)$ using Equation 2, and $c_{ij} = \nabla_\theta \hat{\Delta}_{ij}(\theta_0)$. This pool may come from a new scene or contain unsupervised queries.

Training-loss geometry. The first-order changes under $\theta_0 + \delta$ are $J_{\text{fut}}\delta$, $J_{\text{out}}\delta$, and $c_{ij}^\top\delta$. The generalized Gauss–Newton (GGN) surrogate of the outcome and future terms of Equation 7 has the curvature term $\frac{1}{2}\delta^\top(J_{\text{out}}^\top W_{\text{out}} J_{\text{out}} + \lambda_{\text{fut}} J_{\text{fut}}^\top W_{\text{fut}} J_{\text{fut}})\delta$. Here W_{out} holds the logit-space curvature $w_i\, p(1-p)$ of each binary cross-entropy term, including soft-label progress. The bank's progress

term is an L1 loss on $p_{\mathrm{EP}} = \sigma(\ell_{\mathrm{EP}})$; it is piecewise linear in p_{EP} but not in the logit, and we omit it from the logit-space GGN, so G_{out} is a partial curvature surrogate rather than the exact curvature of the full mixed loss. On unit vectors the cosine loss equals $\frac{1}{2}\|v_n - \bar{u}_n\|_2^2$ with $\bar{u}_n = u_n^{\mathrm{fut}}/\|u_n^{\mathrm{fut}}\|_2$, so $W_{\mathrm{fut}} = I/M$. Write $A_O = W_{\mathrm{out}}^{1/2} J_{\mathrm{out}}$, $A_F = W_{\mathrm{fut}}^{1/2} J_{\mathrm{fut}}$, and $\lambda = \lambda_{\mathrm{fut}}$. For $\mu, \epsilon, \lambda > 0$, define

$$\mathcal{E}_\epsilon(G) = \left\{\delta : \mu\|\delta\|_2^2 + \|A_O\delta\|_2^2 + \lambda\|A_F\delta\|_2^2 \le \epsilon^2\right\}. \tag{11}$$

The quadratic form is the positive-semidefinite curvature part of this local surrogate plus a trust-region term $\mu\|\delta\|_2^2$ that keeps δ where the linearization holds. It defines the trust region of a first-order sensitivity analysis; it is not the complete change in the mixed training loss, whose first-order and residual-curvature terms are omitted. The budget ϵ bounds the change; neither μ nor ϵ enters training or inference. Equivalently, $\mathcal{E}_\epsilon(G) = \{\delta : \delta^\top G\delta \le \epsilon^2\}$ with the positive-definite matrix

$$G = G_O + \lambda A_F^\top A_F, \qquad G_O = \mu I + A_O^\top A_O. \tag{12}$$

All comparisons below hold at the same θ_0 and perturbation budget. Define the maximal first-order change in a candidate score difference, or radius, as

$$\rho_{ij}(G) := \sup_{\delta \in \mathcal{E}_\epsilon(G)} |c_{ij}^\top \delta| = \epsilon\sqrt{c_{ij}^\top G^{-1} c_{ij}}. \tag{13}$$

The main-text quantities are $\rho_{ij}^{\mathrm{out}} = \rho_{ij}(G_O)$ and $\rho_{ij}^{\mathrm{out+fut}} = \rho_{ij}(G)$. Thus both use the same model and budget; only the added visual-output constraints differ.

Proof of Proposition 1. Set $v = G^{1/2}\delta$, so $\|v\|_2 \le \epsilon$. Cauchy–Schwarz yields

$$|c_{ij}^\top \delta| = |(G^{-1/2}c_{ij})^\top v| \le \epsilon\sqrt{c_{ij}^\top G^{-1} c_{ij}}.$$

For $c_{ij} \neq 0$, equality is attained at both signs of

$$\delta^* = \frac{\epsilon G^{-1} c_{ij}}{\sqrt{c_{ij}^\top G^{-1} c_{ij}}}.$$

Scaling attains every intermediate value; the case $c_{ij} = 0$ has radius zero. This proves Equation 13.

