DON'T REWRITE, BUT CORRECT: CLOSED-LOOP MANIFOLD CORRECTION VIA RL POST-TRAINING FOR GENERATIVE DRIVING POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative driving policies have become a dominant source of pretrained driving priors in autonomous driving. In particular, flow-based planners trained by imitation learning capture expert behavior well on routine scenarios, whereas a few long-tail states can trigger compounding errors during closed-loop interaction, leading to a substantial gap between offline generative quality and closed-loop performance. To this end, we propose **N**eighborhood **A**ction **C**orrection, a closed-loop neighborhood post-training framework for generative driving planners. Different from the typical RL fine-tuning approach that updates the whole pretrained planner and may reshape its behavior on routine states, NAC freezes the generative planner and only trains a lightweight corrector to adaptively correct its prior action, particularly on long-tail states. The trainable correction is bounded by a neighborhood and confined to the only variable that enters vehicle dynamics under receding-horizon execution. NAC enables efficient improvement of the planner's closed-loop performance while preserving the strong prior manifold of the generative policy. Experiments on nuPlan show that NAC achieves state-of-the-art closed-loop performance with flow-matching planner and outperforms the evaluated RL fine-tuning methods.

1 INTRODUCTION

Generative imitation learning has become a dominant paradigm for learning-based autonomous driving planners (Zheng et al., 2025; Liao et al., 2025; Tan et al., 2025; Wang et al., 2025), achieving strong open-loop trajectory regression by modeling structured multimodal trajectory distributions. However, planning quality is evaluated in closed-loop interaction, where open-loop accuracy does not necessarily translate into closed-loop robustness (Dauner et al., 2023). These planners capture rich driving regularities from demonstrations (Li & He, 2026), inducing a trajectory-level prior over plausible behaviors. While their average error is small on routine scenarios, the remaining failures are often structural rather than capacity-related (Pan et al., 2026). Due to the non-uniform coverage of demonstration data, sparsely supported states may retain residual biases that open-loop training cannot fully remove. Under receding-horizon execution, these biases would accumulate into closed-loop drift. Hence, post-training should preserve the learned driving prior while introducing closed-loop supervision for corner cases where the prior remains insufficient.

Since the gap appears in closed-loop execution, reinforcement learning provides a natural source of supervision. The most direct approach is to fine-tune the entire pretrained generator with closed-loop reward (Fig.1(a))(Li et al., 2026; Tang et al., 2026; Huang et al., 2025), allowing the prior itself to absorb the reward signal. However, such fine-tuning is structurally invasive: as gradients update the whole prior, local corrections for corner cases may also reshape routine states. This can distort learned driving regularities and destabilize training. To limit this shift, existing methods introduce hardcoded structural priors at the trajectory or candidate level (Li et al., 2026; Gao et al., 2026). These designs compress the search space, but the hardcoded decomposition tightly coupled driving trajectories with formulaic regularities, therefore introduces bias and adapts poorly to generalized state.

A more flexible scheme is to keep the generator frozen and learn only a bounded correction, preserving the driving prior while incorporating closed-loop supervision without handcrafted rules. A natural

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.

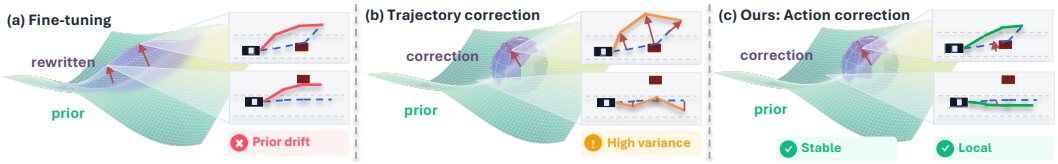

Figure 1: **Three approaches for RL post-training of a pretrained generative driving prior.** *(a) Fine-tuning*: directly updates the generator weights under closed-loop reward, which reshapes the prior manifold and shifts the policy away from the demonstration-supported region. *(b) Trajectory-correction*: freezes the generator and learns a bounded trajectory-level correction, which preserves the prior. However, the high-dimensional correction inflates supervision variance. *(c) Action-correction (ours)*: keeps the generator frozen, and confines trainable correction to the neighborhood of executed action. The prior and its action anchor are preserved, while reward precisely supervises the low-dimensional action, keeping the correction stable and local.

place for performing such a correction is trajectory space, since the planner outputs trajectories and the prior manifold remains explicit (Fig.1(b)). However, this choice is costly. A trajectory spans many future steps and coordinates, making scalar-reward optimization unstable and high-variance. Furthermore, the supervision target is misaligned with execution. Trajectory-level training supervises the entire planned future, whereas closed-loop simulation follows receding-horizon execution: at each cycle, only the first action enters vehicle dynamics and is evaluated by the per-step reward(Jaeger et al., 2025; Dauner et al., 2023). The rest of the trajectory is replaced by the next prediction and never directly affects the vehicle state. Hence, although trajectory-level correction preserves the prior, a mismatch remains between what is optimized and what is executed.

In contrast, correcting the executed action substantially reduces the optimization space from a multi step trajectory to a single per step action, while improving optimization stability by focusing learning on execution-relevant directions. The action is also exactly what enters vehicle dynamics and what the per-step reward evaluates, aligning supervision with closed-loop execution by construction. Note that this preserves trajectory-level pretraining: the full trajectory shapes the encoder representation and the controller-projected anchor through which the prior influences behavior. Building on these observations, we propose a closed-loop neighborhood post-training framework dubbed **N**eighborhood **A**ction **C**orrection (NAC) for generative policies. We freeze the pretrained planner, and project the planned trajectory into a prior action a_t^0 by controller. Then NAC restricts all trainable correction to a closed neighborhood around a_t^0 (Fig.1(c)). The prior and its action anchor are preserved, while reward supervises the variable that directly drives the vehicle. We instantiate NAC as a PPO-trained(Schulman et al., 2017) corrector on top of the frozen Flow-Planner (Tan et al., 2025). Our main contributions are:

- We propose NAC, a closed-loop manifold correction framework for pretrained generative driving policies. NAC preserves the trajectory prior learned from imitation pretraining, while better aligning the planner with closed loop objectives.
- To our knowledge, NAC is the first framework to post-train frozen trajectory-generative driving planners through RL corrections to controller-projected actions. Our analysis explains how this design bridges trajectory-level imitation priors and step-wise RL supervision by avoiding controller-null trajectory directions.
- Experiments demonstrate NAC's effectiveness across flow-based and diffusion-based priors with light planner-specific tuning. On nuPlan-val14, test14-hard, and InterPlan10, NAC with a lightweight flow matching planner achieves state-of-the-art closed-loop performance among learning-based planners, with up to $+10.85$ over its prior policy on InterPlan10 and $+6.11$ over fine-tuning baselines on test14-hard.

2 RELATED WORK

2.1 GENERATIVE PLANNERS FOR AUTONOMOUS DRIVING

Generative driving planners learn structured multimodal trajectory distributions from demonstrations and have become the dominant paradigm in recent autonomous driving research. Diffusion-based planners denoise the future trajectory under rich driving conditions, including Diffusion-Planner (Zheng et al., 2025) with flexible guidance, DiffusionDrive (Liao et al., 2025) with truncated diffusion for efficient end-to-end driving, TransDiffuser (Jiang et al., 2025) with decorrelated multimodal representation, and Hyper Diffusion-Planner (Zheng et al., 2026a). Flow-matching planners offer a complementary path with shorter sampling horizons and cleaner conditioning, including Flow-Drive (Wang et al., 2025) with moderated flow matching, Flow-Planner (Tan et al., 2025) integrating advanced interactive behavior modeling, and GuideFlow (Liu et al., 2026) with constraint-guided flow matching.

2.2 POST-PROCESSING AND REINFORCEMENT LEARNING FOR AUTONOMOUS DRIVING

In autonomous driving, post-processing over a frozen generator applies reward through auxiliary modules rather than policy updates, including RAD-2 (Gao et al., 2026) with discriminator-based reranking, Gen-Drive (Huang et al., 2025) with reward-model guidance on diffusion samples, and Diffusion-ES (Yang et al., 2024) with gradient-free trajectory search. Fine-tuning on the generator updates the trajectory distribution itself, as in PlannerRFT (Li et al., 2026) and Plan-R1 (Tang et al., 2026). ResAD (Zheng et al., 2026b) instead places a trajectory-level residual on a physics-extrapolated reference. Pure RL approaches train the driving policy directly without generative pretraining, including CaRL (Jaeger et al., 2025) with simple rewards, Roach (Zhang et al., 2021) with privileged-coach distillation, CarPlanner (Zhang et al., 2025) with autoregressive trajectory RL, and self-play at scale (Cusumano-Towner et al., 2025).

2.3 RELATION TO PRIOR CORRECTION METHODS

Residual action learning is well established in embodied control, where a learned additive policy refines the output of a nominal controller (Johannink et al., 2019; Silver et al., 2018). RuN (Li et al., 2025) pairs a pretrained conditional motion generator with a lightweight residual policy, Policy Decorator (Yuan et al., 2025) refines behavior-cloning, diffusion, and transformer policies through a residual policy, EXPO (Dong et al., 2026) uses a lightweight Gaussian edit policy over an expressive base policy, and Dodeja et al. (2026) guides residual learning with epistemic uncertainty. Although their update mechanisms differ, these methods operate on policy outputs that are directly executable actions or action chunks.

A trajectory-based driving planner, by contrast, generates a future trajectory which is mapped to the corresponding action by a tracking controller. This intermediate controller permits correction at two locations: before the controller in trajectory space or after it in action space. CorrectionPlanner (Guo et al., 2026) realizes the pre-controller placement through an iterative self-correction loop in autoregressive motion-token space. NAC instead adopts the post-controller placement by adding a bounded correction to the controller output. Relative to trajectory-space correction, it reduces the optimization dimensionality, aligns the correction with the variable executed under receding-horizon control, and preserves the pretrained trajectory manifold. Our analysis identifies controller-null trajectory directions as a source of excess gradient variance under matched action marginals, providing a principled basis for this correction design.

