DriveAnchor: Progressive Anchor-based Flow Learning for Autonomous Driving Planning

Anonymous authors
Paper under double-blind review

Abstract

Autonomous-driving planners must generate route- and maneuver-compliant trajectories under a limited decoder-call budget while adapting offline to black-box static-collision feedback. We propose **DriveAnchor**, a three-stage framework built around a shared vocabulary of recorded trajectories and a continuous trajectory interface. The vocabulary supports flow-matching (FM) residual learning, Energy Field (EF) corridor-conditioned initialization, and anchor-guided zeroth-order reward fine-tuning. The reward update operates in trajectory space without policy likelihoods or reward gradients, enabling offline adaptation of a deterministic decoder. On sampled subsets of nuPlan Val14 and Test14-hard per simulation mode, DriveAnchor achieves the highest reactive scores among the compared methods: 81.50 and 65.74, respectively. We further integrate the trajectory generator into a real vehicle's planning pipeline and demonstrate its operation on urban roads. On 10,000 held-out internal scenarios, under the original non-EF prior and a fixed two-call FM budget, reward fine-tuning reduces short- and long-horizon static candidate collision rates from 27.22% and 56.54% to 2.94% and 7.15%, respectively, and improves 8-s best-of-set minADE from 2.27 to 1.34 m. In a separate junction diagnostic on checkpoints before Stage 3, one and two EF calls raise corridor-guidance satisfaction from 22.10% to 74.13% and 87.92%, respectively. Additional ablations support the benefits of real-trajectory priors and anchor-guided exploration. Code and deployment videos are available at the anonymous project page `https://driveanchor.github.io/`. In real-vehicle tests, trajectory generation takes 2.06 ms on average.

1 Introduction

A driving planner must generate trajectories that follow route and maneuver requests, select one for execution, and operate within a limited computation budget. A logged ego trajectory records one realized behavior, while the same scene may admit several plausible alternatives. When a case exposes an unsuitable behavior, developers also need to adjust the control request or improve the model's response while reusing the components already trained. These requirements call for a planning framework that connects diverse trajectory generation, explicit control, feedback-driven adaptation, and deployment.

These requirements meet at the trajectory interface. A control request changes where the decoder should start, while collision feedback indicates how its outputs should improve. For deterministic decoders, learning from this feedback is challenging: black-box collision checks may be non-differentiable, and the decoder does not directly provide the policy likelihoods used by stochastic reward-fine-tuning methods (Black et al., 2024; Zhang et al., 2025b; Xue et al., 2025; Liu et al., 2025). Exploration must also remain possible when the current model generates similar candidates.

DriveAnchor integrates scene encoding, controllable generation, reward adaptation, and candidate selection through a shared vocabulary of recorded trajectories. During FM training, locally perturbed anchors form trajectory starting states, so local anchor–demonstration pairs teach the decoder corrections across this structured distribution.

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.

The three stages use this representation to learn FM residual corrections, construct EF corridor-conditioned starts, and explore reward improvements around decoded outputs. At deployment, EF and FM compose through a small number of forward calls, followed by candidate selection.

Our contributions are threefold:

An integrated framework for controllable planning and targeted adaptation. A common trajectory interface connects supervised generation, corridor control, and black-box static-collision reward learning, and supports changes to request rules or selected model components.

A shared anchor prior for generation and reward exploration. Anchors serve as noisy FM starting states and provide reward-query directions independently of decoder-output diversity. We derive a likelihood-free trajectory-space update and characterize its expectation through a local, directionally smoothed surrogate objective in trajectory space.

System evaluation and controlled analysis. We evaluate complete planners on matched nuPlan scenarios, demonstrate urban-road deployment with Orin runtime measurements, and use controlled internal experiments to assess Stage 3 and EF.

2 Related Work

Generative planning and structured priors. Diffusion and flow-matching models (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023; Chen & Lipman, 2024) have been applied to trajectory generation and planning (Janner et al., 2022; Zheng et al., 2025; Tan et al., 2025). End-to-end planners combine scene representations with trajectory prediction (Hu et al., 2023; Jiang et al., 2023; Cheng et al., 2024b), while anchor-based methods represent possible futures through candidate trajectories, goals, or intentions (Chai et al., 2020; Zhao et al., 2021; Shi et al., 2022). Goal-conditioned Transformers unify prediction and planning with feasibility and interaction terms (Selzer & Flohr, 2026), and uncertainty-aware modules adapt prediction and ego planning to challenging agents in hybrid learning/rule-based planners (Chen et al., 2025b). DiffusionDrive combines multimodal anchors with truncated denoising (Liao et al., 2025), and DiffusionDriveV2 uses intra- and inter-anchor GRPO to improve generated trajectories (Zou et al., 2025). DriveAnchor trains FM from locally perturbed trajectory anchors and reuses the vocabulary to guide corridor-based initialization and define exploration directions for trajectory-space reward updates.

Controllable generation. Classifier-free guidance combines conditional and unconditional predictions to steer generation (Ho & Salimans, 2021), while cost-based guidance uses objective gradients during sampling (Zhong et al., 2023b;a). Int2Planner conditions multimodal planning on route intention points and has been deployed on real vehicles (Chen et al., 2025a). DriveAnchor instead learns a correction conditioned on the corridor geometry and current trajectory, and draws multimodality from the trajectory vocabulary rather than from intention points. EF is trained by regression and guides initialization through a forward pass at deployment.

RL fine-tuning of generative models. Reward-aligned fine-tuning includes RLHF (Ouyang et al., 2022) and GRPO (Shao et al., 2024), which estimates advantages within a sampled group and optimizes a clipped likelihood-ratio objective. DDPO (Black et al., 2024), ReinFlow (Zhang et al., 2025b), and DanceGRPO (Xue et al., 2025) optimize stochastic generative policies without differentiating the reward function. Flow-GRPO (Liu et al., 2025) introduces stochastic flow transitions through an ODE-to-SDE conversion. Applying these approaches requires a probability model that our deterministic FM decoder does not directly provide. MeZO (Malladi et al., 2023) instead studies zeroth-order fine-tuning in parameter space, ZPG estimates policy gradients from preference queries without reward inference (Zhang & Ying, 2025), and Guided ES (Maheswaranathan et al., 2019) shapes the search distribution using surrogate-gradient directions. RL has also been applied directly to driving planners: CarPlanner trains a consistent autoregressive policy with a generation-selection scheme at nuPlan scale (Zhang et al., 2025a), and NeuralParker learns a continuous-trajectory parking policy validated on a real vehicle (Wang et al., 2026). DriveAnchor instead constructs exploration directions from nearby vocabulary trajectories and propagates the resulting output-space signal to the model parameters through the decoder Jacobian.

3 Method

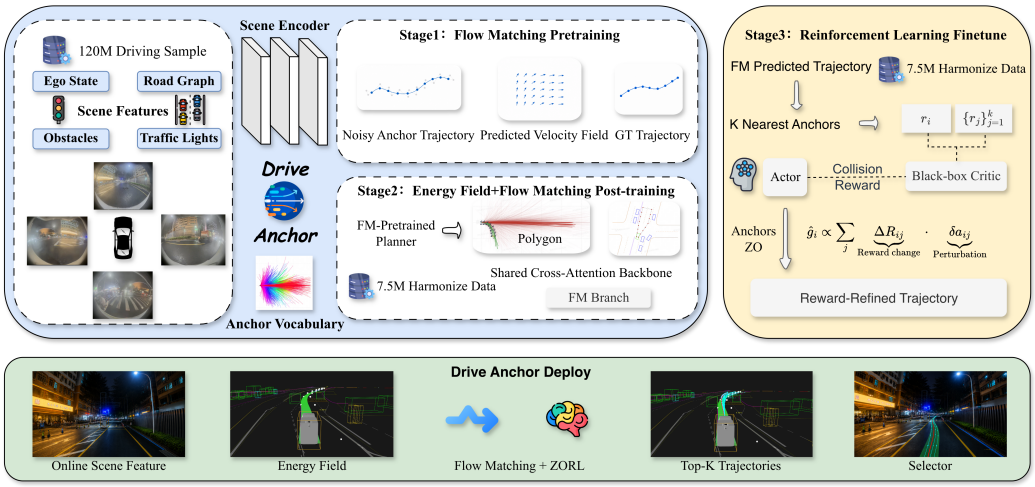


Figure 1: DriveAnchor training and deployment for the internal experiments. **Stage 1** (120M samples) learns residual corrections from perturbed local anchor–GT pairs. **Stage 2** (7.5M samples) buckets the data by dedicated scenario type, continues training FM on the bucketed data, and trains EF per scenario; the two are trained jointly with separate supervision, and EF predictions are not used as FM training inputs. **Stage 3** (2.5M samples) perturbs FM outputs, queries their rewards, and fine-tunes FM with a stop-gradient ZO signal. At inference, EF supplies initial trajectories to the reward-fine-tuned FM decoder, followed by residual screening and downstream selection; reward queries and ZO updates are performed offline.

Given a scene and a corridor request, DriveAnchor generates candidates by repeatedly correcting continuous trajectories. A fixed vocabulary provides reference trajectories for training and reward exploration; EF and FM share the trajectory representation passed between modules. At deployment, the scene is encoded once, EF adjusts the initialization, and FM refines it for downstream selection. Figure 1 shows the internal pipeline. All ZO reward queries and parameter updates occur offline. We use "FM decoder" for the learned module and "residual correction" for its inference operation.