To establish strict contraction, set $b = A_F G_O^{-1} c_{ij}$. The Woodbury identity gives

$$\rho_{ij}(G_O)^2 - \rho_{ij}(G)^2 = \epsilon^2 b^\top \left(\lambda^{-1} I + A_F G_O^{-1} A_F^\top\right)^{-1} b. \tag{14}$$

The middle matrix is positive definite. Hence the reduction is nonnegative, proving Equation 6, and is strictly positive exactly when $A_F G_O^{-1} c_{ij} \neq 0$. For nonzero c_{ij}, the maximal-change direction under outcome constraints is proportional to $G_O^{-1} c_{ij}$, so this is precisely the condition that visual predictions vary along that direction. Because W_{fut} is invertible, it is the condition $J_{\mathrm{fut}} G_{\mathrm{out}}^{-1} c_{ij} \neq 0$ of Proposition 1, with $G_{\mathrm{out}} = G_O$ and $G_{\mathrm{fut}} = J_{\mathrm{fut}}^\top W_{\mathrm{fut}} J_{\mathrm{fut}}$. This completes the proof. The inverse G_O^{-1} emphasizes comparison directions weakly constrained by the existing outcome observations; A_F determines whether those same directions are also constrained by visual supervision.

Score differences and selection. The linearized score difference spans the interval

$$[\hat{\Delta}_{ij}(\theta_0) - \rho_{ij}(G),\ \hat{\Delta}_{ij}(\theta_0) + \rho_{ij}(G)].$$

Every local model orders i strictly above j if and only if $\hat{\Delta}_{ij}(\theta_0) > \rho_{ij}(G)$. Consequently, a fitted winner k remains the unique winner throughout $\mathcal{E}_\epsilon(G)$ if and only if this inequality holds for every $j \neq k$. The condition allows nonzero variation in both score values and predictive states, provided it does not reverse a winning comparison between candidates.

With exact first-order output preservation, the unconstrained directions are $\mathcal{N} = \ker A_F \cap \ker A_O$. A comparison is fixed along every direction in $\mathcal{N}$ precisely when $c_{ij} \perp \mathcal{N}$, or equivalently when c_{ij} belongs to the row space of the vertically stacked observation Jacobians. This is the zero-output-change counterpart of the finite-radius analysis of candidate comparisons.

Additional candidate outcomes. Let B stack the Jacobians of outcome predictions at newly supervised candidate queries, with their fixed weights absorbed into its rows. Adding these constraints yields $G_+ = G + B^\top B$. Applying the same identity gives

$$\rho_{ij}(G)^2 - \rho_{ij}(G_+)^2 = \epsilon^2 a^\top (I + BG^{-1}B^\top)^{-1} a, \qquad a = BG^{-1}c_{ij}. \tag{15}$$

The reduction is positive exactly when $BG^{-1}c_{ij} \neq 0$. Additional outcome labels therefore help when they constrain comparison directions that existing supervision leaves unresolved.

Relation to learning. The result characterizes local sensitivity at a fitted network under the same quadratic perturbation budget, not the optimization trajectory or global planning performance after training. Its quadratic form is a partial generalized Gauss–Newton metric for the BCE and cosine terms, not the complete mixed training objective. No stationarity of the individual loss terms is assumed; the analysis concerns local first-order score sensitivity under this metric, and μ restricts it to the region where the linearization holds. Target values affect the fitted network and hence its local geometry; replacing a target at fixed parameters does not itself change the output Jacobian. Section 4.3 evaluates the learning effects of future targets and correctly paired candidate outcomes. The analysis identifies a condition for complementary supervision without assuming that latent-state coordinates or complete alternative futures are recovered.