3 METHODOLOGY

3.1 PRELIMINARIES: THE CLOSED-LOOP DRIVING STACK

Prior flow policy. We adopt the Flow-Planner training framework (Tan et al., 2025) with further efficiency optimization in the supervised setting, and train a lightweight flow planner on nuPlan demonstration data to serve as the frozen generative driving prior G_θ. The planner factorizes into a

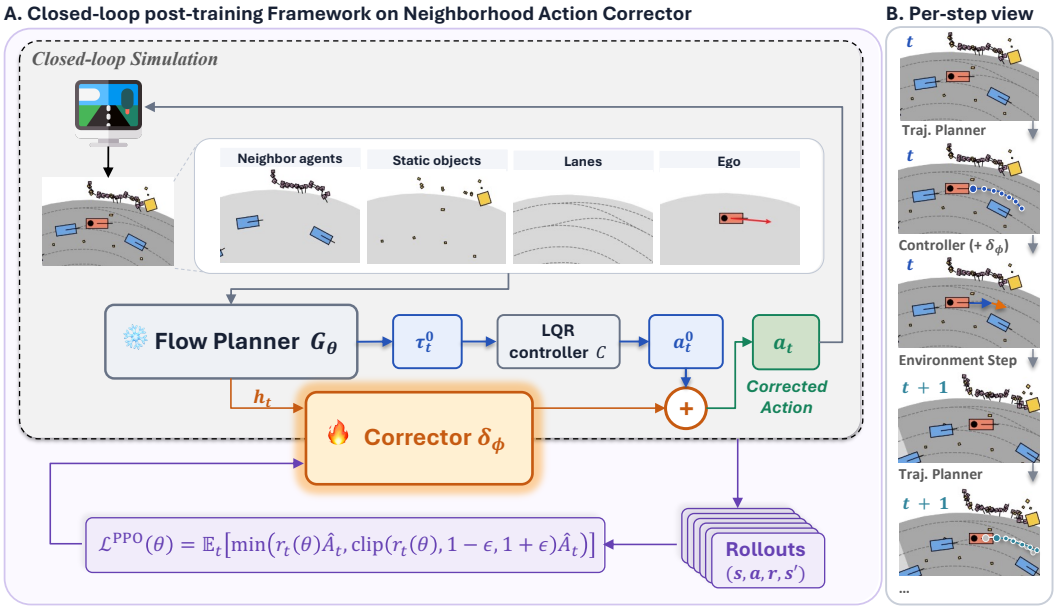

Figure 2: Overview of NAC. *(A) Closed-loop post-training framework*: the frozen Flow-Planner produces a prior trajectory from the current observation, then the LQR Controller projects the prior trajectory into the prior action (Sec. 3.1). Next, the Corrector emits a bounded correction, applying it to the prior action to form the executed action (Sec. 3.2). Online rollouts are collected into an on-policy batch and supervise the corrector by PPO post-training. *(B) Per-step receding-horizon view*: at plan cycle t the planner generates a fresh trajectory from the current observation, while only the first executed action enters environment dynamics. The remaining future portion is discarded at $t+1$, as the planner produces a brand-new trajectory on the new observation.

scene encoder E_θ and a flow-matching trajectory decoder D_θ, which is formulated as:

$$h_t = E_\theta(o_t), \qquad \tau_t^0 \sim D_\theta(\cdot \mid h_t), \tag{1}$$

where E_θ maps the current observation o_t to a scene-level hidden representation h_t, and D_θ generates a trajectory sample $\tau_t^0 \in \mathbb{R}^{H\times d}$ from Gaussian noise conditioned on h_t. Only the pair (τ_t^0, h_t) produced at each plan cycle enters post-training, which is abbreviated as

$$(\tau_t^0, h_t) = G_\theta(o_t). \tag{2}$$

D_θ instantiates the pretrained driving prior manifold introduced in Section 1. Its state-conditioned distribution $p_\theta(\tau \mid h_t)$ absorbs the driving regularities present in demonstrations, and each sample τ_t^0 lies in the supported region of this distribution for the current scene. The encoder output h_t therefore carries a compact summary of the scene context that shapes which trajectories are plausible under the prior. G_θ is held frozen throughout post-training and remains the sole source of this driving prior, while subsequent post-training only modulates how τ_t^0 and h_t are consumed downstream.

Closed-loop MDP and controller projection. The closed-loop MDP is given by states, observations, actions, and rewards,

$$s_t \in \mathcal{S}, \quad o_t \in \mathcal{O}, \quad a_t \in \mathcal{A}, \quad r_t = r(s_t, a_t, s_{t+1}). \tag{3}$$

Note that the trajectory does not enter vehicle dynamics directly. It is first projected by the LQR controller C built into the nuplan simulator at the current state s_t, yielding the executable action at the next execution frame,

$$a_t^0 = C(\tau_t^0, s_t), \qquad a_t^0 \in \mathbb{R}^2, \tag{4}$$

where the two channels of a_t^0 correspond to acceleration and steering rate, respectively. Subsequently, a_t^0 is applied to a kinematic bicycle model which constitutes the vehicle dynamics for the current plan cycle.

Receding-horizon execution. The whole stack rolls out in a receding-horizon manner. At every plan cycle, G_θ is invoked anew on the latest observation o_t to produce a fresh trajectory, then the controller derives the action of current step for execution. Closed-loop return enters dynamics only through a_t in each step, while the future portion of the trajectory is overwritten at the next plan cycle. No explicit consistency constraint is enforced between trajectories produced at consecutive plan cycles, some works even claim that adding history trajectories worse the trajectory performance (Cheng et al., 2024b;a).

3.2 CLOSED-LOOP NEIGHBORHOOD CORRECTOR

Building on the froze pipeline in Section 3.1, NAC introduces a closed-loop neighborhood corrector which confines the post-training variable to the closed α-neighborhood of the controller-projected prior action.

Method definition. The corrector δ_ϕ takes the encoder hidden representation h_t and the current ego state x_t as input, then produces a bounded correction in $[-1, 1]^2$, which is formulated as:

$$\delta_\phi : (h_t, x_t) \mapsto [-1, 1]^2, \tag{5}$$

where x_t contains kinematic features, providing a real-time signal of ego-vehicle deviation in addition to h_t. The final action is obtained by:

$$a_t \;=\; a_t^0 + \alpha\, \delta_\phi(h_t, x_t), \qquad \delta_\phi(h_t, x_t) \in [-1, 1]^2, \tag{6}$$

where a_t^0 is normalized to $[-1, 1]^2$ at the controller output to align with δ_ϕ. Under this normalization, α directly measures the granularity of correction relative to the magnitude of the prior action, serving as the radius of the closed α-neighborhood. Hence the trainable corrector at each plan cycle is confined to:

$$\mathcal{T}_\alpha(a_t^0) \;=\; \{a \in \mathcal{A} : \|a - a_t^0\|_\infty \le \alpha\}, \tag{7}$$

Specifically, α acts as a trust-region radius for the corrector: a small α keeps the policy close to the controller-projected prior with predictable improvement, while a larger α enlarges the correction range at the cost of a looser linearization (Appendix A.3).

The entire post-training procedure optimizes only the corrector parameters ϕ on closed-loop reward over this neighborhood, while G_θ are kept frozen.

3.3 ADAPTATION APPROACH ANALYSIS

Setup. The closed-loop driving stack is the composition of three maps,

$$o_t \xrightarrow{G_\theta} (\tau_t^0, h_t) \xrightarrow{C} a_t^0 \xrightarrow{f} s_{t+1}, \tag{8}$$

where G_θ, C, and f respectively denote the frozen pretrained planner, the LQR controller (Sec. 3.1), and the vehicle dynamics. The closed-loop reward $r_t = r(s_t, a_t, s_{t+1})$ is jointly determined by the action a_t that actually enters dynamics and the resulting transition.

Three candidate adaptation approaches. A post-training method can in principle place its trainable variable at one of three positions. The first is the *fine-tuning*, which fine-tunes G_θ directly with optimization variable θ in the planner's full parameter space, corresponding to a trajectory-distribution-level modification. The second is the *trajectory-correction*, which freezes G_θ and learns a trajectory-level perturbation Δ_ϕ^τ:

$$\tilde{\tau}_t = \tau_t^0 + \Delta_\phi^\tau, \qquad a_t = C(\tilde{\tau}_t, s_t). \tag{9}$$

Its optimization variable lives in $\mathbb{R}^{H\times d}$ and must still pass through the controller before entering dynamics. The third is the *action-correction*, which freezes both G_θ and C and learns a bounded correction on top of the controller-projected prior action a_t^0:

$$a_t = a_t^0 + \alpha\, \delta_\phi(h_t, x_t), \qquad \delta_\phi \in [-1, 1]^2. \tag{10}$$

NAC adopts the action-correction approach.

Prior manifold and controller pushforward. Following the manifold view of generative priors over structured demonstrations (Li & He, 2026; Pan et al., 2026), we adopt two assumptions. First, the conditional prior $p_\theta(\cdot \mid h_t)$ is supported on a compact subset $\mathcal{M}_\tau(h_t) \subset \mathbb{R}^{H\times d}$ of effective dimension $\ell \ll H \cdot d$, referred to as the *prior trajectory manifold*. Second, the controller $C(\cdot, s_t)$ is C^1 and L_C-Lipschitz on an open neighborhood of $\mathcal{M}_\tau(h_t)$ (cf. Appendix A.1). The pushforward $\mathcal{M}_a(h_t, s_t) := C\big(\mathcal{M}_\tau(h_t),\, s_t\big) \subset \mathcal{A}$ defines the *prior-induced action set*. The prior action a_t^0 lies in $\mathcal{M}_a$ almost surely, and $\mathcal{T}_\alpha(a_t^0)$ from Section 3.2 sits inside the closed α-tube around $\mathcal{M}_a$. Two lemmas follow. Their proofs are in Appendix A.