3.1 Stage 1: FM Pretraining

A shared scene-conditioned FM decoder is trained on local anchor–GT pairs induced by a vocabulary of M=2,398 anchors. The vocabulary is built by iterative farthest-point sampling (Qi et al., 2017) over a historical corpus $\mathcal{H}$ of 100M+ frames:

$$\mathcal{A} = \mathrm{FPS}\Big(\{a_k^*\}_{k=1}^{|\mathcal{H}|},\ M\Big), \quad a_k^* \in \mathbb{R}^{T\times 2} \tag{1}$$

Trajectories are flattened to $2T$-dimensional vectors for distances and inner products. The corpus is temporally disjoint from training and evaluation data. For each GT trajectory x_1, we select its K_{GT} nearest vocabulary anchors as pairing partners; the local GT pairing count K_{GT} is the number of nearest anchors paired with each GT trajectory, fixed to $K_{\mathrm{GT}} = 50$ throughout. Stage 1 constructs training pairs only from these anchors. We first perturb the selected anchor, $\tilde{x}_0 = a + \varepsilon_{\mathrm{in}}$, and then sample the linear path:

$$x_t = (1-t)\,\tilde{x}_0 + t\,x_1, \quad t \sim \mathrm{Uniform}(0,1) \tag{2}$$

The perturbed anchor defines the path's starting state; scene features s provide conditioning.

The noise $\varepsilon_{\mathrm{in}}$ used in FM training is sampled as follows. In Stages 1 and 2, noise is drawn uniformly from a ball, and a perturbed trajectory is accepted only if its nearest vocabulary anchor remains unchanged, so the perturbed start still belongs to the same anchor. Appendix J.2 discusses the sampling procedure further.

Let F_θ denote the FM decoder: it takes the current trajectory $x_t \in \mathbb{R}^{T\times 2}$, the time $t \in [0,1]$, and the scene features s, and outputs a predicted remaining displacement $F_\theta(x_t, t, s) \in \mathbb{R}^{T\times 2}$ toward the target trajectory x_1. The time-conditioned decoder learns this displacement using the regression term retained in all stages:

$$\mathcal{L}_{\text{res}}(\theta) = \mathbb{E}\big[\ell_{\text{SL1}}(F_\theta(x_t, t, s), x_1 - x_t)\big]. \tag{3}$$

Here ℓ_{SL1} denotes the coordinate-mean SmoothL1 loss. For the linear-path velocity target $v^\star = x_1 - \tilde{x}_0$, the residual satisfies

$$x_1 - x_t = (1-t)v^\star. \tag{4}$$

At inference, each FM call updates the trajectory:

$$T_\theta(x, s) \triangleq x + F_\theta(x, 0, s). \tag{5}$$

Starting from a raw anchor or an EF output, we repeatedly apply T_θ, using the previous output as input and resetting $t = 0$ at each call. This performs iterative residual correction. Section 4.5 evaluates the effect of repeated calls.

3.2 Stage 2: EF Post-training

EF predicts an initialization correction $\Delta a_{\text{EF}} \in \mathbb{R}^{T\times 2}$ from the current trajectory, corridor request, and static-geometry features of the shared scene encoder. The corridor polygon and designated exit specify the maneuver. EF outputs a correction vector rather than a scalar energy.

Control-request construction. The corridor polygon and designated exit are constructed from the navigation route, map geometry, and current ego state, independently of the recorded ground-truth trajectory; the ground truth need not satisfy the corridor request. The six corridor presets and their geometric construction are detailed in Appendix J.1 (Figure 4).

One EF call followed by one FM call gives:

$$\tilde{a}_{\text{EF}} = a_{\text{anchor}} + \Delta a_{\text{EF}}, \qquad a_{\text{final}} = T_\theta(\tilde{a}_{\text{EF}},\, s). \tag{6}$$

EF passes its continuous output directly to FM, without vocabulary rematching. Further FM calls follow Eq. (5).

Training and composition. Stage 2 first buckets the training data by dedicated scenario type, then continues training FM on the bucketed data and trains EF specifically for each scenario. The two are trained jointly with separate targets: corridor corrections for EF and anchor–demonstration residuals for FM. Both use a frozen scene encoder and a shared trainable trajectory projector. FM training does not use EF predictions as inputs. At inference, EF can refine its output repeatedly before passing it to FM.

Stage 3 decodes raw anchors during training. Composing EF with the trained FM decoder therefore requires generalization to EF-adjusted inputs. We test whether the EF+FM composition is effective through candidate quality after EF initialization and final-output guidance satisfaction.

Training objective. Let $\text{touch}_{\mathcal{P}}(a)$ indicate that the trajectory enters the polygon or starts inside it, with an additional permitted-entry-side check for lane changes. Let $\text{inside}_{\text{req}}(a)$ indicate occupancy at a specified time when that optional check is enabled, $n_{\text{exit}}(a)$ count exits, and $e_{\text{exit}}(a)$ identify the edge used for a single exit. The time check is part of label construction, not an extra EF input, and is always true when disabled. The geometric label is

$$\begin{aligned}\text{good}(a) = \text{touch}_{\mathcal{P}}(a) \wedge \text{inside}_{\text{req}}(a)\\ \wedge \big[n_{\text{exit}}(a) = 0 \;\vee\; (n_{\text{exit}}(a) = 1 \wedge e_{\text{exit}}(a) = \mathcal{E}_k)\big].\end{aligned} \tag{7}$$

A trajectory can satisfy the request by remaining inside the polygon after entry; if it leaves, it must use the designated exit. We evaluate the geometric criterion separately for unperturbed vocabulary entries and noisy inputs. The former define the reference pool $\mathcal{A}_{\text{good}} = \{b \in \mathcal{A} : \text{good}(b)\}$, while the latter determine which supervision branch to use. For a nonempty reference pool,

$$\delta^*(a_{\text{noisy}}) = \begin{cases}\mathbf{0}, & \text{good}(a_{\text{noisy}}),\\ b^*(a_{\text{noisy}}) - a_{\text{noisy}}, & \text{otherwise},\end{cases} \quad b^*(a_{\text{noisy}}) \in \arg\min_{b\in\mathcal{A}_{\text{good}}} \frac{1}{T}\sum_{t=1}^{T} \|b_t - a_{\text{noisy},t}\|_2. \tag{8}$$

EF uses mean pointwise Euclidean distance to retrieve references and SmoothL1 with kinematic losses to learn their displacements.

3.3 Stage 3: RL Fine-tuning

Stage 3 samples starting anchors uniformly from the full vocabulary and adapts the decoder's outputs using black-box static-collision feedback. We use "RL fine-tuning" as shorthand for RL-inspired zeroth-order reward fine-tuning; the update does not use policy likelihoods or policy gradients. We seek to maximize expected output reward:

$$J(\theta) = \mathbb{E}_{s\sim\rho,\, x_0\sim\mathcal{A}}[R(s,\, T_\theta(x_0, s))] \tag{9}$$

where ρ is the scene distribution and R is a black-box reward. Reward differences under anchor-guided output perturbations provide the learning signal. Its expectation is the gradient of a query-local smoothed surrogate (Eq. (16)), providing a surrogate update direction for optimizing J.

Sampling from the full vocabulary extends reward feedback beyond Stage 1's GT-neighbor pairs, and neighboring vocabulary trajectories provide multiple exploration directions even when the decoder generates only one trajectory. The resulting signal is backpropagated through the trajectory output to model parameters, without differentiating the reward function.

Stage 3 neighbor retrieval. For the current model output a_i, compute Euclidean distances to the full vocabulary. Excluding zero-distance entries gives $\mathcal{A}_i^+ = \{b \in \mathcal{A} : \|b - a_i\|_2 > 0\}$. Retrieve its nearest N anchors:

$$\mathcal{N}(a_i) = \text{TopNearest}_N(\mathcal{A}_i^+; a_i), \qquad N = \min(N_{\text{ZO}}, |\mathcal{A}_i^+|). \tag{10}$$

Step 1: Direction Retrieval and Sampling. For a sampled starting anchor $x_{0,i}$, generate $a_i = T_\theta(x_{0,i}, s)$ and retrieve N nearest anchors $b_{ij} \neq a_i$ using Eq. (10). Construct

$$\sigma_i = \frac{1}{N}\sum_{j=1}^{N} \|b_{ij} - a_i\|_2, \quad u_{ij} = \frac{b_{ij} - a_i}{\|b_{ij} - a_i\|_2}, \quad z_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1), \quad \widetilde{a}_{ij} = a_i + \sigma_i z_{ij} u_{ij}. \tag{11}$$

The mean neighbor distance σ_i sets the perturbation scale, while the Gaussian scalar z_{ij} samples both signs of each neighbor direction. Neighbors, directions, and scale remain fixed within each estimate.

Step 2: Reward Querying. Evaluate $r_i = R(s, a_i)$ and $\widetilde{r}_{ij} = R(s, \widetilde{a}_{ij})$. Higher reward is better:

$$R(s,a) = \begin{cases} \min\{j \in \{0,\ldots,T-1\} : \text{collision at waypoint } j\}, & \text{if a collision occurs,} \\ T+1, & \text{otherwise,} \end{cases} \tag{12}$$

Collision checks cover static obstacles and road boundaries.

Step 3: Directional Signal Estimation.

$$\hat{g}_i = \frac{1}{N\sigma_i}\sum_{j=1}^{N}[R(s,\widetilde{a}_{ij}) - R(s,a_i)]z_{ij}u_{ij}. \tag{13}$$

Step 4: Policy Update. Treating $\hat{g}_i$ as a stop-gradient constant, the RL loss and combined training objective are:

$$\mathcal{L}_{\text{RL}}(\theta) = -\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \text{sg}(\hat{g}_i)^\top a_i, \qquad \mathcal{L}(\theta) = \mathcal{L}_{\text{FM}}(\theta) + \beta\, \mathcal{L}_{\text{RL}}(\theta) \tag{14}$$

The update uses a finite-difference learning signal, as in zeroth-order methods such as SPSA (Spall, 1992) and ES (Salimans et al., 2017). Here, perturbations are constructed in trajectory space using vocabulary directions. The weight β balances reward learning with the FM training objective.