D Candidate States and Trajectory Selection

With fixed trained weights, we measure how state reassignment affects selection and describe the OGBench-Cube setup with the LeWM planner.

D.1 Candidate-State Interventions with Fixed Pools

With the checkpoint and pool fixed, Table 10 compares matched states, scene-mean replacement, and within-scene reassignment on the full test set. The common oracle is 99.33 PDMS; selected PDMS equals this oracle minus the regret.

Candidate-state intervention	Oracle regret (95% CI)	Δ vs. matched (95% CI)
Matched candidate state	5.14 [4.66, 5.62]	–
Scene-mean replacement	28.23 [25.97, 30.50]	+23.09 [+20.98, +25.17]
Within-scene derangement	19.86 [18.65, 21.08]	+14.72 [+13.72, +15.75]

Table 10: **Candidate-state interventions.** Every intervention uses World4Scorer and the same 64 candidates of the released DrivoR model for all 12,146 NAVSIM-v1 test scenes, so matched PDMS is not the generator-pool result of Table 1. Regret is measured relative to the common pool oracle; positive differences are worse. Paired intervals bootstrap over the 136 evaluation logs.

Replacing all candidate states with their scene mean gives every candidate the same score, so the first-index tie rule selects one fixed pool index on all 12,146 scenes. This row measures selection without a ranking signal. Reassignment preserves states but changes their assigned candidates.

State reassignment disrupts trajectory selection. Reassignment changes the selected candidate on 100% of scenes and reduces ordering accuracy to 49.9%, approximately chance. Together with the higher regret, this shows that selection depends on the candidate–state correspondence.

Variation across reassignments. All 100 random reassignments reduce selected PDMS. Their means range from 77.44 to 81.99, averaging 79.73 (95% log-cluster interval [78.64, 80.85]). The average loss against matched states is 14.47 points (paired interval [13.65, 15.29]).

These repeats use the epoch-23 checkpoint and all 12,146 scenes. Each draws one fixed-point-free permutation of the 64 indices and applies it across scenes, using seeds 20260909–20261008. The score heads read each state plus the same encoded ego state, so reassigning states permutes the candidate scores. Every repeat uses the same native-V1 utilities and first-index tie rule.

We average repeats within each scene, then form intervals from 100,000 bootstrap draws of the 136 logs. Because all repeats share one checkpoint, this variation isolates the reassignment draw.

Comparison with DrivoR. Reassigning DrivoR's scorer-attention outputs raises regret by +14.11 points over its matched 5.64 (95% log-cluster interval [+13.01, +15.19]) and reduces ordering accuracy from 72.5% to 50.2%. State assignment thus matters even without future prediction.

The control matches the World4Scorer pools and native-V1 utilities across 12,146 scenes and 136 logs, with permutation seed 20260802, first-index ties, and 100,000 log-bootstrap draws.

Selected scenes. Figure 7 shows twelve scenes from eleven logs: four left turns, four straight paths, and four right turns. The scenes show diverse geometry, distinct paths, and agreement with the driver. Matched states pick a pool-best path in every case. Purple stars mark these selections; amber dotted paths show the driver. Paths and maps use real, equally scaled coordinates.

D.2 Outcome-Based Scoring Inside the LeWM Planner

We use the official LeWM (Maes et al., 2026) implementation on OGBench-Cube (Park et al., 2025). Its world model predicts the result of an action sequence, and its planner searches for a sequence that reaches the goal. We freeze the world model and add outcome-based scoring to this search.

The planner runs cross-entropy-method (CEM) search over 300 action sequences of 5 blocks, each block 5 environment steps, and keeps the 30 best sequences at every iteration. The goal is the expert's cube pose 25 steps ahead, and each episode has a budget of 50 environment steps. Success requires placing the cube within 4 cm of this goal pose.