***Lemma* 1** (Variance dominance of the action-correction)**.** *Consider the per-step policy-gradient estimator $\hat{g} = \nabla \log \pi(a_t \mid s_t)\, \hat{A}_t$ and, more generally, by any score-function policy-gradient method, where $\hat{A}_t$ is an advantage estimator that depends on the trajectory-correction variable Δ_t^τ only through the executed-action sequence. At a fixed closed-loop step, let trajectory-correction use an isotropic Gaussian policy $\pi_2 = \mathcal{N}(0, \sigma_2^2 I_{H\cdot d})$ on Δ^τ, and let action-correction use a policy π_3 on δ_ϕ matched so that π_2 and π_3 induce the same marginal distribution of a_t. Let $\hat{g}_2 \in \mathbb{R}^{H\cdot d}$ and $\hat{g}_3 \in \mathbb{R}^2$ be the corresponding gradient estimators, and write $J_C = \partial C/\partial \tau$ at (τ_t^0, s_t), with $\ker(J_C)$ denoting the null space of this controller Jacobian. Then $\hat{g}_2$ admits the orthogonal decomposition*

$$\hat{g}_2 \;=\; J_C^\top \hat{g}_3 \;+\; \hat{g}_2^{\ker}, \qquad \hat{g}_2^{\ker} \in \ker(J_C), \tag{11}$$

with $\mathbb{E}[\hat{g}_2^{\ker}] = 0$ and kernel-induced excess variance

$$\mathbb{E}\,\|\hat{g}_2^{\ker}\|_2^2 \;=\; \frac{(H\cdot d - 2)\,\mathbb{E}[\hat{A}_t^2]}{\sigma_2^2} \;>\; 0 \tag{12}$$

whenever $\hat{A}_t$ is non-degenerate. Consequently,

$$\mathrm{Cov}[\hat{g}_2] \;=\; J_C^\top \mathrm{Cov}[\hat{g}_3]\, J_C \;+\; \mathrm{Cov}[\hat{g}_2^{\ker}] \;\succeq\; J_C^\top \mathrm{Cov}[\hat{g}_3]\, J_C \tag{13}$$

in Loewner order on $\mathbb{R}^{H\cdot d}$, with strict dominance whenever $\mathrm{Cov}[\hat{g}_2^{\ker}] \neq 0$. Fine-tuning inherits the same dominance via the additional planner Jacobian $\partial \tau_t^0/\partial\theta$.

Lemma 1 shows that, under the matched marginal on a_t, the trajectory-correction gradient $\hat{g}_2$ admits an orthogonal decomposition $\hat{g}_2 = J_C^\top \hat{g}_3 + \hat{g}_2^{\ker}$, where $\hat{g}_2^{\ker} \in \ker(J_C)$ has zero mean and strictly inflates $\mathrm{Cov}[\hat{g}_2]$ whenever the advantages are non-degenerate. Geometrically, the controller collapses the $H \cdot d$-dimensional trajectory correction onto the two-dimensional executed action, with $\dim\ker(J_C) = H\cdot d - 2 \approx 238$ versus $\dim\mathrm{im}(J_C) = 2$. The kernel components leave a_t unchanged and cancel in expectation while inflating finite-batch variance. Action-correction parameterizes the corrector directly in $\mathrm{im}(J_C)$ where $\partial a_t/\partial\delta_\phi = \alpha I$, and avoids this excess term.

Accordingly, the lemma isolates variance due to parameterization under matched executed-action marginals. Practical corrector training may also depend on learned action distributions and planner-specific optimization configurations. The bounded radius also admits a local first order performance expansion. Appendix A.3 gives the result and its assumptions. The radius ablation in Section 5.2 evaluates the resulting locality trade-off.

3.4 Implementation details

The corrector δ_ϕ uses three 256-dimensional MLP encoders over the planner hidden state h_t, the ego state x_t, and the prior action a_t^0. The actor concatenates the first two embeddings and parameterizes a per-channel Beta distribution over $[-1, 1]^2$ via a $[256, 256]$ head. The critic concatenates all three embeddings with a four-dimensional measurement vector and produces the value V via a $[256, 256, 1]$ head. The correction radius is fixed to $\alpha = 0.3$. Post-training uses distributed PPO on CaRL's online nuPlan architecture (Jaeger et al., 2025) with the native nuPlan closed-loop reward as r_t, GAE (γ=0.99, λ=0.95), and clip $\varepsilon = 0.1$. On flow-matching planner, the corrector adds 5.729M parameters and only 2.331 ms per planning cycle, without another planner query. Training uses 100M environment steps. Full training settings and computational measurements are provided in Appendices B and C.

Table 1: Closed-loop scores on nuPlan val14, test14-hard, and InterPlan10 (*NR* = non-reactive, *R* = reactive). The *Paradigm* column summarizes the training recipe. The *Output* column describes Det. / Gen. = deterministic / generative and traj. / act. = trajectory / action. **Bold** marks the best score per column within the Learning-based group, and shaded cells mark the second best. CorrectionPlanner does not report InterPlan10.

Planner	Paradigm	Output	val14		test14-hard		InterPlan10
			NR	R	NR	R	R
Log-replay (Caesar et al., 2021)	Expert	—	93.53	80.32	85.96	68.80	25.40
IDM (Treiber et al., 2000)	Rule-based	Det., act.	75.60	77.33	56.15	62.26	37.82
PDM-Closed (Dauner et al., 2023)	Rule-based	Det., traj.	92.84	92.12	65.08	75.19	43.61
PDM-Open (Dauner et al., 2023)	IL	Det., traj.	53.53	54.24	33.52	35.83	11.47
UrbanDriver (Scheel et al., 2022)	IL	Det., traj.	68.57	64.11	50.40	49.95	9.98
GameFormer (Huang et al., 2023)	IL	Det., traj.	13.32	8.69	7.08	6.69	6.24
planTF (Cheng et al., 2024b)	IL	Det., traj.	84.27	76.95	69.70	61.61	33.70
PLUTO (Cheng et al., 2024a)	IL	Det., traj.	88.89	78.11	70.03	59.74	22.69
Diffusion-Planner (Zheng et al., 2025)	IL	Gen., traj.	89.87	82.80	75.99	69.22	32.28
Flow-Planner (Tan et al., 2025)	IL	Gen., traj.	90.43	83.31	76.47	70.42	30.54
PlannerRFT (Li et al., 2026)	RL Fine-tune	Gen., traj.	89.96	84.46	77.16	72.21	33.70
Plan-R1 (Tang et al., 2026)	RL Fine-tune	Gen., traj.	88.98	**87.69**	77.45	77.20	40.95
CorrectionPlanner (Guo et al., 2026)	RL Correction	Gen., traj.	91.22	85.19	75.37	77.29	–
NAC (ours)	RL Correction	Gen., act.	**93.28**	87.36	**83.56**	**79.30**	**41.39**

4 EXPERIMENTS

4.1 BENCHMARKS AND BASELINES

Benchmarks. We evaluate NAC on three closed-loop planning benchmarks in the nuPlan family (Caesar et al., 2021): val14 (Dauner et al., 2023), test14-hard (Cheng et al., 2024b), and InterPlan10 (Hallgarten et al., 2024). nuPlan is the first large-scale closed-loop planning benchmark built on real-world urban driving data, organized into 75 representative scenario types, and equipped on the simulation side with both reactive and non-reactive agent settings. Specifically, the three subsets cover nominal driving regimes, long-tail scenes from the nuPlan test split, and programmatically inserted high-interaction corner cases, respectively. InterPlan10 is evaluated without benchmark-specific training or retuning, using the same correction radius $\alpha = 0.3$. Per-subset construction details are deferred to Appendix D.

Baselines. We compare 14 methods across three groups, including Expert, Rule and Learning, following PDM-Closed (Dauner et al., 2023) and the convention adopted by subsequent nuPlan work (Zheng et al., 2025; Tan et al., 2025; Li et al., 2026). The Expert group contains only log-replay, directly replaying human expert driving logs under the current evaluation protocol. The Rule group consists of methods that produce decisions entirely from hand-crafted rules and scores without learning from driving data, including IDM (Treiber et al., 2000) and PDM-Closed (Dauner et al., 2023). The Learning group contains all learnable planners trained directly on driving data, including imitation-learning as well as RL post-training. The Learning group includes PDM-Open (Dauner et al., 2023), UrbanDriver (Scheel et al., 2022), GameFormer (Huang et al., 2023), planTF (Cheng et al., 2024b), PLUTO (Cheng et al., 2024a), Diffusion-Planner (Zheng et al., 2025), Flow-Planner (Tan et al., 2025), PlannerRFT (Li et al., 2026), Plan-R1 (Tang et al., 2026), and CorrectionPlanner (Guo et al., 2026). Per-method descriptions are given in Appendix D.

4.2 MAIN RESULTS

Table 1 summarizes our main experimental results on listed nuPlan splits. Within the Learning-based group, NAC attains the best score on four splits and the second on val14-R, where it trails Plan-R1 by only a fractional margin. Taking the Flow-Planner on which our method depends as the prior, the gains across five splits are uniformly positive and growing with split difficulty.

Table 2: Cross-family applicability and correction-radius analysis of NAC.