Local interpretation of the update. For a fixed query, scene, and set of directions, let α shift the mean of the one-dimensional radial smoothing distribution. The smoothed reward and the expected estimator satisfy

$$\mathcal{Q}_{ij,\sigma_i}(\alpha) = \mathbb{E}_z R(s, a_i + (\alpha + \sigma_i z)u_{ij}), \qquad \mathbb{E}[\hat{g}_i \mid s, a_i, \{u_{ij}\}, \sigma_i] = \frac{1}{N}\sum_{j=1}^{N} \mathcal{Q}'_{ij,\sigma_i}(0)u_{ij}. \quad (15)$$

The prime denotes differentiation with respect to α, evaluated at $\alpha = 0$, so the perturbation distribution remains centered at the current trajectory a_i. This identity holds for bounded measurable rewards, including the piecewise-constant collision reward. It relates the expected signal to derivatives of directionally smoothed rewards. Appendix K gives the proof, variance bound, and connection to parameter updates.

Local reward objective. Fixing the current output and its neighbor directions, define the locally smoothed reward:

$$H_i(\Delta) = \frac{1}{N}\sum_{j=1}^{N} \mathbb{E}_z R\big(s, a_i + (u_{ij}^\top \Delta + \sigma_i z)u_{ij}\big), \quad \nabla_\Delta H_i(0) = \mathbb{E}[\hat{g}_i \mid s, a_i, \{u_{ij}\}, \sigma_i]. \quad (16)$$

H_i averages rewards under directional exploration with shifted radial means, rather than evaluating the full trajectory $a_i + \Delta$. When $\nabla_\Delta H_i(0) \neq 0$, a sufficiently small step along this exact gradient increases the fixed local surrogate. This differs from ZPG (Zhang & Ying, 2025), which targets the policy gradient of the global objective: our estimator is unbiased for the gradient of the query-dependent surrogate H_i, and its expected signal lies in the span of the N retrieved directions. For sufficiently smooth rewards and small perturbations, this signal approximates a directionally weighted reward gradient; its connection to ∇J requires the conditions in Appendix K.

3.4 Training Pipeline and Inference

For EF-initialized configurations, the decoding start $x_k^{(0)}$ denotes the EF-refined trajectory; for the original-prior configuration, it is the raw vocabulary anchor. The FM call count is an inference-time budget: every call reuses the same decoder at $t = 0$, so latency can be traded for correction quality without retraining (Section 4.5).

All stages retain FM regression. Stage 2 jointly trains EF and FM; Stage 3 freezes EF-specific weights ϕ and adds reward learning to FM. The encoder and vocabulary remain fixed. The FM parameters θ include the shared trajectory projector, so Stage 3 can still change EF's input–output mapping through this projector. Algorithm 1 summarizes training.

Internal inference and candidate selection. We decode all $M = 2{,}398$ anchors in parallel. For initial trajectory $x_k^{(0)}$, let $x_k^{(1)} = T_\theta(x_k^{(0)}, s)$ and $\Delta_k^{(2)} = F_\theta(x_k^{(1)}, 0, s)$. We retain the $K_{\text{sel}} = 50$ candidates with the smallest $\|\Delta_k^{(2)}\|_2$. This residual measures how much the decoder would change its first-call output, indicating proximity to a fixed point rather than collision safety; screening therefore costs one extra FM forward pass and no reward queries. The downstream selector rejects candidates that fail kinematic, static-collision, or dynamic time-to-collision (TTC) checks, then ranks survivors by safety, efficiency, and comfort. In the production-release configuration, a rule-based planner takes over if none pass. The real-vehicle demonstration configuration reported in Section 4.3 does not include this rule-based fallback planner. All internal metrics in Sections 4.4–4.6 are computed on candidates before this selector, over either the full pool or the residual-screened Top-K_{sel} subset, and they involve static geometry only; the public nuPlan score in Section 4.2 instead evaluates the single executed plan in closed loop.

4 Experiments

We first evaluate the complete planner on nuPlan and demonstrate real-vehicle deployment, then use controlled internal studies to examine Stage 3, EF initialization, decoding calls, and candidate screening; further ablations are in the appendix.

4.1 Experimental Setup

Public evaluation and deployment. We evaluate nuPlan (Caesar et al., 2021) Val14 and Test14-hard using a sampled subset of scenarios per benchmark and simulation mode. All methods use the same scenario lists and official closed-loop scoring. Non-reactive (NR) simulation replays logged traffic; reactive (R) simulation uses IDM-driven vehicles. The nuPlan model is trained with a vocabulary of 3,000 anchor trajectories, each with 40 waypoints. It generates 3,000 candidates with EF$\times$2 and FM$\times$2, then selects one execution trajectory using the score head. We also test the generator on urban roads and measure its runtime on NVIDIA Drive Orin. Appendix A details the public evaluation protocol and scenario sampling.

Internal data and training. The main internal Stage 1, Stage 2, and Stage 3 experiments use 120M, 7.5M, and 2.5M samples, respectively, with disjoint training and evaluation cases. Unless stated otherwise, internal studies use a 10,000-scene held-out set and 2,398 candidates per scene; the main table and RL comparisons use the main list, while junction-guidance and training-curve diagnostics use separate lists. Training ablations start from the main Stage 2 checkpoint and use 2.5M Stage 3 samples unless noted otherwise. The lightweight exploration and vocabulary ablations in Tables 8 and 9 instead use a smaller, intersection-specific dataset containing approximately 100,000 scenarios, with evaluation on 1,000 held-out scenarios excluded from training.

Metrics and comparisons. The nuPlan closed-loop score evaluates the executed plan on a 0–100 scale (higher is better), combining collision safety, drivable-area and driving-direction compliance, route progress, TTC, speed-limit compliance, and comfort (Appendix A). Internal experiments evaluate candidates before downstream selection. For trajectory accuracy in meters, we use

$$\text{ADE} = \frac{1}{H}\sum_{t=1}^{H} \|\hat{a}_t - a_t^*\|_2, \qquad \text{FDE} = \|\hat{a}_H - a_H^*\|_2 \tag{17}$$

minADE/FDE is the minimum over the candidate set, evaluated at 3 s and 8 s. Static collision rates cover obstacles and road boundaries over the first 40 or all 80 waypoints ($R < 40$ or $R < 80$); mean reward averages Eq. (12). EF satisfaction measures the fraction meeting Eq. (7). FM$\times n$ denotes n residual-correction calls; FMRL uses the reward-fine-tuned decoder. We fix the FM call count when comparing training stages and report EF initialization separately.

RL/ZO comparison setup. We compare the deterministic FM update with trajectory-space GRPO/DPO-inspired surrogates and standard ZO under matched query budgets; implementation details are given in Appendices D and C.

4.2 Public Closed-Loop Evaluation on nuPlan

Table 1 compares DriveAnchor with public baseline configurations under the matched-scenario protocol. Configurations marked with an asterisk receive a pre-searched reference line or additional proposals as input, so their scores reflect extra prior information; DriveAnchor builds its corridor request from the navigation route, map boundaries, and current ego state only.

DriveAnchor achieves the highest R scores among the compared methods on both sampled subsets: 81.50 on Val14 and 65.74 on Test14-hard. Under NR simulation, it scores 84.80 on Val14, close to PlanTF (84.72), and 63.37 on Test14-hard, below PlanTF (69.70) and PLUTO (70.03). The comparison therefore favors DriveAnchor in the setting with reactive surrounding vehicles, while a small performance gap remains on the hard subset with fixed traffic replay. CarPlanner (Zhang et al., 2025a) also uses generation-selection but generates candidates autoregressively with an RL-trained policy, favoring temporal consistency, whereas DriveAnchor decodes its pool in parallel from a fixed anchor set at a fixed call budget; its protocol differs from our matched lists, so it is not in Table 1.

4.3 Real-Vehicle Deployment

DriveAnchor is integrated into a real vehicle's planning and execution pipeline and tested on urban roads. EF-guided candidates are refined by FM, then passed to the vehicle's downstream trajectory selector for execution. The reported road tests and accompanying videos use a demonstration

Table 1: nuPlan official closed-loop scores (0–100, higher is better). Val14 and Test14-hard each use a fixed sampled subset of scenarios for each simulation mode, and every re-evaluated method uses the same list. DriveAnchor generates the full pool with EF$\times$2 and FM$\times$2, then selects one trajectory. An asterisk marks a configuration whose input includes a pre-searched reference line or additional proposals. "w/o refine." removes the method's post-processing/refinement module; "w/o LLM" removes Diffusion-ES's LLM guidance; STR2-CPKS-800M names the checkpoint/configuration of the 800M-parameter STR2 model. A dash means not reported or not evaluated.

	Val14		Test14-hard	
Method	NR ↑	R ↑	NR ↑	R ↑
PLUTO w/o refine.* (Cheng et al., 2024a)	88.89	78.11	70.03	59.74
PlanTF (Cheng et al., 2024b)	84.72	76.95	69.70	61.61
UrbanDriver (Scheel et al., 2022)	68.57	64.11	50.40	49.95
STR2-CPKS-800M w/o refine.*	65.16	–	52.57	–
PDM-Open* (Dauner et al., 2023)	53.53	54.24	33.51	35.83
Diffusion-ES w/o LLM (Yang et al., 2024)	50.00	–	–	–
GameFormer w/o refine. (Huang et al., 2023)	13.32	8.69	7.08	6.69
DriveAnchor	84.80	81.50	63.37	65.74

configuration without a rule-based fallback planner, distinct from the production-release configuration described in Section 3.4. Figure 2 shows test environments with intersections, pedestrians, two-wheelers, and surrounding vehicles. Deployment configurations, full videos of each scenario, and the complete on-road recording are available on the project page (`https://driveanchor.github.io/`).