LeWM scores each sequence by the distance between its predicted final feature and the goal feature, favoring smaller distances. Our scorer takes these two features, their distance, and the action sequence, and predicts its outcome score. At each CEM iteration, we standardize the negative distance and the learned score across candidates and add the score to the distance with weight $\lambda_{\mathrm{cube}} = 0.3$. This combined score selects the retained sequences under the same world model and search budget.

The scorer learns from simulator outcomes of candidates collected by the LeWM planner, using training seeds separate from evaluation. Both planners use 10 CEM iterations, following LeWM's Appendix D. We choose λ_{cube} by success on tuning seeds 200–203; these seeds are separate from the reported evaluation. Table 5 reports the mean over seeds 0–9 and 42, LeWM's released evaluation seed, each with 50 paired episodes. Each pair shares an initial state and goal.

Evaluation seeds. All seeds use the same trained scorer, weight, and 10-iteration CEM budget. LeWM succeeds on 379/550 episodes (68.91%) and outcome-based scoring on 405/550 (73.64%), a gain of +4.73 percentage points. The scorer rescues 39 episodes and loses 13; success improves on 9 seeds, ties on 1, and decreases on 1. These runs vary the evaluation initial states and goals while the world model and scorer stay fixed.

Scorer architecture and training. The scorer input concatenates the layer-normalized predicted final and goal features, their difference and elementwise product, the log-transformed LeWM distance, and the mean and standard deviation of the action sequence (819 dimensions). A layer-normalized MLP with GELU activations and dropout 0.05 maps this input through widths 256 and 128 to a scalar. The target is the negated minimum cube-to-goal distance along each candidate's simulator rollout, standardized within its candidate group. The loss adds smooth-L_1 regression and a listwise cross-entropy term with weight 0.25 and softmax temperature 0.5.

Training candidates come from LeWM planning runs with seeds 100–107. Episodes used in any evaluation are excluded, as are candidate groups whose target standard deviation is below 10^{-4}. An episode-disjoint split gives 2,982 training groups from 262 episodes and 725 validation groups from 65 episodes. We train for 80 epochs with AdamW (learning rate 10^{-3}, weight decay 10^{-4}, cosine schedule, 64 groups per batch) and keep the epoch with the highest validation pairwise ordering accuracy (epoch 6, counting from zero).

E Sequential Planning and Closed-Loop Evaluation

We analyze failures between consecutive selections, measure the effect of inertial re-ranking and its history input, and describe the Bench2Drive closed-loop evaluation.

E.1 Extended-Comfort Failure Taxonomy

Table 11 classifies our planner's 2,717 framewise extended-comfort failures among 10,040 consecutive pairs, according to the alternatives available at each decision:

- **Score–continuity conflict:** a passing alternative exists, but the selected failing candidate has higher utility by more than the near-tie threshold.
- **Near-tie:** a passing alternative is within $\varepsilon = 4.1 \times 10^{-5}$ of the selected utility.
- **Pool-limited:** no candidate passes the EC check.

Score–continuity conflicts account for 71.3% of failures, whereas near-ties account for 0.7%. Most pool-limited failures (88.7%) fall in the most dynamic quarter of pairs (Appendix B.2). After inertial re-ranking, 91.7% of residual failures have no compatible candidate in the pool.

For DrivoR, 7.3% of pairs lack compatible candidates, so no selection from its pools can exceed an EC of 92.66.

The EC check also rejects some logged driving: human plans pass on 90.1% of all adjacent pairs and on 16.5% of our pool-limited pairs. Thus part of the remaining failure rate comes from motion-profile differences that also occur in logged human driving.

Failure class	Share of failures	Repaired by Ψ
Score–continuity conflict	71.3%	97.6%
Near-tie	0.7%	95%
Pool-limited	27.9%	3.8%

Table 11: **Consecutive-choice failures.** Classification of 2,717 framewise EC failures among 10,040 consecutive pairs.