Configuration	val14		test14-hard		Interplan10
	NR	R	NR	R	R
Panel A: Cross-family applicability					
Flow-Planner	90.43	83.31	76.47	70.42	30.54
+ NAC	93.28	87.36	83.56	79.30	41.39
Diffusion-Planner	89.87	82.80	75.99	69.22	32.28
+ NAC	88.72	84.66	74.58	71.81	33.52
+ NAC (tuned)	90.78	83.92	76.47	69.88	32.81
Panel B: Correction radius α sweep					
0 (prior)	90.43	83.31	76.47	70.42	30.54
0.1	91.59	86.74	78.87	74.15	32.65
0.2	92.55	86.94	78.31	75.00	35.67
0.3	**93.28**	**87.36**	**83.56**	**79.30**	**41.39**
0.4	91.33	87.50	76.88	70.34	32.98
0.5	90.55	82.73	75.20	68.98	31.88
1.0	88.23	80.32	70.84	67.51	25.95

On the most challenging test14-hard-R and InterPlan10 splits, the improvement of NAC over Flow-Planner ($+8.88$, $+10.85$) is notably larger than on val14-R ($+4.05$). This is consistent with the positioning developed in our methodology. Under imitation learning, the generative planner has already built fairly robust driving behavior over the bulk of the data distribution, so neighborhood correction has limited room to act in nominal regimes. On hard scenarios, however, the generative planner has only partially captured the relevant prior, while distribution shift makes it more vulnerable in closed loop, where prior-anchored neighborhood correction yields directionally clear gains.

PDM-Closed remains a particularly strong baseline on val14-R and InterPlan10, partly reflecting a structural fit between the conservatism preference of the official nuPlan reactive agent and PDM's IDM-based candidate-and-score pipeline. On test14-hard, however, the advantage of NAC over this rule-based baseline indicates that as the closed-loop scenario approaches the realistic long tail, execution-side post-training becomes increasingly favorable relative to a heavily tuned rule-based pipeline. Qualitative comparisons illustrating well-timed avoidance, prompt initiation, in-lane cornering, and emergency yielding are provided in Appendix I.

5 Ablation Studies

5.1 Does the recipe generalize across generative planner families?

We investigate whether NAC applies across generative planner families rather than only to Flow-Planner. We run NAC post-training pipeline with identical hyperparameters on top of Diffusion-Planner and Flow-Planner, then evaluate on the same nuPlan splits in Table 2, Panel A.

NAC delivers consistent gains across all four splits on Flow-Planner. For Diffusion-Planner, the same recipe improves the reactive splits but not the non-reactive ones. With light planner-specific adjustment of the actor learning rate lr $= 2.5$e-4 and PPO trust region clip $\epsilon = 0.05$, it improves all splits, as shown by the tuned row in Table 2, Panel A. This contrast supports our claim at two levels. First, the bounded-correction approach on the executed-action neighborhood produces non-trivial gains on two distinct prior families, rather than being tied to a particular prior manifold. Second, the magnitude and split distribution of the gains depend on the shape of the prior and on its coupling with the controller. These results support cross-family applicability with light planner-specific tuning. Additional evaluations with planTF and the pre-RL Plan-R1 checkpoint are reported in Appendix E.

5.2 How local should the executed-action neighborhood be?

The radius α in NAC's correction $a_t = a_t^0 + \alpha\,\delta_\phi(h_t, x_t)$ instantiates the local prior-anchored design assumption: too small and the anchor caps correction capacity, too large and the method drifts off

Table 3: Empirical comparison of the three post-training approaches from Section 3.3 on the Flow-Planner prior. **Bold** marks the best score per column among RL-trained variants.

Post-training	Method	val14		test14-hard		Interplan10
		NR	R	NR	R	R
Flow-Planner (prior)		90.43	83.31	76.47	70.42	30.54
Fine-tuning	PPO	89.63	83.40	77.64	71.63	32.38
	SAC	63.07	60.02	51.85	53.87	–
	TD3	60.28	55.98	55.19	48.16	–
Trajectory-correction	PPO	89.90	86.39	79.71	77.36	34.33
Action-correction	PPO(NAC)	**93.28**	**87.36**	**83.56**	**79.30**	**41.39**

the prior. We sweep $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 1.0\}$ while holding all other settings fixed in Table 2, Panel B.

The remaining settings shows two failure modes of locality assumption. A tighter neighborhood narrows an overly small anchor caps correction capacity, instead a wider neighborhoods weakens the anchor and let RL drift off the prior. This brings out a U-shape with the optimum in the middle, mirroring the bounded-locality assumption of Section 3.

5.3 Empirical comparison of post-training approaches

Section 3.3 identified three candidate approaches for closed-loop post-training: fine-tuning, trajectory-correction and action-correction. We compare all three on the same settings in Table 3. The fine-tuning baselines apply three RL recipes, including PPO (Schulman et al., 2017) with an FPO surrogate (McAllister et al., 2026), SAC (Haarnoja et al., 2018), and TD3 (Fujimoto et al., 2018), to the Flow-Planner decoder while freezing the encoder. The trajectory-correction baseline places the corrector at trajectory output in NAC. Full settings are deferred to Appendix G.

The comparison supports our claim at two levels. First, direct RL fine-tuning collapses the driving prior under our training budget: SAC and TD3 fall far below the prior on every split, while even on-policy PPO can barely surpass the prior. Second, both correction clearly outperform direct fine-tuning, confirming that a bounded correction on a frozen planner is more reliable. Within the correction family, action-correction further outperforms trajectory-correction on every column. These gains are consistent with Lemma 1, but because the matched marginal is not enforced between the independently trained correctors, the empirical gap may also reflect their learned action distributions and closed-loop optimization. An auxiliary projection analysis of how trajectory manifold adherence relates to task improvement is reported in Appendix H.

6 Conclusion

We presented NAC, a closed-loop neighborhood post-training framework which keeps a generative driving planner frozen and learns only a bounded correction within an α-neighborhood of its prior action. This preserves the pretrained prior manifold by construction, while allowing reward to supervise the exact variable entering vehicle dynamics. NAC achieves the best closed-loop score among learning-based planners on nuPlan, with the largest gains on long-tail splits.

Future work will explore richer corrector conditioning and broader planner classes. In particular, agent-wise cross-attention over neighboring agents and lane tokens could augment the pooled planner representation while retaining a two-dimensional correction output. Multimodal scene descriptions may further provide semantic cues for long-tail situations that are difficult to capture from geometric features alone. Beyond the planner backbones considered here, the same post-training principle could be applied to sensor-to-trajectory driving foundation models by correcting their controller-projected actions while keeping the pretrained representations frozen.

AI USE STATEMENT

Generative AI tools were used to improve wording and grammar, assist with literature and reference checks, and prepare and organize the LaTeX manuscript. AI assistance also informed the presentation of existing claims and results. Generative AI was not used to develop the research method, derive theoretical results, implement the research code, generate experimental data, or run experiments. The authors take responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

The main paper specifies the model, training objective, evaluation protocol, and primary hyperparameters. Appendices B and C provide complete implementation settings and computational measurements. Appendices D and G document the benchmarks, baseline definitions, and training configurations. Appendix A gives full proofs for the theoretical claims.

REFERENCES

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles. In *CVPR ADP3 workshop*, 2021. URL https://arxiv.org/abs/2106.11810.

Jie Cheng, Yingbing Chen, and Qifeng Chen. PLUTO: Pushing the Limit of Imitation Learning-based Planning for Autonomous Driving. arXiv preprint arXiv:2404.14327, April 2024a.

Jie Cheng, Yingbing Chen, Xiaodong Mei, Bowen Yang, Bo Li, and Ming Liu. Rethinking Imitation-based Planners for Autonomous Driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14123–14130. IEEE, May 2024b. doi: 10.1109/ICRA57147.2024.10611364.

Marco Cusumano-Towner, David Hafner, Alexander Hertzberg, Brody Huval, Aleksei Petrenko, Eugene Vinitsky, Erik Wijmans, Taylor W. Killian, Stuart Bowers, Ozan Sener, Philipp Kraehenbuehl, and Vladlen Koltun. Robust Autonomy Emerges from Self-Play. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 11710–11737. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/cusumano-towner25a.html.

Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with Misconceptions about Learning-based Vehicle Motion Planning. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1268–1281. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/dauner23a.html.

Lakshita Dodeja, Karl Schmeckpeper, Shivam Vats, Thomas Weng, Mingxi Jia, George Konidaris, and Stefanie Tellex. Accelerating Residual Reinforcement Learning with Uncertainty Estimation. *IEEE Robotics and Automation Letters*, 11(1):970–977, January 2026. doi: 10.1109/LRA.2025.3636808.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. EXPO: Stable Reinforcement Learning with Expressive Policies. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=aFjSjkB6CV.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Hao Gao, Shaoyu Chen, Yifan Zhu, Yuehao Song, Wenyu Liu, Qian Zhang, and Xinggang Wang. RAD-2: Scaling Reinforcement Learning in a Generator-Discriminator Framework. arXiv preprint arXiv:2604.15308, April 2026.

Yihong Guo, Dongqiangzi Ye, Sijia Chen, Anqi Liu, and Xianming Liu. CorrectionPlanner: Self-Correction Planner with Reinforcement Learning in Autonomous Driving. arXiv preprint arXiv:2603.15771, March 2026.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Marcel Hallgarten, Julian Zapata, Martin Stoll, Katrin Renz, and Andreas Zell. Can Vehicle Motion Planning Generalize to Realistic Long-tail Scenarios? In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5388–5395. IEEE, October 2024. doi: 10.1109/IROS58592.2024.10803052. URL https://arxiv.org/abs/2404.07569.