On-vehicle runtime. In a separate timing measurement on NVIDIA Drive Orin using float16, two EF calls and two FM calls take **2.06 ms**/scene on average. This measures the trajectory-generation module, excluding the encoder and downstream selector.

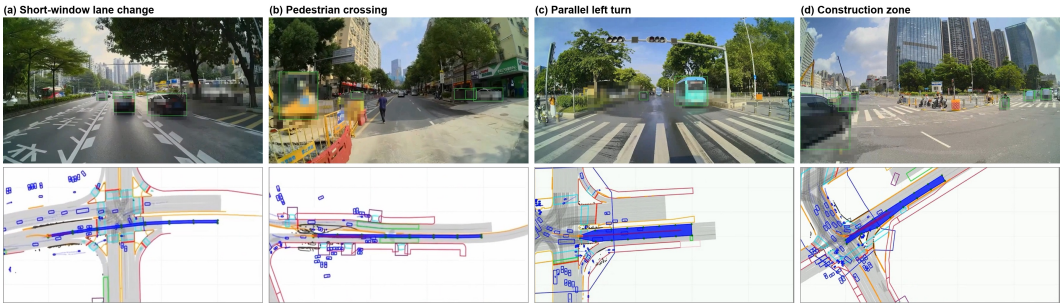


Figure 2: Four anonymized real-vehicle deployment scenarios: (a) short-window lane change, (b) pedestrian crossing, (c) parallel left turn, and (d) construction zone. **Top:** forward-camera frame. **Bottom:** the planner's bird's-eye view at the same moment, with the selected trajectory in blue.

4.4 Controlled Internal Candidate Analysis

Table 2 reports full-candidate results on the 10,000-scenario internal evaluation set. We compare checkpoints at fixed FM call counts to assess Stage 3, then vary EF initialization to examine its effect on candidate quality.

At a fixed two-call FM budget, Stage 3 reduces short-/long-horizon candidate collision from 27.22%/56.54% to 2.94%/7.15% and improves minADE@80 from 2.27 to 1.34 m. With EF initialization, one additional EF call improves short-horizon collision and minADE@80 but increases long-horizon collision, while raising the corridor-guidance satisfaction rate; Section 4.5 gives the experimental details. This reveals a horizon-dependent accuracy–collision trade-off. Replacing FM with FMRL under EF initialization again reduces both collision rates and minADE.

Table 2: Main internal results on 10,000 randomly selected scenarios. Stage 3 comparisons fix the prior type and FM call count. EF (†) changes the initialization, and additional calls increase inference computation. Collision rates are fractions; bold marks the best result within each prior group.

	Original prior		*EF-modified prior (†)*		
Metric	**FM**×2	**FMRL**×2	**EF**×1 + **FM**×1	**EF**×1 + **FMRL**×1	**EF**×1 + **FMRL**×2
Trajectory Accuracy (m)					
minADE@30 ↓	0.35	**0.25**	0.56	0.28	**0.22**
minFDE@30 ↓	0.80	**0.54**	1.27	0.55	**0.44**
minADE@80 ↓	2.27	**1.34**	2.35	1.25	**0.95**
minFDE@80 ↓	4.42	**2.82**	4.44	2.65	**1.87**
Candidate static collision					
Short-horizon collision ↓	0.2722	**0.0294**	0.1313	0.0423	**0.0188**
Long-horizon collision ↓	0.5654	**0.0715**	0.2627	0.1178	**0.0873**
Reward					
Mean reward ↑	59.22	**78.46**	70.13	77.14	**78.87**

4.5 EF Guidance and Decoding-Call Analysis

EF calls and guidance satisfaction. On a separate 10,000-scene junction set, each configuration evaluates 2,398 candidates per scene, corresponding to 23,980,000 candidate evaluations. Before Stage 3, one and two EF calls raise guidance satisfaction from 22.10% to 74.13% and 87.92%; with the final checkpoint and two FMRL calls fixed, they raise final-output satisfaction from 62.02% to 70.31% and 73.42%. Thus, repeated EF corrections improve request satisfaction.

FM calls after EF initialization. Additional FM calls remain beneficial after Stage 3 but show diminishing returns; the full checkpoint-by-call analysis is in Appendix I.

4.6 Candidate Screening

Residual screening and candidate quality. Residual screening improves both collision rates and mean reward for all three FMRL configurations after Stage 3, but worsens short-horizon collision from 13.13% to 22.76% for EF×1+FM×1 before Stage 3. These results support combining Stage 3 with residual screening (Appendix I).

5 Conclusion and Limitations

DriveAnchor connects FM generation, EF control, and zeroth-order reward adaptation through a shared trajectory vocabulary. On matched nuPlan scenarios it achieves the highest reactive closed-loop scores among the compared planners, and the same generator has been deployed on a real vehicle in urban traffic. The gains are system-level rather than additive: one vocabulary serves FM training, EF reference pools, and ZO exploration; EF initialization with reward fine-tuning gives the best accuracy and short-horizon collision (Table 2); and residual screening pays off only after Stage 3 (Section 4.6).

Limitations and future work. DriveAnchor relies on a fixed trajectory vocabulary to structure generation and exploration. Its coverage depends on the source data and may be limited for rare maneuvers or unfamiliar road layouts; expanding the vocabulary to cover such cases is a natural extension. The current Stage 3 reward focuses on static obstacles and road boundaries; incorporating dynamic interactions, for example through uncertainty-aware handling of challenging agents (Chen et al., 2025b), would better align reward adaptation with interactive driving. The ZO analysis characterizes only local optimization behavior under its stated assumptions and does not guarantee global convergence; the global structure of the reward landscape remains to be studied. Our public evaluation uses one sampled scenario subset per benchmark and simulation mode; larger evaluation sets and repeated sampling would better test stability and representativeness, and targeted component adaptation warrants a study of case-repair efficiency.

AI Use Statement

In this work, we used generative AI tools for translation between Chinese and English and for critiquing the experimental methodology and result interpretation. We have not used generative AI tools for creating synthetic data, formulating mathematical claims, supplying proof ingredients, proposing hypotheses, or implementing the method; the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for editing text for readability, checking the mathematical exposition, preparing figures and tables, and LaTeX formatting. We have reviewed all AI-assisted work, and all technical claims and experimental results were verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

References

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=YCWjhGrJFD.

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 2021 Workshop on Autonomous Driving: Perception, Prediction and Planning (ADP3)*, 2021. URL https://arxiv.org/abs/2106.11810.

Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. MultiPath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 100, pp. 86–99. PMLR, 2020. URL https://proceedings.mlr.press/v100/chai20a.html.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=g7ohDlTITL.

Xiaolei Chen, Junchi Yan, Wenlong Liao, Tao He, and Pai Peng. Int2Planner: An intention-based multi-modal motion planner for integrated prediction and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14558–14566, 2025a. doi: 10.1609/aaai.v39i14.33595. URL https://arxiv.org/abs/2501.12799.

Yuheng Chen, Yesheng Zhang, and Xu Zhao. Uncertainty-aware module for trajectory prediction and ego-planning of challenging agents in autonomous driving. In *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pp. 1167–1172, 2025b. doi: 10.1109/RCAR65431.2025.11139566. URL https://ieeexplore.ieee.org/document/11139566.

Jie Cheng, Yingbing Chen, and Qifeng Chen. PLUTO: Pushing the limit of imitation learning-based planning for autonomous driving. *arXiv preprint arXiv:2404.14327*, 2024a. URL https://arxiv.org/abs/2404.14327.

Jie Cheng, Yingbing Chen, Xiaodong Mei, Bowen Yang, Bo Li, and Ming Liu. Rethinking imitation-based planners for autonomous driving. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14123–14130, 2024b. doi: 10.1109/ICRA57147.2024.10611364. URL https://ieeexplore.ieee.org/document/10611364.

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 (CoRL)*, volume 229, pp. 1268–1281. PMLR, 2023. URL https://proceedings.mlr.press/v229/dauner23a.html.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17853–17862, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Hu_Planning-Oriented_Autonomous_Driving_CVPR_2023_paper.html.

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. 3880–3890, 2023. doi: 10.1109/ICCV51070.2023.00361. URL https://arxiv.org/abs/2303.05760.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pp. 9902–9915. PMLR, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8340–8350, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Jiang_VAD_Vectorized_Scene_Representation_for_Efficient_Autonomous_Driving_ICCV_2023_paper.html.

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, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liao_DiffusionDrive_Truncated_Diffusion_Model_for_End-to-End_Autonomous_Driving_CVPR_2025_paper.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 40783–40818. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3a10c46572628d58cb44fb705f25cbbf-Abstract-Conference.html.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Niru Maheswaranathan, Luke Metz, George Tucker, Dami Choi, and Jascha Sohl-Dickstein. Guided evolutionary strategies: Augmenting random search with surrogate gradients. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pp. 4264–4273. PMLR, 2019. URL https://proceedings.mlr.press/v97/maheswaranathan19a.html.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 53038–53075, 2023. doi: 10.52202/075280-2308. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a627810151be4d13f907ac898ff7e948-Abstract-Conference.html.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 27730–27744, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5099–5108, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d8bf84be3800d12f74d8b05e9b89836f-Abstract.html.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 53728–53741, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017. URL https://arxiv.org/abs/1703.03864.