E.2 Changes in Consecutive Selections

With our planner's weights and candidates fixed, inertial re-ranking changes 1939 consecutive pairs from EC failure to pass and 18 from pass to failure. On the repaired pairs, re-ranking reduces heading jitter by 32%, lateral discrepancy by 21%, and the L_2 distance between consecutive plans by 11%. Among changed selections, distance to the logged plan drops from 1.87 to 1.51 m.

E.3 Re-Ranking with Fixed Candidate Pools

Inertial re-ranking also improves DrivoR: with DrivoR's weights and candidates fixed, it raises EPDMS from 90.66 to 92.27, a +1.61-point gain from changing the final selection rule alone.

Sensitivity to the state weight. The reported results fix $\lambda_{\text{state}} = 1$. We re-run inertial re-ranking at $\lambda_{\text{state}} \in \{0.25, 0.5, 1, 2\}$ on the fixed pool and scores of Coverage, a variant of our planner whose generator is also trained to cover high-scoring bank trajectories. The four runs give 92.42, 92.47, 92.49, and 92.45 EPDMS: varying the weight eight-fold moves EPDMS by at most 0.07 points. The gain over framewise selection persists throughout this range of compatibility-penalty weights.

E.4 History Corruption Controls

Using the correct previous plan improves EPDMS by 2.96 points over wrong-log history (95% CI [2.63, 3.34]) and by 1.01 points over two-step-old history (95% CI [0.86, 1.16]). Table 12 uses the same fixed Coverage pool and scores as Appendix E.3 and changes only the history supplied to the EC check, testing whether the benefit comes from the immediately preceding plan. Its last row

instead resubmits the previous frame's selected plan without choosing among the current candidates; it scores below framewise selection, so the gain does not come from repeating plans.

Previous plan supplied	V2 EPDMS
Memoryless	89.9
Wrong-log previous plan	89.5
Same-log, two-step-old plan	91.5
Correct previous plan	92.5
Copy previous plan	82.0

Table 12: **History-source controls for inertial re-ranking** on Coverage. Memoryless is framewise selection; Copy previous plan resubmits the previous frame's plan. Differences use unrounded EPDMS.

E.5 Closed-Loop Evaluation on Bench2Drive

We train the planner on the official 1000-clip subset and evaluate it in CARLA using Bench2Drive (Jia et al., 2024). We evaluate an adapted system; published baselines are not retrained under matched conditions. We initialize from a route-blind checkpoint trained on the same clips, add the route-point input, and use the simulator and route inputs below. Table 13 compares Driving Scores, and Figure 14 shows six completed scenarios with camera frames every five seconds.

The CARLA ego input adds the route point 20 m ahead, expressed in ego-frame meters, to the NAVSIM pose, velocity, acceleration, and command inputs. The same coordinate conversion supplies this route point during training and deployment.

We choose the distance by scanning 10–40 m over 15,589 frames from 60 training clips. At 20 m, the point lies on the side of the turn for 97.4% of left-turn and 100.0% of right-turn frames, while the route ends before the point on only 10.5% of frames. We then add two zero-initialized input columns to the route-blind checkpoint and retain its training data, loss, and schedule.

Training. The model and losses follow NAVSIM with two changes. Outcome labels are computed against the logged agent boxes and a drivable-area raster of each clip, and extra negatives come from a trajectory vocabulary instead of the CLOVER bank. The future loss has weight 0.1, and its gradient into the predictor is scaled by 0.05. Both the route-blind and the route-point model train for 15 epochs with AdamW at a learning rate of 7.5×10^{-5} and 24 clips per GPU.

Two deployment rules need no retraining:

- **Command retention** keeps a route node active until the vehicle passes it, preserving the turn command through an intersection.
- **Route re-ranking** measures agreement with the route polyline within 30 m using an exponential kernel of average waypoint-to-route distance, weighted toward later waypoints. We standardize this agreement across candidates and add it to the learned utility with weight 4σ, where σ is the pool's utility standard deviation.