Zhiyu Huang, Haochen Liu, and Chen Lv. GameFormer: Game-theoretic Modeling and Learning of Transformer-based Interactive Prediction and Planning for Autonomous Driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3903–3913, October 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Huang_GameFormer_Game-theoretic_Modeling_and_Learning_of_Transformer-based_Interactive_Prediction_and_ICCV_2023_paper.html.

Zhiyu Huang, Xinshuo Weng, Maximilian Igl, Yuxiao Chen, Yulong Cao, Boris Ivanovic, Marco Pavone, and Chen Lv. Gen-Drive: Enhancing Diffusion Generative Driving Policies with Reward Modeling and Reinforcement Learning Fine-tuning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3445–3451. IEEE, May 2025. doi: 10.1109/ICRA55743.2025.11127286.

Bernhard Jaeger, Daniel Dauner, Jens Beißwenger, Simon Gerstenecker, Kashyap Chitta, and Andreas Geiger. CaRL: Learning Scalable Planning Policies with Simple Rewards. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 5301–5338. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/jaeger25a.html.

Xuefeng Jiang, Yuan Ma, Pengxiang Li, Leimeng Xu, Xin Wen, Kun Zhan, Zhongpu Xia, Peng Jia, Xianpeng Lang, and Sheng Sun. TransDiffuser: Diverse Trajectory Generation with Decorrelated Multi-modal Representation for End-to-end Autonomous Driving. arXiv preprint arXiv:2505.09315, May 2025.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual Reinforcement Learning for Robot Control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029. IEEE, May 2019. doi: 10.1109/ICRA.2019.8794127.

Sham M. Kakade and John Langford. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML)*, pp. 267–274, 2002. URL https://mlanthology.org/icml/2002/kakade2002icml-approximately/.

Hongchen Li, Tianyu Li, Jiazhi Yang, Mingyang Shang, Gaoqiang Wu, Caojun Wang, Haochen Tian, Zengrong Lin, Zhihui Hao, XianPeng Lang, Jia Hu, and Hongyang Li. PlannerRFT: Reinforcing Diffusion Planners through Closed-Loop and Sample-Efficient Fine-Tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24929–24938, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Li_PlannerRFT_Reinforcing_Diffusion_Planners_through_Closed-Loop_and_Sample-Efficient_Fine-Tuning_CVPR_2026_paper.html.

Qingpeng Li, Chengrui Zhu, Yanming Wu, Xin Yuan, Zhen Zhang, Jian Yang, and Yong Liu. RuN: Residual Policy for Natural Humanoid Locomotion. arXiv preprint arXiv:2509.20696, September 2025.

Tianhong Li and Kaiming He. Back to Basics: Let Denoising Generative Models Denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 36115–36125, June 2026. URL https://arxiv.org/abs/2511.13720.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12037–12047, June 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liao_DiffusionDrive_Truncated_Diffusion_Model_for_End-to-End_Autonomous_Driving_CVPR_2025_paper.html.

Lin Liu, Caiyan Jia, Guanyi Yu, Ziying Song, Junqiao Li, Feiyang Jia, Peiliang Wu, Xiaoshuai Hao, and Yadan Luo. GuideFlow: Constraint-Guided Flow Matching for Planning in End-to-End Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3719–3728, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Liu_GuideFlow_Constraint-Guided_Flow_Matching_for_Planning_in_End-to-End_Autonomous_Driving_CVPR_2026_paper.html.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow Matching Policy Gradients. In *International Conference on Learning Representations*, pp. 36352–36372, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/3d43cc5692bf68944ee7cd31b97d0c11-Paper-Conference.pdf.

Chaoyi Pan, Giridharan Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much Ado About Noising: Dispelling the Myths of Generative Robotic Control. In *International Conference on Learning Representations*, pp. 90575–90614, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/9299ca9ed58731945e934adb5b71728c-Paper-Conference.pdf.

Oliver Scheel, Luca Bergamini, Maciej Wołczyk, Błażej Osiński, and Peter Ondruska. Urban Driver: Learning to Drive from Real-world Demonstrations Using Policy Gradients. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 718–728. PMLR, 2022. URL https://proceedings.mlr.press/v164/scheel22a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347, July 2017.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual Policy Learning. arXiv preprint arXiv:1812.06298, December 2018.

Tianyi Tan, Yinan Zheng, Ruiming Liang, Zexu Wang, Kexin Zheng, Jinliang Zheng, Jianxiong Li, Xianyuan Zhan, and Jingjing Liu. Flow Matching-Based Autonomous Driving Planning with Advanced Interactive Behavior Modeling. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 38310–38335. Curran Associates, Inc., 2025. doi: 10.52202/085713-1281. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/36d1e8aa9ceec3b781682bf5e63c31bf-Paper-Conference.pdf.

Xiaolong Tang, Meina Kan, Shiguang Shan, and Xilin Chen. Plan-R1: Safe and Feasible Trajectory Planning as Language Modeling. In *International Conference on Learning Representations*, pp. 23715–23735, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/28a80012e6f564c0cc8e7661e1db83fe-Paper-Conference.pdf.

Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824, August 2000. doi: 10.1103/PhysRevE.62.1805.

Lingguang Wang, Ömer Şahin Taş, Marlon Steiner, and Christoph Stiller. FlowDrive: moderated flow matching with data balancing for trajectory planning. arXiv preprint arXiv:2509.21961, September 2025.

Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Jain, Jeff Schneider, and Katerina Fragkiadaki. Diffusion-ES: Gradient-free Planning with Diffusion for Autonomous and Instruction-guided Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15342–15353, June 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Yang_Diffusion-ES_Gradient-free_Planning_with_Diffusion_for_Autonomous_and_Instruction-guided_Driving_CVPR_2024_paper.html.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy Decorator: Model-Agnostic Online Refinement for Large Policy Model. In *International Conference on Learning Representations*, pp. 28129–28164, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/45c361d4117d598d4bb6568b407e9ac9-Paper-Conference.pdf.

Dongkun Zhang, Jiaming Liang, Ke Guo, Sha Lu, Qi Wang, Rong Xiong, Zhenwei Miao, and Yue Wang. CarPlanner: Consistent Auto-regressive Trajectory Planning for Large-scale Reinforcement Learning in Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17239–17248, June 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Zhang_CarPlanner_Consistent_Auto-regressive_Trajectory_Planning_for_Large-Scale_Reinforcement_Learning_in_CVPR_2025_paper.html.

Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-End Urban Driving by Imitating a Reinforcement Learning Coach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15222–15232, October 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Zhang_End-to-End_Urban_Driving_by_Imitating_a_Reinforcement_Learning_Coach_ICCV_2021_paper.html.

Yinan Zheng, Ruiming Liang, Kexin Zheng, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Diffusion-Based Planning for Autonomous Driving with Flexible Guidance. In *International Conference on Learning Representations*, pp. 37207–37227, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5c0cd43063f087eb32e572d040a19cd2-Paper-Conference.pdf.

Yinan Zheng, Tianyi Tan, Bin Huang, Enguang Liu, Ruiming Liang, Jianlin Zhang, Jianwei Cui, Guang Chen, Kun Ma, Hangjun Ye, Long Chen, Ya-Qin Zhang, Xianyuan Zhan, and Jingjing Liu. Unleashing the Potential of Diffusion Models for End-to-End Autonomous Driving. arXiv preprint arXiv:2602.22801, February 2026a.

Zhiyu Zheng, Shaoyu Chen, Haoran Yin, Xinbang Zhang, Jialv Zou, Xinggang Wang, Qian Zhang, and Lefei Zhang. ResAD: Normalized Residual Trajectory Modeling for End-to-End Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3729–3739, June 2026b. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zheng_ResAD_Normalized_Residual_Trajectory_Modeling_for_End-to-End_Autonomous_Driving_CVPR_2026_paper.html.

A PROOFS FOR SECTION 3.3

This appendix collects the proofs of the two lemmas stated in Section 3.3, together with a smoothness justification of the LQR controller that underpins assumption (A2). We keep the notation of the main text.

A.1 SMOOTHNESS OF THE LQR CONTROLLER

The LQR block $C(\tau, s_t)$ used inside the nuPlan simulator admits an analytical one-shot solution (Sec. 3.1). Starting from a reference trajectory $\tau \in \mathbb{R}^{H\times d}$ and the current ego state s_t, the controller (i) interpolates τ at a fixed $0.1\,\mathrm{s}$ grid to obtain a dense pose sequence, (ii) recovers a reference velocity profile and a reference curvature profile through two ridge-regularized least-squares fits (jerk penalty 10^{-4}, curvature-rate penalty 10^{-2}), (iii) solves a one-step scalar LQR for the longitudinal command and a one-step time-varying lateral LQR for the steering-rate command using the recovered profiles, and (iv) outputs $a_t^0 = C(\tau, s_t) \in \mathbb{R}^2$.

Each operation is either linear in τ or a closed-form rational map whose coefficients depend smoothly on s_t, the fixed LQR cost matrices, and the kinematic constants of the vehicle. Away from the stopping branch ($v_0 \le 0.2\,\mathrm{m/s}$ and $v_{\mathrm{ref}} \le 0.2\,\mathrm{m/s}$), the map $\tau \mapsto C(\tau, s_t)$ is C^∞ on the open neighborhood of $\mathcal{M}_\tau(h_t)$ implicit in assumption (A2). Composing the moduli of the individual operations yields a finite Lipschitz constant $L_C = L_C(s_t)$ that remains uniformly bounded on any ego-speed interval $[v_{\min}, v_{\max}]$ with $v_{\min} > 0.2\,\mathrm{m/s}$. On the stopping branch, C reduces to the proportional controller $a^\star = -k_p(v_0 - v_{\mathrm{ref}})$ with $\dot\delta^\star = 0$, which is C^∞ in τ (indeed, it depends on τ only through the interpolated v_{ref}) and Lipschitz. Hence (A2) holds on the entire operating regime.