Oliver Scheel, Luca Bergamini, Maciej Wołczyk, Błażej Osiński, and Peter Ondrúška. Urban Driver: Learning to drive from real-world demonstrations using policy gradients. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, volume 164, pp. 718–728. PMLR, 2022. URL https://proceedings.mlr.press/v164/scheel22a.html.

Constantin Selzer and Fabian B. Flohr. PlanTRansformer: Unified prediction and planning with goal-conditioned transformer. In *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1340–1347. IEEE, 2026. doi: 10.1109/IV66570.2026.11624123. URL https://ieeexplore.ieee.org/document/11624123.

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

Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 6531–6543, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/2ab47c960bfee4f86dfc362f26ad066a-Abstract-Conference.html.

James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992. doi: 10.1109/9.119632. URL https://ieeexplore.ieee.org/document/119632.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 38310–38335. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/36d1e8aa9ceec3b781682bf5e63c31bf-Abstract-Conference.html.

Zihan Wang, Bai Huang, Yang Guan, Xiao Li, Haoyu Xu, Naizheng Wang, and Shengbo Eben Li. NeuralParker: A reinforcement learning planner for irregular parking environments. *arXiv preprint arXiv:2608.24485*, 2026. URL https://arxiv.org/abs/2608.24485.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. DanceGRPO: Unleashing GRPO on visual generation. *arXiv preprint arXiv:2505.07818*, 2025. URL https://arxiv.org/abs/2505.07818.

Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Kumar 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, 2024. doi: 10.1109/CVPR52733.2024.01453. URL https://arxiv.org/abs/2402.06559.

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, 2025a. URL https://arxiv.org/abs/2502.19908.

Qining Zhang and Lei Ying. Zeroth-order policy gradient for reinforcement learning from human feedback without reward inference. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=cmYScmfu4Q.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning flow matching policy with online reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 106282–106319. Curran Associates, Inc., 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/98d6f928497d9eaad9f320fb2db3040d-Abstract-Conference.html.

Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, Congcong Li, and Dragomir Anguelov. TNT: Target-driven trajectory prediction. In *Proceedings of the 2020 Conference on Robot Learning (CoRL)*, volume 155, pp. 895–904. PMLR, 2021. URL https://proceedings.mlr.press/v155/zhao21b.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 (ICLR)*, 2025. URL https://openreview.net/forum?id=wM2sfVgMDH. Oral.

Ziyuan Zhong, Davis Rempe, Yuxiao Chen, Boris Ivanovic, Yulong Cao, Danfei Xu, Marco Pavone, and Baishakhi Ray. Language-guided traffic simulation via scene-level diffusion. In *Proceedings of The 7th Conference on Robot Learning (CoRL)*, volume 229, pp. 144–177. PMLR, 2023a. URL https://proceedings.mlr.press/v229/zhong23a.html.

Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3560–3566, 2023b. doi: 10.1109/ICRA48891.2023.10161463. URL https://ieeexplore.ieee.org/document/10161463.

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. URL https://arxiv.org/abs/2512.07745.

A Evaluation Protocol and Internal Experimental Setup

nuPlan evaluation protocol. We evaluate Val14 and Test14-hard in non-reactive (NR) and reactive (R) closed-loop simulation: NR replays logged traffic, while R uses nuPlan's official IDM configuration. For each benchmark and simulation mode, we draw 100 unique scenarios with a fixed random seed; every method uses the same list, and we report the mean score over that list. The nuPlan score ranges from 0 to 100 (higher is better) and combines collision safety, drivable-area and driving-direction compliance, route progress, TTC, speed-limit compliance, and comfort. The adapter constructs EF requests only from the navigation route, map boundaries, and current ego state; logged future actors used for offline reward calculation are not inputs to the generator or deployed planner.[1]

Public-model configuration. The public model uses 3,000 anchor trajectories, each with 40 two-dimensional waypoints, giving a vocabulary of shape $3{,}000 \times 40 \times 2$. All public candidates receive EF$\times 2$ and FM$\times 2$. The score head uses anchor geometry, encoded scene context, and a 50-dimensional route-corridor representation to provide scores for selecting a single execution trajectory.

Data provenance. Stage 1 covers intersections, roundabouts, lane changes, and ramp merges; Stage 2 data are bucketed by dedicated scenario type and cover all EF polygon presets. We construct $\mathcal{A}$ by FPS on a separate corpus of 100M+ frames collected outside the training and evaluation periods. The stage-wise data counts in Section 4.1 exclude pretraining of the inherited frozen encoder, whose full data volume is unavailable.

Evaluation and initialization. The case-disjoint splits and dataset sizes are specified in Section 4.1. The main table and RL comparisons use the main 10,000-scene list; junction-guidance and training-curve diagnostics use separate lists. Tables 8 and 9 use the smaller intersection-specific dataset and its 1,000 held-out evaluation scenarios, rather than the main evaluation list. ZO statistics aggregate across ranks. Unless noted otherwise, the RL ablations reuse the main Stage 2 EF/FM checkpoint; the $N = 16$ reference in Table 10 retains the main training and evaluation settings.

Comparison protocol. Stage 3 comparisons hold the prior and FM call count fixed and measure the combined FM, kinematic, and reward objective in Eq. (14). Shared upstream projections remain trainable in EF-initialized configurations. Main-table results come from single training runs; reported noise-repetition SDs do not estimate training-seed variability. Appendix D specifies the alternative RL updates.

Metric details. Candidate metrics aggregate over the decoded vocabulary and test scenes before downstream selection; residual-screened subsets are reported separately. The horizons use waypoint indices 0–39 and 0–79, respectively. Dynamic-agent collisions are outside the static reward and collision metrics.

Training dynamics. Figure 5 tracks learning across the three stages. During reward fine-tuning, the FM loss stays close to the frozen baseline, while collision metrics level off late in training. Figure 9 shows Stage 3 loss and gradient diagnostics; Appendix J.2 provides the remaining training visualizations.

B Training Procedure

In Algorithm 1, ϕ denotes only EF-specific weights; the trajectory projector is shared by EF and FM (Section 3.4).

Regression scaling and kinematic scheduling. The coordinate-mean SmoothL1 loss is $\ell_{\mathrm{SL1}}(p, q) = (2T)^{-1} \sum_{k=1}^{2T} \rho(p_k - q_k)$, where $\rho(e) = e^2/2$ for $|e| < 1$ and $\rho(e) = |e| - 1/2$ otherwise. The FM objective is $\mathcal{L}_{\mathrm{FM}} = 0.005\, \mathcal{L}_{\mathrm{res,impl}} + \chi_{\mathrm{kin}} \mathcal{L}_{\mathrm{kin}}^{\mathrm{FM}}$. Here $\mathcal{L}_{\mathrm{res,impl}}$ includes the reduction and pairing weights below, and χ_{kin} controls kinematic-loss activation. FM regression is retained in all three stages.

[1] Official metric configuration and IDM configuration.

Algorithm 1 DriveAnchor training pipeline (internal configuration). Stage 3 applies the query construction to each scene in a batch.

Require: FM decoder F_θ and reset map T_θ, Energy Field module EF_ϕ, training data $\mathcal{D}$, reward function R

Require: Anchor table $\mathcal{A} = \{a_k^*\}$ (pre-built via FPS from historical corpus), neighbors N, RL weight β

Stage 1 (FM Pretraining): Train F_θ on $\mathcal{D}$ with $\mathcal{L}_{\text{FM}}$ {anchor-conditioned residual FM}

Stage 2 (EF Post-training): Jointly train F_θ and EF_ϕ on 7.5M Stage 2 training samples with $\mathcal{L}_{\text{FM}} + \mathcal{L}_{\text{EF}}$ (encoder frozen); fix ϕ after Stage 2 training {shared frozen features; separate trajectory inputs}

Stage 3 (RL Fine-tuning): Starting from Stage 2 checkpoint with ϕ frozen:

for each training batch $\mathcal{B} = \{(s_i, x_{1,i})\} \subset \mathcal{D}$ **do**

  For each scene i, sample $k_i \sim \text{Uniform}(\{1, \dots, M\})$, set $x_{0,i} = a_{k_i}^*$, and generate $a_i = T_\theta(x_{0,i}, s_i)$ {starts sampled from the full vocabulary}

  Retrieve $\{b_{ij}\}_{j=1}^N \leftarrow \mathcal{N}(a_i)$ from $\mathcal{A}$ {top-k distance lookup on FM output}

  Compute the query-adaptive scale and directions, draw independent z_{ij}, and generate $\widetilde{a}_{ij}$ via Eq. (11)

  Query $r_i = R(s_i, a_i)$ and all $\widetilde{r}_{ij} = R(s_i, \widetilde{a}_{ij})$ {GridMap collision detection}

  Compute $\hat{g}_i$ via Eq. (13) {detach from graph}

  Compute $\mathcal{L}_{\text{FM}}$ from local anchor–GT pairs and noisy interpolation (Eqs. (2)–(3)), with the configured kinematic schedule

  $\mathcal{L} \leftarrow \mathcal{L}_{\text{FM}}(\theta) - \frac{\beta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \text{sg}(\hat{g}_i)^\top a_i$ {optimize FM objective; EF-specific ϕ frozen}

  Update θ using AdamW with gradient $\nabla_\theta \mathcal{L}$

end for

The FM and EF heads use a default regression weight of $w_{\text{reg}} = 0.005$. Let $T = 80$ be the number of trajectory points, distinct from the ZO direction count N. Denoting the reduced primary SmoothL1 loss by ℓ_{top1} and the mean neighborhood SmoothL1 loss by $\bar{\ell}_{50}$, the FM implementation computes

$$\mathcal{L}_{\text{res,impl}} = T(2\ell_{\text{top1}} + \bar{\ell}_{50}), \qquad w_{\text{reg}}\mathcal{L}_{\text{res,impl}} = 0.8\ell_{\text{top1}} + 0.4\bar{\ell}_{50}.$$

The EF regression contribution is $2Tw_{\text{reg}}\ell_{\text{EF}} = 0.8\ell_{\text{EF}}$. These coefficients apply to the already-reduced losses, rather than to individual coordinates independently. Kinematic losses are summed with their own per-term coefficients and receive no additional factor of w_{reg}.