We select the plan with the highest combined score.

Method	Driving Score
AD-MLP (2024)	18.05
TCP (2022)	40.70
UniAD-Tiny (2023)	40.73
VAD (2023)	42.35
UniAD-Base (2023)	45.81
TCP-traj w/o distillation (2022)	49.30
TCP-traj (2022)	59.90
WoTE (2025b)	61.71
ThinkTwice (2023b)	62.44
DriveAdapter (2023a)	64.22
Drive-JEPA (2026)	64.52
SafeDrive (2026)	66.77
ReCogDrive (2026b)	71.36
Ours	**73.27**

Table 13: **Extended Bench2Drive comparison.** Driving Score.

Route re-ranking uses the 1 m `set_global_plan` polyline; command retention uses the reference agents' 50 m waypoints (Jia et al., 2024). Ego pose, velocity, and acceleration come from CARLA; the reference agents estimate these quantities from GNSS and IMU measurements.

To avoid map-streaming crashes, we load the map within 4000 m on routes 11715, 11755, 23659, 23695, 23708, 24041, and 24071, and 2000 m on route 3800.

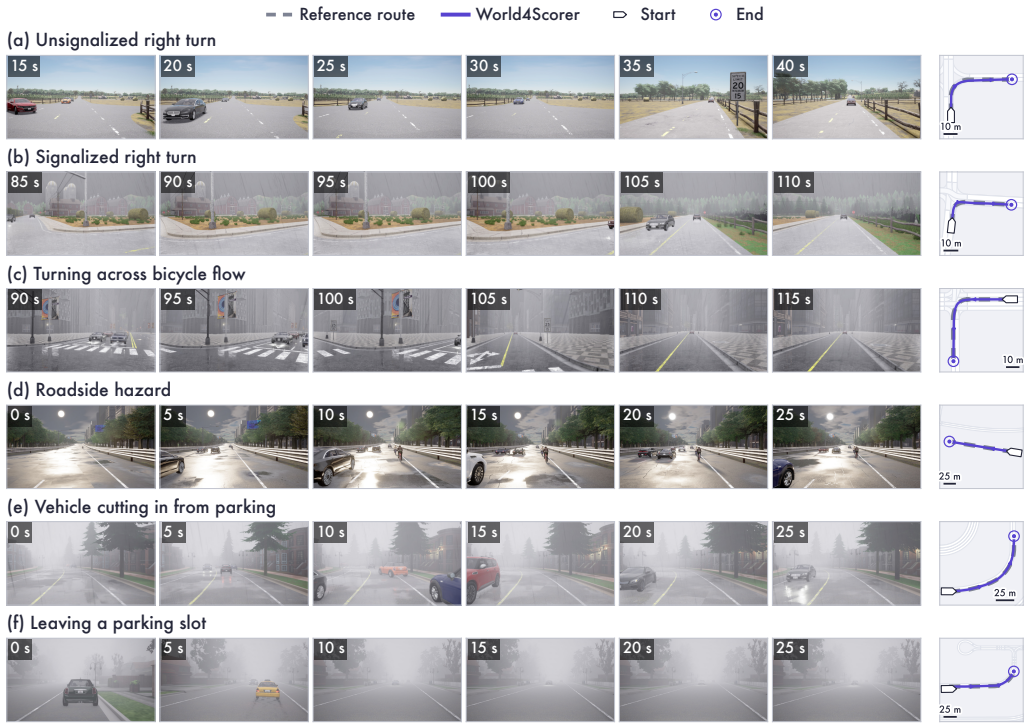

Figure 14: **World4Scorer in six Bench2Drive scenarios.** Each row shows six consecutive camera views at 5 s intervals around a driving maneuver. The small map at right shows the full reference route (dashed) and driven trajectory (purple). All six routes are completed by the Ours configuration of Table 13, with Driving Score and Route Completion both equal to 100.