A.2 PROOF OF LEMMA 1

Fix (o_t, h_t, s_t) and let $J_C = \partial C/\partial \tau$ at (τ_t^0, s_t). Let $\Pi_{\mathrm{im}}, \Pi_{\mathrm{ker}}$ denote the orthogonal projections onto $\mathrm{im}(J_C^\top)$ and $\ker(J_C)$ respectively.

Advantage depends on a_t only. Under receding-horizon overwriting (Sec. 3.1), the current-step trajectory variable Δ_t^τ enters r_t and all downstream states only through a_t: once a_t is fixed, $s_{t+1} = f(s_t, a_t)$ is determined, the next observation is determined, and G_θ then produces τ_{t+1}^0 from the new observation, irrespective of the kernel components of Δ_t^τ. Hence any advantage estimator $\hat A_t$ formed from $\{r_\tau, s_\tau, V(s_\tau)\}_{\tau\ge t}$, including the GAE estimator used by PPO, satisfies $\hat A_t = \hat A(a_t)$ for some measurable $\hat A$, after marginalizing the future policy randomness.

Score decomposition and kernel-component statistics. The score $s_2 = \Delta^\tau/\sigma_2^2$ decomposes as $\Pi_{\mathrm{im}} s_2 + \Pi_{\mathrm{ker}} s_2$, with the two components independent and zero-mean by isotropy of π_2. Linearization of C around τ_t^0 (justified by (A2) at scale $\sigma_2 \to 0$) gives $a_t = a_t^0 + J_C \Pi_{\mathrm{im}} \Delta^\tau + o(\|\Delta^\tau\|)$, so a_t is independent of $\Pi_{\mathrm{ker}}\Delta^\tau$ to leading order. Define $\hat g_2^{\mathrm{ker}} := \Pi_{\mathrm{ker}} s_2 \cdot \hat A_t$. Independence of $\Pi_{\mathrm{ker}}\Delta_t^\tau$ and a_t (hence of $\hat A_t$) gives $\mathbb{E}[\hat g_2^{\mathrm{ker}}] = 0$, and using $\mathbb{E}[(\Pi_{\mathrm{ker}} s_2)(\Pi_{\mathrm{ker}} s_2)^\top] = \Pi_{\mathrm{ker}}/\sigma_2^2$,

$$\mathrm{Cov}[\hat g_2^{\mathrm{ker}}] = \frac{\mathbb{E}[\hat A_t^2]}{\sigma_2^2}\,\Pi_{\mathrm{ker}}, \qquad \mathbb{E}\,\|\hat g_2^{\mathrm{ker}}\|_2^2 = \mathrm{tr}\big(\mathrm{Cov}[\hat g_2^{\mathrm{ker}}]\big) = \frac{(H\cdot d - 2)\,\mathbb{E}[\hat A_t^2]}{\sigma_2^2}.$$

Image component equals the pulled-back $\hat g_3$. The matched policy π_3 is the J_C-pushforward of π_2, so $\pi_3 = \mathcal{N}(0, \sigma_2^2 J_C J_C^\top)$ and $\hat g_3 = (\sigma_2^2 J_C J_C^\top)^{-1}\delta_\phi \cdot \hat A_t$ with $\delta_\phi = J_C \Pi_{\mathrm{im}} \Delta^\tau$ to leading order. Using the Moore–Penrose identity $J_C^\top (J_C J_C^\top)^{-1} J_C = \Pi_{\mathrm{im}}$,

$$J_C^\top \hat g_3 = \tfrac{1}{\sigma_2^2}\,\Pi_{\mathrm{im}}\Delta^\tau \cdot \hat A_t = \Pi_{\mathrm{im}} \hat g_2,$$

so $\hat g_2 = J_C^\top \hat g_3 + \hat g_2^{\mathrm{ker}}$ orthogonally. The two summands are uncorrelated because $\hat g_2^{\mathrm{ker}}$ depends on $\Pi_{\mathrm{ker}}\Delta_t^\tau$, which is independent of a_t (and hence of $\hat A_t$ and $\hat g_3$), giving

$$\mathrm{Cov}[\hat g_2] = J_C^\top\,\mathrm{Cov}[\hat g_3]\,J_C + \mathrm{Cov}[\hat g_2^{\mathrm{ker}}] \succeq J_C^\top\,\mathrm{Cov}[\hat g_3]\,J_C,$$

with strict Loewner dominance whenever $\hat{A}_t$ is non-degenerate. The fine-tuning estimator $\hat{g}_1 = (\partial\tau_t^0/\partial\theta)^\top \hat{g}_2$ inherits the same dominance by left-multiplying through $(\partial\tau_t^0/\partial\theta)^\top$. The same conclusion holds for the PPO surrogate: the clipped objective $\min(r_t\hat{A}_t, \mathrm{clip}(r_t, 1\pm\epsilon)\hat{A}_t)$ produces, at every sample, a gradient of the form $w_t \cdot \nabla \log \pi \cdot \hat{A}_t$ with a nonnegative weight $w_t \in \{0, 1, r_t\}$, which commutes with the image/kernel decomposition above and preserves the kernel-induced excess variance. □

A.3 First order analysis of the correction radius

Lemma 2 (First-order improvement on the α-tube, via PDL)**.** *Let the prior policy be the deterministic map $\pi_0(s_t) \equiv a_t^0$ with $a_t^0 \in \mathcal{M}_a(h_t, s_t)$, and the adapted policy be $\pi_\phi(s_t) \equiv a_t^0 + \alpha\,\delta_\phi(h_t, x_t)$ with $\delta_\phi \in [-1, 1]^2$. Assume the prior-policy action-value $Q^{\pi_0}(s, a)$ is C^2 on $\mathcal{T}_\alpha(a_t^0)$. Then by the Performance Difference Lemma (Kakade & Langford, 2002),*

$$J(\pi_\phi) - J(\pi_0) \;=\; \alpha\,\mathbb{E}_{\pi_0}\Big[\textstyle\sum_t \gamma^t\,\nabla_a Q^{\pi_0}(s_t, a_t^0)^\top \delta_\phi(h_t, x_t)\Big] \;+\; \mathcal{O}(\alpha^2). \tag{14}$$

Two readings follow. The $\|\cdot\|_\infty$ constraint makes the first-order optimum saturating, with $\delta_\phi^\star(h_t, x_t) = \mathrm{sign}(\nabla_a Q^{\pi_0})$ per channel, consistent with the empirical convergence of PPO under bounded action distributions. The $\mathcal{O}(\alpha^2)$ remainder identifies α as the maximal step size at which linearization remains valid, giving the radius ablation in Section 5.2 a principled reading: small α caps the first-order gain, while large α lets the remainder dominate.

Proof. The Performance Difference Lemma (Kakade & Langford, 2002) gives

$$J(\pi_\phi) - J(\pi_0) \;=\; \mathbb{E}_{\pi_\phi}\Big[\textstyle\sum_t \gamma^t\,A^{\pi_0}(s_t, a_t)\Big],$$

with $A^{\pi_0} = Q^{\pi_0} - V^{\pi_0}$ and $a_t = a_t^0 + \alpha\,\delta_\phi(h_t, x_t)$. Since π_0 is deterministic, $A^{\pi_0}(s, a_t^0(s)) = 0$. A second-order Taylor expansion of $Q^{\pi_0}(s_t, \cdot)$ at a_t^0 along the segment $[a_t^0, a_t^0 + \alpha\delta_\phi] \subset \mathcal{T}_\alpha(a_t^0)$ yields

$$A^{\pi_0}(s_t, a_t^0 + \alpha\delta_\phi) \;=\; \alpha\,\nabla_a Q^{\pi_0}(s_t, a_t^0)^\top\delta_\phi \;+\; \tfrac{\alpha^2}{2}\,\delta_\phi^\top \nabla_a^2 Q^{\pi_0}(s_t, \xi_t)\,\delta_\phi,$$

with ξ_t on the segment. The C^2 assumption on $\mathcal{T}_\alpha(a_t^0)$ bounds $\|\nabla_a^2 Q^{\pi_0}\|$ uniformly, and $\|\delta_\phi\|_\infty \le 1$ gives a Hessian contribution of order α^2. Moreover, the trajectory distributions under π_ϕ and π_0 differ by $\mathcal{O}(\alpha)$ in total variation since $\|a_t - a_t^0\|_\infty \le \alpha$ on $\mathcal{T}_\alpha(a_t^0)$, which combines with the leading $\mathcal{O}(\alpha)$ advantage term to contribute an additional $\mathcal{O}(\alpha^2)$ remainder. Summing these bounds yields

$$J(\pi_\phi) - J(\pi_0) \;=\; \alpha\,\mathbb{E}_{\pi_0}\Big[\textstyle\sum_t \gamma^t\,\nabla_a Q^{\pi_0}(s_t, a_t^0)^\top\delta_\phi(h_t, x_t)\Big] \;+\; \mathcal{O}(\alpha^2). \quad \square$$

B Implementation details: full settings

This appendix gives the full architectural, optimization, and hardware settings for NAC that are summarized in Section 3.4. The corrector is anchored on the frozen Flow-Planner prior, and only the corrector parameters are updated during post-training.

Architecture. The three input encoders, the actor head, and the critic head are all MLPs with LayerNorm and ReLU activations. The actor concatenates the embeddings of the planner hidden state h_t and the ego state x_t, and emits the per-channel Beta parameters of the bounded correction over $[-1, 1]^2$. The critic concatenates the embeddings of h_t, x_t, and the prior action a_t^0 together with a four-dimensional measurement vector, and outputs the scalar value V. The correction radius $\alpha = 0.3$ is held fixed across all main results.