In the main Stage 3 experiment, FM kinematic penalties are enabled only in the final epoch: $\chi_{\text{kin}} = 0$ in earlier epochs and $\chi_{\text{kin}} = 1$ in the final epoch. The RL weight is $\beta = 0.2$.

C ZO Sampling Configurations

Reward queries and budget. Eq. (12) uses zero-based waypoint indices: for $T = 80$, collision rewards are 0–79 and the collision-free reward is 81. Perturbed outputs are scored directly, without another decoder call or flow-time assignment. The unperturbed reward is shared across directions, requiring $N + 1$ queries per output, or $|\mathcal{B}|(N + 1)$ per batch with a common N, before additional filtering or resampling. Delaying a collision also increases reward, so we report collision rates alongside mean reward.

For an FM output $x \in \mathbb{R}^{160}$, let b_j be its jth nearest vocabulary anchor and $s_k(x) = k^{-1} \sum_{j=1}^k \|b_j - x\|_2$. Both samplers set their scale from raw L2 distances in trajectory space. The standard ZO implementation recovers this scale by multiplying its internally normalized distances by $\sqrt{T}$, where $T = 80$ is the number of trajectory points. Thus, the two methods use the same distance units and scale rule; their scales agree when evaluated at the same query with the same neighbor pool.

Neighbor-scaled standard ZO sampling. The standard ZO configuration uses $k = N = 16$. With $\lambda = s_{16}(x)$ and independent $\xi_j \sim \mathcal{N}(0, I_{160})$,

$$u_j = \xi_j / \|\xi_j\|_2, \qquad \widetilde{x}_j = x + \lambda u_j. \tag{18}$$

The perturbations have fixed length λ, so $\mathbb{E}\|\widetilde{x}_j - x\|_2^2 = \lambda^2$.

Anchor-guided sampling. Anchor neighbors follow the nonzero-distance convention in Eq. (10). The anchor-guided configuration uses $k = 16$, matching the main experiments. With $\sigma = s_{16}(x)$ and independent $z_j \sim \mathcal{N}(0, 1)$,

$$u_j = \frac{b_j - x}{\|b_j - x\|_2}, \qquad \widetilde{x}_j = x + \sigma z_j u_j. \tag{19}$$

The scalar Gaussian permits exploration in both signs of each anchor direction, with $\mathbb{E}\|\widetilde{x}_j - x\|_2^2 = \sigma^2$.

Both methods use 17 reward queries per center, including the unperturbed trajectory, without filtering or resampling. Although they share the scale $s_{16}(x)$ at a fixed query with a common neighbor pool, they differ in both direction and radial distribution.

D Baseline Adaptations for the FM Decoder

The following GRPO-inspired and DPO-inspired baselines adapt group-relative updates and preference learning to the anchor-conditioned deterministic FM decoder. They use the surrogate objectives defined below rather than the original likelihood-based objectives. Let x_i be a decoded trajectory, $r_i = R(s, x_i)$ its detached reward, and sg denote stop-gradient. Gradients do not pass through group construction, pair selection, or reward evaluation.

GRPO-inspired group-relative updates. We form candidate groups from seed anchors and nearby vocabulary entries, decode each candidate, and evaluate its reward. We then select the group with the largest reward standard deviation. For its G entries, let σ_r and $\bar{r}$ denote the sample standard deviation and mean reward, respectively. The update is

$$\begin{aligned} A_i &= \frac{r_i - \bar{r}}{\max(\sigma_r, \varepsilon_r)}, \\ g_i^{\text{grp}} &= \frac{1}{G-1} \sum_{j \neq i} \frac{A_j - A_i}{\max(\|x_j - x_i\|_2^2, \varepsilon_d)} (x_j - x_i), \\ \mathcal{L}_{\text{grp}} &= -\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \text{sg}(g_i^{\text{grp}})^\top x_i. \end{aligned} \tag{20}$$

Here $\varepsilon_r = 10^{-4}$ and $\varepsilon_d = 10^{-6}$ are numerical floors; groups with $\sigma_r < \varepsilon_r$ contribute zero. $\mathcal{I}$ includes all entries unless the collision-only option is enabled, in which case it includes only colliding entries. This update retains group-relative reward normalization, using pairwise trajectory directions and a linear surrogate in place of policy-likelihood ratios and clipping. Because it differs from the Gaussian radial estimator used in Stage 3, the expectation identity derived for that estimator does not establish a corresponding identity for this baseline.

DPO-inspired preference regression (Rafailov et al., 2023). Among the sampled decoded trajectories, let $\mathcal{P}$ and $\mathcal{N}$ denote the collision-free and colliding sets, respectively. We pair each colliding trajectory with its nearest collision-free trajectory. For trajectory dimension D, the regression objective is

$$\begin{aligned} j^*(i) &= \arg\min_{j \in \mathcal{P}} \|x_i - x_j\|_2^2, \\ d_i &= D^{-1} \|x_i - \text{sg}(x_{j^*(i)})\|_2^2, \\ \mathcal{L}_{\text{pref}} &= \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \text{sg}(w_i) d_i. \end{aligned} \tag{21}$$

The base weight is $w_i = \min(1, \tau/(d_i + \varepsilon))$, where τ is a positive scale and ε is a numerical floor. An optional severity factor assigns larger weights to earlier collisions. The loss is zero when no valid positive–negative pair exists or when the number of negative samples falls below the configured

minimum. Both the collision-free targets and the weights are detached. This objective encourages higher-quality outputs through regression; it does not use DPO's reference-policy log-ratio or logistic preference objective.

E Network Architecture

Table 3 summarizes the modules. The scene encoder is pretrained and frozen throughout all three stages. FM is trained in every stage, while EF is trained in Stage 2. In the internal experiments, the batched trajectory output before screening has shape $[B, 2{,}398, 160]$, with B scenes and 80×2 coordinates per candidate.

Table 3: Network architecture and training hyperparameters for the main internal experiments.

Module	Parameter	Value
Shared Encoder (frozen throughout training)		
	Input streams	ego, moving/static obstacles, road graph, traffic light
	Transformer layers	6
	Token dim (d_{model})	384
	Attention heads	6
	Output projection	MLP(384 $\rightarrow$ 256)
	Output shape	$[B,\ 634,\ 256]$
Anchor Vocabulary & Shared Projector		
	Vocabulary size M	2,398 trajectories
	Anchor dim	$80 \times 2 = 160$
	Trajectory projector (shared)	MLP(160 $\rightarrow$ 256); shared by FM and EF
	MLP block	Linear $\rightarrow$ LayerNorm $\rightarrow$ GELU $\rightarrow$ Linear $\rightarrow$ Dropout
FM Decoder (trainable in all stages)		
	Cross-attention heads	2
	d_{model}	256
	Residual head	MLP(256 $\rightarrow$ 256) $\rightarrow$ MLP(256 $\rightarrow$ 160)
	FM supervision	Residual regression, target $= x_1 - x_t$ (Eq. (3))
	Inference passes	2 (FM$\times$2)
	After screening	$[B,\ K_{\text{sel}},\ 160]$, with $K_{\text{sel}} = 50$
EF Decoder (Stage 2 trainable)		
	EF input	50-dim request (scene + 49-dim region)
	Scene projector	MLP(1 $\rightarrow$ 128)
	Region projector	MLP(49 $\rightarrow$ 128)
	Fusion projector	concat(128, 128, 256) $\rightarrow$ MLP(512 $\rightarrow$ 256)
	Cross-attention heads	2
	d_{model}	256
	EF output head	MLP(256 $\rightarrow$ 256) $\rightarrow$ MLP(256 $\rightarrow$ 160)
	EF supervision	SmoothL1; branch on good(a_{noisy}) (Eq. (8))
Training Hyperparameters		
	Optimizer	AdamW, lr $= 10^{-5}$, batch 128/GPU
	FM / EF regression weight	0.005
	RL loss weight β	0.2
	ZO neighbors N	16
	RL retrieval	Full-vocabulary top-N; no ϵ filter
	Kinematic constraints	speed, accel, jerk, curvature, lat-accel, lat-jerk

F GridMap-Accelerated Collision Detection

Checking all $N{+}1$ trajectories against every obstacle costs $\mathcal{O}((N{+}1) \cdot T \cdot |\mathcal{O}|)$. We accelerate these checks in two stages. (1) **Build**: rasterize road boundaries and static objects into a coarse 2D grid (10 m/cell) using Bresenham traversal, and store the IDs of overlapping obstacles in each cell. (2) **Query**: project each ego oriented bounding box (OBB) onto the grid, retrieve nearby obstacles, and run precise OBB intersection tests. Once a collision is detected, we skip the remaining waypoints of that trajectory. The grid thus filters obstacle candidates before the geometric checks.

G Stage 2 EF Post-training: Kinematic Constraint Configuration

Empty reference pools. When $\mathcal{A}_{\text{good}}$ is empty, satisfactory noisy inputs retain zero targets; the remaining inputs are masked out of EF regression. If all targets are masked, EF regression contributes zero loss while other training objectives remain active.