Reinforcement learning. The closed-loop reward r_t is the native nuPlan reward without additional shaping. Advantages are computed via GAE ($\gamma = 0.99$, $\lambda = 0.95$) with per-minibatch normalization. Each rollout is reused for two minibatch updates with target-KL early stopping at $D_{\mathrm{KL}}^\star = 0.015$. The overall loss is $\mathcal{L} = \mathcal{L}_{\mathrm{PG}} + c_V\mathcal{L}_V$ with clip $\varepsilon = 0.1$ and $c_V = 0.5$.

Table 4: Model size and inference latency. A leading "+" denotes the overhead added by NAC.

Method	Params (M)	Latency / plan cycle (ms)
PDM-Closed	n/a	104
Flow-Planner	1.244	24.580
Flow-Planner + NAC	+5.729	+2.331
Diffusion-Planner	6.043	401.697
Diffusion-Planner + NAC	+5.594	+2.340

Optimizer and hardware. Optimization uses Adam with learning rate 2.5×10^{-4}, linear decay, and gradient clipping at 0.5. Training runs on eight NVIDIA A100 GPUs with 128 vectorized environments per worker for a total of 100M environment steps.

C Computational efficiency

Model size. The corrector projects the flattened planner hidden features into 256-dimensional embeddings and uses a $[256, 256]$ MLP actor head. It adds 5.729M parameters to Flow-Planner and 5.594M to Diffusion-Planner, with most parameters arising from the direct feature projections.

Training state. Only the corrector is optimized during post-training. The prior planner remains frozen and therefore requires no gradient or optimizer state, although its forward pass remains part of closed-loop rollout collection. Changing the prior planner affects rollout latency but not which parameters are updated.

Inference overhead. The corrector reuses the hidden state produced by the planner and requires only one additional feed-forward pass, without another planner query or flow/diffusion sampling procedure. It adds 11.45 MFLOPs, equivalent to 1.17% of the Flow-Planner compute, and approximately 2.34 ms per planning cycle. The LQR projection is already part of the standard nuPlan execution stack and is not an additional NAC operation. Table 4 reports the planner-wise parameter and latency measurements.

Latency across planners. The base planner dominates the end-to-end planning latency: Diffusion-Planner incurs a substantially longer sampling chain than Flow-Planner, whereas the added NAC latency remains nearly constant across the two priors (2.331 ms versus 2.340 ms).

D Benchmark and baseline details

This appendix gives the full descriptions of the three nuPlan-family benchmarks and the twelve learnable / rule-based baselines used in Section 4.1.

Benchmarks.

- **val14** follows PDM-Closed (Dauner et al., 2023) and assembles a standard evaluation subset by sampling 100 scenarios from each of 14 scenario types on the official nuPlan validation set, measuring closed-loop quality on nominal driving regimes.
- **test14-hard** is constructed by planTF (Cheng et al., 2024b) on the nuPlan test split: using PDM-Closed (Dauner et al., 2023) closed-loop scores over 100 scenarios per type as the reference, the 20 lowest-scoring scenarios in each type are concatenated into a hard subset that emphasizes long-tail scenes with dense interaction, large maneuvers, and small tolerance margins.
- **InterPlan10** (Hallgarten et al., 2024) is a long-tail evaluation benchmark built on top of nuPlan that programmatically inserts high-interaction corner cases such as jaywalkers, construction zones, and long-time parked-vehicle detours into real driving logs, probing planner generalization beyond the training distribution. We use its default 10-scenario configuration.

Baselines.

- **IDM** (Treiber et al., 2000) is a classical rule-based car-following controller that determines longitudinal acceleration through an analytical formula and follows the lane centerline geometrically in the lateral channel.
- **PDM-Closed** (Dauner et al., 2023) extends IDM by generating multiple candidate trajectories at different target speeds, performing a short closed-loop rollout for each, and selecting the best by a hand-crafted scoring function. It is currently the strongest pure-rule baseline on nuPlan.
- **PDM-Open** (Dauner et al., 2023) is the open-loop variant of PDM that uses IDM candidate trajectories as templates and trains a single MLP under pure open-loop supervised regression, without any pretraining or RL stage.
- **UrbanDriver** (Scheel et al., 2022) relies on a differentiable closed-loop simulator and minimizes deviation from expert trajectories under a policy-gradient formulation on expert logs. The supervision signal remains imitation-style rather than reward-based, and the policy is trained end-to-end from random initialization in a single stage.
- **GameFormer** (Huang et al., 2023) jointly models multi-agent interaction prediction and trajectory output through level-k game-theoretic reasoning, trained end-to-end on nuPlan data with supervised losses.
- **planTF** (Cheng et al., 2024b) combines a transformer backbone with systematic data augmentation to suppress covariate shift, becoming a strong imitation baseline trained from scratch with pure IL.
- **PLUTO** (Cheng et al., 2024a) adopts the same IL training paradigm and further introduces contrastive learning together with several auxiliary tasks, pushing the upper bound achievable by imitation a step further.
- **Diffusion-Planner** (Zheng et al., 2025) and **Flow-Planner** (Tan et al., 2025) are state-of-the-art generative driving planners based on diffusion and flow matching, respectively. Both are trained from scratch with pure IL using their respective score or flow-matching losses, and serve as the most direct prior counterparts of NAC.
- **CorrectionPlanner** (Guo et al., 2026) is an autoregressive planner trained with imitation learning and model-based reinforcement learning that performs correction in motion-token space through a propose–evaluate–correct loop.
- **PlannerRFT** (Li et al., 2026) represents the trajectory-side closed-loop post-training route, applying sample-efficient closed-loop GRPO fine-tuning to the denoising process of a pretrained diffusion planner.
- **Plan-R1** (Tang et al., 2026) formulates trajectory planning as language modeling over trajectory tokens: it first does IL pretraining on expert trajectories, then performs GRPO post-training using rule-based safety and feasibility rewards.

E TRANSFER TO ADDITIONAL PLANNER PRIORS

Table 5 reports NAC on planTF and the released pre-RL checkpoint of Plan-R1. planTF addresses covariate shift during imitation training through data perturbation, augmentation, and feature design, whereas NAC freezes the pretrained planner and learns an execution-side action correction from closed-loop reward. The actor learning rate and PPO clip are adjusted per planner while the normalized correction radius remains $\alpha = 0.3$. The Plan-R1 paper reports its pretraining-only checkpoint on a different split. We therefore evaluate the released checkpoint under the same settings used here.

F DETAILED CLOSED-LOOP METRICS

We follow the standard nuPlan closed-loop evaluation metrics introduced by Caesar et al. (2021). Table 6 reports the complete metric breakdown for NAC, while Table 7 provides a planner-level comparison on val14-NR.

Table 5: Transfer of NAC to planTF and the pre-RL Plan-R1 checkpoint.

Planner	val14 NR	val14 R	test14-hard NR	test14-hard R	InterPlan10 R
planTF	84.27	76.95	69.70	61.61	33.70
+ NAC (LR 2.5×10^{-4}, clip 0.15)	89.14	81.22	77.03	67.96	36.42
Plan-R1 (pre-trained)	85.50	80.69	72.15	65.38	42.85
+ NAC (LR 1×10^{-4}, clip 0.1)	90.63	87.41	79.49	75.19	42.38
Plan-R1 (full)	88.98	87.69	77.45	77.20	40.95

Table 6: Per-metric results of NAC across the five evaluation settings.

Setting	Score	Collision-free	TTC	Drivable	Comfort	Progress	Speed	Direction	Making progress
val14 NR	93.28	96.91	92.75	99.64	96.96	97.56	96.50	99.28	99.91
val14 R	87.36	94.72	90.43	99.37	90.34	88.71	97.98	99.55	99.02
test14-hard NR	83.56	91.91	82.72	98.16	95.96	94.68	95.14	97.79	99.63
test14-hard R	79.30	92.46	85.66	98.16	87.50	81.45	97.77	98.53	96.32
InterPlan10 R	41.39	82.50	75.00	85.00	68.75	56.82	99.22	95.63	93.75

Table 7: Detailed val14-NR comparison following the component-metric reporting protocol used by CorrectionPlanner. The reported aggregate scores here do not replace the main-table evaluation entries.

Planner	Score	Collision-free	TTC	Drivable	Comfort	Progress	Speed
planTF (Cheng et al., 2024b)	85.30	94.13	90.73	96.79	93.67	89.83	97.78
PLUTO (Cheng et al., 2024a)	89.04	96.18	93.28	98.53	96.41	89.56	98.13
CorrectionPlanner (Guo et al., 2026)	91.22	97.96	92.25	98.76	99.31	91.49	97.49
NAC (ours)	93.28	96.91	92.75	99.64	96.96	97.56	96.50

G POST-TRAINING APPROACH BASELINES: DETAILS

This appendix gives the full training and architectural settings for the fine-tuning (PPO / SAC / TD3) and trajectory-correction baselines compared against NAC in Section 5.3. All three fine-tuning baselines share the same closed-loop reward and reward shaping as NAC. Only the optimization variable and update rule differ.

PPO with flow-policy optimization (FPO). The PPO baseline freezes the Flow-Planner scene encoder and updates only the decoder's route encoder, trajectory pre-projection, time embedder, and MLP head. At each rollout step the actor draws a future trajectory by integrating the conditional flow-matching decoder for $K{=}8$ Euler steps from Gaussian noise. Trajectories are represented internally as (x, y, θ) over four 0.5 s waypoints and converted to $(x, y, \cos\theta, \sin\theta)$ before execution. Because the flow policy has no tractable action density, we replace the PPO log-prob ratio with an FPO surrogate (McAllister et al., 2026). For every stored trajectory x_1 we sample $M{=}4$ flow times t and noise vectors x_0, form interpolated states $x_t = (1-t)\,x_0 + t\,x_1$, and compute the conditional flow-matching MSE under the current policy as a negative-log-likelihood-style energy $\ell(x_1)$. The policy ratio used in the clipped PPO objective is then $\exp\big(\ell_{\text{old}}(x_1) - \ell_{\text{new}}(x_1)\big)$ with clip range $\epsilon{=}0.1$. Advantages are estimated by GAE, and an independent two-layer 256-d MLP value head consumes the pooled scene encoding, route encoding, ego state, and a 4-d measurement vector. Stochasticity comes from the rollout noise and the FPO Monte-Carlo time/noise samples. No explicit entropy bonus is used.

SAC route50. The SAC baseline (*route50*) combines the same frozen-encoder Flow-Planner actor with an off-policy Twin Q critic that augments the standard scene/ego/trajectory features with a 100-dimensional context formed by 50 sampled route waypoints (hence *route50*). Both Q-networks share architecture but are optimized independently, and a target Twin Q is maintained via Polyak averaging with $\tau{=}5\times10^{-3}$. The policy objective is the conservative clipped-double-Q form $-\min(Q_1, Q_2)$, augmented by a behavior-cloning loss to the frozen prior Flow-Planner ($\lambda_{\text{BC}}{=}1.0$) and a trajectory-smoothness penalty ($\lambda_{\text{smooth}}{=}0.5$). The actor is updated every two Q-updates. Replay data is

Table 8: Off-manifold projection error on the trajectory manifold induced by the prior policy.

Split	Prior L_2	Final L_2	Prior-relative reduction	Positive scenario ratio	Sign-test p-value	Effect size
task_better	0.0363	0.0253	30.4%	68.4%	0.0835	0.50
task_worse	0.0352	0.0333	5.6%	46.7%	0.6964	0.07

collected in V-MAX closed-loop simulation under a mixed behavior policy that combines the learned actor, a frozen goal-conditioned Explorer, and the original Flow-Planner. The reported checkpoint corresponds to 450k Q-updates and 225k policy updates.

TD3 route50. TD3 *route50* reuses the SAC *route50* actor and critic architecture but replaces the SAC update rule with the three TD3 stabilizers. Target actions are smoothed with clipped Gaussian noise, $a' = \pi(s') + \mathrm{clip}(\epsilon, -0.3, 0.3)$ with $\epsilon \sim \mathcal{N}(0, 0.1^2)$, before being scored by the target Twin Q. The actor is optimized against $Q_1\big(s, \pi(s)\big)$ with the same behavior-cloning and smoothness regularizers, but only once every three Q-updates (delayed policy updates), and the target Q is Polyak-averaged only on policy-update steps. Replay buffer composition, behavior mixture, reward configuration, and remaining hyperparameters otherwise match SAC *route50*. The reported checkpoint corresponds to 650k Q-updates and 217k policy updates.

Trajectory-correction. The trajectory-correction baseline keeps the closed-loop training pipeline of NAC unchanged but moves the correction from the controller output to the planner trajectory output. Concretely, a corrector module Δ^{τ}_{ϕ} takes the same scene encoding h_t used by the action-correction module and outputs a trajectory-shaped correction that is added to τ^0_t before being passed through the LQR controller, $a_t = C(\tau^0_t + \Delta^{\tau}_{\phi}, s_t)$. The corrector head shares the depth and width of the action-correction module (cf. Section 3.4) but with output dimension $H \cdot d$ (the trajectory dimension) instead of 2 (the action dimension), and is trained under the same closed-loop reward, optimizer settings, and reward shaping as the action-correction variant. The corrected trajectory is then projected through the same LQR controller as the prior before entering closed-loop dynamics.

Policy Decorator transfer. We additionally transfer Policy Decorator (Yuan et al., 2025) to the same Flow-Planner prior. The transfer preserves its SAC-trained Gaussian correction, tanh-bounded output, and transferable hyperparameters while adapting the observation, output, and reward interfaces to the nuPlan simulator. This corrector is applied to the trajectory before the LQR controller.

H Off-manifold projection metric: details

To quantify prior preservation, we adopt the off-manifold-norm idea of Pan et al. (2026) with the reference manifold built from the prior policy's own closed-loop logs. The metric projects the prior trajectory and the NAC final trajectory onto this manifold and reports the per-scenario prior-relative reduction in projection error, which quantifies how much closer the corrected trajectory is to the prior. Scenarios are split into task_better and task_worse by whether NAC improves or degrades the task score.

The projection-error reduction on task_better is several times that on task_worse, and the positive scenario ratio crosses 50% only on task_better. This indicates that task improvement and manifold adherence move together rather than NAC acting as a generic trajectory regularizer. The sign-test p-value remains above the standard significance threshold while the per-group effect size on task_better is non-trivial, so we read the result as a directional trend.

Manifold construction. We extract a 21-dimensional state descriptor at each step of the prior-policy closed-loop logs (covering ego dynamics, route geometry, and interaction context) and use the future 0.5 s closed-loop trajectory snippet at the same step as the manifold entry. Given a query state, we select the k nearest entries in the bank under an IQR-normalized weighted Euclidean distance, fit a local affine subspace by SVD on their future trajectories (retaining 95% explained variance, max rank 8), and project both the prior trajectory y_{prior} and the NAC final trajectory y_{final} for the same

query onto this subspace. The paired projection errors $e_{\text{prior}}, e_{\text{final}}$ yield a per-scenario prior-relative reduction $r = (e_{\text{prior}} - e_{\text{final}})/e_{\text{prior}}$, and the metric is aggregated at scenario level (multiple queries within the same scenario are not treated as independent evidence). The target space is the real future closed-loop xy snippet rather than imagined constant-action rollouts, and the query selection uses a hybrid scheme covering tail steps, large-correction steps, and emergency-brake steps.

Headline preset and statistical design. Table 8 reports projection errors under descriptor preset `no_interaction` with $k = 128$ nearest manifold entries per query, namely ego weight 1.0, route weight 1.0, and interaction weight 0. The $\sim$ 25-percentage-point gap between `task_better` and `task_worse` on the prior-relative reduction column is what we read as the manifold-vs-task-improvement coupling, and the sign-test $p = 0.0835$ on `task_better` does not pass the 0.05 threshold while the per-group effect size of 0.50 is non-trivial. The paired projection design and scenario-level aggregation already absorb common dependence biases, so we read the trajectory-space result as a directional trend rather than a frequentist-significant effect.

Sensitivity to descriptor preset. The qualitative result remains under alternative presets. With `ego_heavy` (ego: 2.0, route: 0.75, interaction: 0.75, $k = 128$), `task_better` still attains a 24.8% reduction while `task_worse` shows -1.6%, giving the cleanest better-vs-worse contrast although the per-group statistical strength is weaker ($p = 0.2517$). With `ego_only` (ego: 1.0, route: 0, interaction: 0, $k = 64$), the statistical signal becomes the strongest (`task_better`: 44.5%, $p = 0.0013$, effect size 0.73), but `task_worse` also shows a 22.6% reduction. This preset captures a generic ego-dynamics-level regularization effect rather than a task-improvement-specific mechanism, so we treat it only as a sensitivity check.

Action-space target. When the target space is changed from the future closed-loop xy snippet to the executed-action prefix, namely comparing action sequences directly under the same nearest-neighbor subspace fit, the final action prefixes are farther from the prior-policy action manifold than the prior prefixes themselves on both `task_better` and `task_worse`. This is consistent with NAC's design that correction happens at the action space while prior preservation manifests in the unrolled trajectory, and the asymmetry between the two target spaces complements the trajectory-space result above.

I BEHAVIORAL ADVANTAGES OVER THE PRIOR

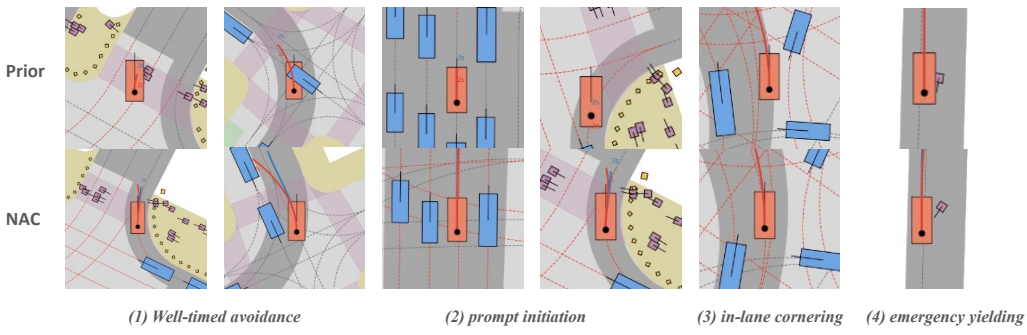

Figure 3: Behavioral advantages of NAC over the Flow-Planner prior on test14-hard. Each column corresponds to one of four representative scenarios, with the prior execution on the top row and NAC on the bottom row.

We group the test14-hard scenarios on which NAC outperforms the Flow-Planner prior into four behavioral categories and present representative side-by-side comparison per category in Figure 3. Specifically, 1) *well-timed avoidance* describes intersections and pedestrian-frequented zones where the prior hesitates or mistimes its decision, while NAC commits to a clean detour at the right moment. 2) *Prompt initiation* describes cases where the prior lingers in place after green-light onset or after an avoidance event ends, while NAC re-engages the throttle on time. 3) *In-lane cornering* describes cases where the prior tracks a wide arc that hits the shoulder or drifts onto adjacent lanes, while NAC

keeps the curve inside its own lane. 4) *Emergency yielding* describes suddenly appearing pedestrians or dense pickup-dropoff curbsides where the prior stalls or fails to give way, while NAC yields and threads through with stable heading. Note that all four corrections remain within the prior's executed-action neighborhood, so the gains arise from prompt and locally-shaped interventions at decision-critical steps rather than from large-scale behavior rewrites.