We use two penalty forms for the six constraints. For upper-bound constraints (speed, acceleration, curvature, and lateral acceleration), the sigmoid-based penalty is zero below the threshold and bounded above. It has a nonzero gradient just above the threshold but a kink at the boundary:

$$\mathcal{L}_{\max}(x,\, x_{\max}) = \mathbb{E}[\text{ReLU}(\text{sigmoid}(x - x_{\max}) - 0.5)] \tag{22}$$

For smoothness constraints (jerk, lateral jerk), we use a ReLU-squared penalty that strongly penalizes large violations:

$$\mathcal{L}_{\text{smooth}}(x,\, x_{\max}) = \mathbb{E}\big[\max(0,\, |x| - x_{\max})^2\big] \tag{23}$$

Table 4 details the six losses, their ego-history context, threshold constants, and final loss weights after scaling.

Table 4: Kinematic constraint loss configuration for Stage 2 EF post-training. "History" indicates how many ego-history frames are concatenated as input; "threshold" is the per-constraint bound; "weight" is the product of the base weight and the loss scale coefficient. The penalty forms are defined in Eqs. (22) and (23).

Constraint	Ego history	Threshold	Effective weight
Speed upper bound	1 frame	$v_{\max} = 25.0\,\text{m/s}$	$1.0 \times 0.1 = 0.1$
Acceleration upper bound	2 frames	$a_{\max} = 4.5\,\text{m/s}^2$	$0.01 \times 0.01 = 1 \times 10^{-4}$
Jerk smoothness	3 frames	$j_{\max} = 4.5\,\text{m/s}^3$	$0.001 \times 0.1 = 1 \times 10^{-4}$
Curvature upper bound	2 frames	$\kappa_{\max} = 0.4339\,\text{m}^{-1}$	$0.005 \times 0.1 = 5 \times 10^{-4}$
Lateral acceleration bound	2 frames	$a_{\text{lat,max}} = 1.0\,\text{m/s}^2$	$0.01 \times 0.005 = 5 \times 10^{-5}$
Lateral jerk smoothness	3 frames	$j_{\text{lat,max}} = 1.0\,\text{m/s}^3$	$0.001 \times 0.1 = 1 \times 10^{-4}$

H Training Compute

All three stages of the main internal experiments run on A100-80G GPUs with PyTorch 2.3 and AdamW (lr$= 10^{-5}$, weight decay 0.01, batch size 128/GPU). Stage 1 uses 120M samples, and Stage 2 jointly trains EF and FM on 7.5M samples. Stage 3 performs zeroth-order reward fine-tuning on 2.5M samples using 64 GPUs. Table 5 reports elapsed training time and GPU·hours for each stage.

Table 5: Training compute per stage for the main internal experiments.

Stage	GPUs	Wall time	GPU·Hours
Stage 1 (FM Pretraining)	32×A100-80G	28.2 h	902
Stage 2 (EF Post-training)	32×A100-80G	16.9 h	541
Stage 3 (RL Fine-tuning)	64×A100-80G	3.8 h	243

I Ablation Studies

I.1 Residual-Screened Candidate Quality

The following table gives the full residual-screening comparison referenced in Section 4.6. The statistics are computed before the downstream selector.

I.2 Stage 3 Transfer across Maneuver Subsets

We fine-tune Stage 3 on lane-change (Lac) or junction (Jct) subsets and evaluate transfer to the other subset. Both maneuver types are present in the shared base-stage training data, so transfer

Table 6: Results for the Top-K_{sel} candidates selected in each scene by the second-call residual norm (Section 3.4), before the downstream selector. Short- and long-horizon collision rates are fractions. Each column corresponds to the same configuration in Table 2. The first two columns use the original prior; the last three use EF initialization. Bold indicates the best result within each group.

Metric	**FM**×2	**FMRL**×2	**EF**×1+**FM**×1	**EF**×1+**FMRL**×1	**EF**×1+**FMRL**×2
Short-horizon ↓	0.1373	**0.0047**	0.2276	0.0264	**0.0062**
Long-horizon ↓	0.2780	**0.0224**	0.4024	0.0842	**0.0441**
Mean reward ↑	71.05	**80.48**	63.22	78.54	**80.14**

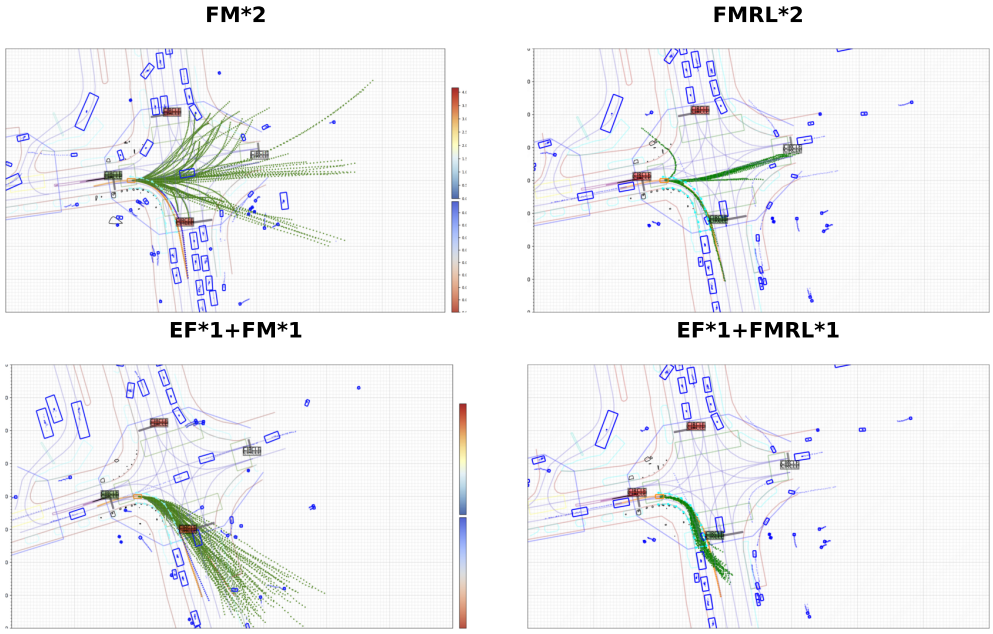


Figure 3: Example trajectories from the residual-selected Top-K_{sel} candidates under four configurations, before the downstream selector. Spatial coordinates are in meters.

here concerns the fine-tuning subset. Results are collision rates after one and two FM calls (%, mean±SD over three repetitions with different noise samples). GRPO-inspired and DPO-inspired use the deterministic FM surrogates in Appendix D rather than the original likelihood objectives.

Table 7: Collision rates after Stage 3 transfer between maneuver subsets on 10,000 evaluation scenarios (%, mean±SD over three repetitions with different noise samples). Baselines use the adaptations described in the text.

Direction	**ZO (ours)**	GRPO-inspired	DPO-inspired
Jct→Lac, FM×1	**4.41±0.60**	6.65±0.96	7.04±0.61
Jct→Lac, FM×2	**1.77±0.55**	3.75±0.90	3.72±0.59
Lac→Jct, FM×1	**3.49±0.68**	7.07±0.90	5.02±1.05
Lac→Jct, FM×2	1.57±0.58	3.80±0.50	**1.44±0.43**

Among the three compared update rules, ZO achieves the lowest mean collision rate in three of the four combinations of transfer direction and call count. DPO-inspired is lowest for Lac→Jct with two calls (1.44%, compared with 1.57% for ZO). Under the shared base-stage initialization, the strongest transfer result therefore depends on the maneuver direction and decoding budget.

I.3 Anchor-Guided versus Standard ZO Exploration

Tables 8 and 9 report lightweight ablations on a smaller, intersection-specific dataset containing approximately 100,000 scenarios. Evaluation uses 1,000 held-out scenarios excluded from training. These experiments use a shared Stage 2 initialization and a separate reward-fine-tuning setting from the main experiment; their absolute collision rates are therefore not directly comparable to Table 2.

Table 8 tests whether the complete anchor-guided sampling configuration improves on standard ZO. Both use $N = 16$, 17 reward queries per center, and the same neighbor-distance scale rule. Evaluation generates candidates with one EF call followed by one FM call, using FMRL after reward fine-tuning. Anchor-guided ZO improves all four metrics relative to Stage 2 initialization and standard ZO; standard ZO lowers collision but increases trajectory error. This comparison supports the sampling configuration, but does not isolate direction selection: the variants differ in both direction distribution and radial perturbation, with standard ZO using a fixed radius and anchor-guided ZO using a signed Gaussian scalar (Appendix C).

Table 8: ZO exploration comparison on 1,000 held-out scenarios from the smaller intersection-specific dataset, with statistics aggregated across ranks. Candidate generation uses EF$\times$1+FM$\times$1 before fine-tuning and EF$\times$1+FMRL$\times$1 afterward. Both ZO variants start from the shared Stage 2 checkpoint and use $N = 16$. Long-horizon static collision rates ($R < 80$) are reported for all 2,398 candidates and the Top-50 candidates selected per scene by residual norm, with the additional screening call described in Section 3.4.

Metric ↓	Before fine-tuning	Standard ZO $N = 16$	Anchor ZO $N = 16$
minADE@80 (m)	1.65	1.78	**1.50**
minFDE@80 (m)	3.97	4.34	**3.47**
All-candidate collision (%)	3.06	1.97	**0.28**
Top-50 collision (%)	2.80	1.11	**0.44**

I.4 Prior, Vocabulary, and Neighbor-Count Ablations

Anchor versus Gaussian prior. Both variants start from the same main Stage 2 checkpoint and are separately fine-tuned on the same Stage 3 data, reusing Stages 1 and 2. Trajectory anchors achieve minADE/FDE@30 of $0.30/0.73$ m, compared with $0.87/1.32$ m for the Gaussian prior. This comparison evaluates the choice of prior when adapting the same pretrained decoder in Stage 3.

Vocabulary size and construction. Using EF$\times$1+FMRL$\times$1 on the 1,000 held-out scenarios from the smaller intersection-specific dataset, we vary vocabulary construction (FPS or K-means) and size M. Each vocabulary configuration undergoes a separate Stage 3 reward-fine-tuning run from the shared Stage 2 initialization using its corresponding vocabulary; the vocabulary is not merely replaced at evaluation time. Increasing the FPS vocabulary from 100 to 2,398 anchors lowers long-horizon collision from 0.54% to 0.28% (Table 9). FPS and K-means perform similarly at matched sizes; increasing from 2,000 to 2,398 anchors yields a 0.01-percentage-point gain.

Table 9: Vocabulary construction and size ablation on 1,000 held-out scenarios from the smaller intersection-specific dataset with EF$\times$1+FMRL$\times$1. Each vocabulary configuration is separately reward-fine-tuned from the shared Stage 2 initialization. Values are long-horizon static candidate collision rates ($R < 80$, %) before the selector, reported as mean$\pm$SD over three repetitions with different noise samples.

M	100	500	1,000	2,000	2,398
FPS (ours)	0.54 ± 0.03	0.38 ± 0.01	0.33 ± 0.01	0.29 ± 0.01	$\mathbf{0.28\pm0.01}$
K-means	0.47 ± 0.02	0.37 ± 0.01	0.32 ± 0.01	0.29 ± 0.01	–

Number of anchor neighbors. The $N = 4$ and $N = 8$ runs use a smaller ablation training set, while $N = 16$ reuses the main training setting; all configurations are evaluated on the same 10,000 scenes. Increasing N from 4 to 8 lowers short- and long-horizon collision from $27.52\%/56.75\%$

to 3.04%/7.69% (Table 10). This change also increases the query budget and changes the direction set and distance scale; the $N = 16$ main result uses different training settings and is listed separately as a reference.

Table 10: Ablation on the number of anchor neighbors N, initialized from the main experiment's trained Stage 2 checkpoint containing EF and FM. Collision rates are measured on the same 10,000 held-out scenes; arrows compare $N = 8$ and $N = 4$ trained with the smaller dataset. The N=16 row reuses the FMRL×2 result from Table 2 as a reference and uses different training data.

N	Short-horizon	Long-horizon	Mean reward
4	0.2752	0.5675	59.10
8	0.0304↓	0.0769↓	78.25↑
16 (main-table reference)	0.0294	0.0715	78.46

Vocabulary sampling and kinematic feasibility. We use all 2,398 anchors in the internal trajectory vocabulary as sampling centers. Each anchor contains 80 XY waypoints. For each center x, we retrieve its 16 nearest neighbors from the full vocabulary, excluding itself, and construct unit directions $u_j = (b_j - x)/\|b_j - x\|_2$ in flattened trajectory space. We evaluate Gaussian perturbations

$$y = x + \lambda z u_j, \qquad z \sim \mathcal{N}(0, 1), \qquad \lambda \in \{1, 5, 10, 20\}.$$

Here λ controls the perturbation scale of the entire trajectory; the realized perturbation distance is $\lambda|z|$. All original anchors are included without feasibility filtering.

We evaluate the perturbed trajectories using a time interval $h = 0.1\,\mathrm{s}$ and the following discrete constraints:

$$\max_t \left\| \frac{y_{t+1} - y_t}{h} \right\|_2 \le 25\,\mathrm{m/s}, \qquad \max_t \left\| \frac{y_{t+2} - 2y_{t+1} + y_t}{h^2} \right\|_2 \le 4.5\,\mathrm{m/s^2}.$$

Only differences within the trajectory are checked. For each center–direction pair, we compute the feasible interval along $y = x + qu_j$ and integrate its Gaussian probability analytically. Table 11 reports the mean probability over all $2{,}398 \times 16$ center–direction pairs.

Table 11: Mean probability of satisfying both discrete speed and acceleration constraints under Gaussian perturbations around all 2,398 anchors, using 16 nearest-neighbor directions per anchor.

Perturbation scale λ	1	5	10	20
Feasibility probability (%)	97.43	96.81	94.38	84.85

The mean feasibility probability decreases as the perturbation scale increases, from 97.43% at $\lambda = 1$ to 84.85% at $\lambda = 20$. This diagnostic measures discrete speed and acceleration feasibility, not collision avoidance or closed-loop safety.

I.5 Additional Training Diagnostics

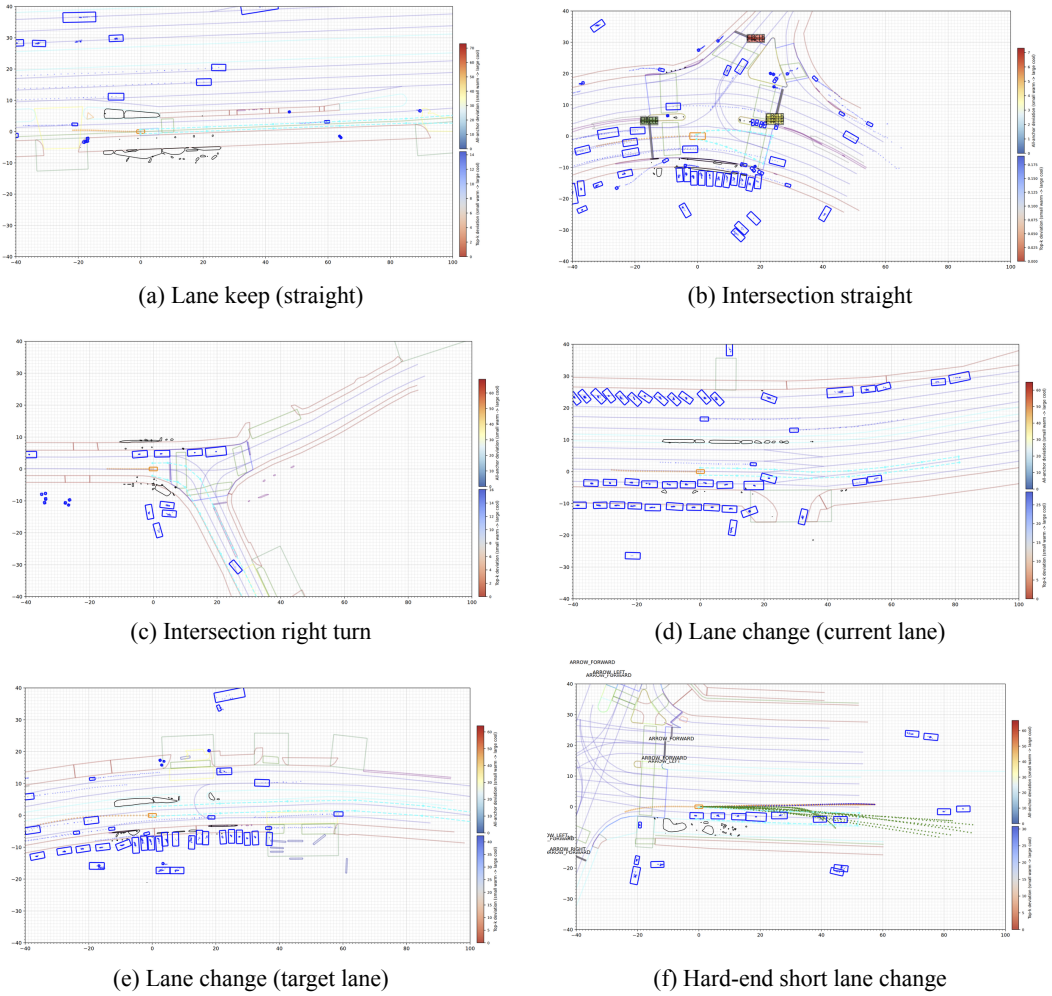
RL training progress and decoding calls. Each configuration applies EF once, then calls FM consecutively with $t = 0$ reset at every call. The table reports collision percentages over all generated candidates before Top-K_{sel} filtering. Epochs correspond to checkpoints from one training run, rather than independent training seeds.

Table 12: Long-horizon static collision rates ($R < 80$, %) over all generated candidates on 10,000 evaluation scenarios. Each configuration applies one EF call, followed by the number of FM calls shown in the column heading; post-RL rows use the corresponding FMRL checkpoint.

Epoch	FM ×1	FM ×2	FM ×3	FM ×4	Epoch	FM ×1	FM ×2	FM ×3	FM ×4
Before RL	26.71	20.62	16.37	13.40	–	–	–	–	–
0	18.19	10.36	5.69	3.60	6	2.02	0.55	0.37	0.35
1	6.72	1.26	0.57	0.43	7	1.90	0.52	0.36	0.35
2	3.54	0.88	0.52	0.41	8	1.93	0.52	0.35	0.34
3	2.96	0.78	0.54	0.44	9	1.91	0.51	0.34	0.34
4	2.60	0.70	0.41	0.36	10	1.94	0.53	0.35	0.34
5	2.27	0.67	0.42	0.36	11	1.91	0.52	0.34	0.34

J Additional Figures

J.1 EF Polygon Presets



(a) Lane keep (straight)

(b) Intersection straight

(c) Intersection right turn

(d) Lane change (current lane)

(e) Lane change (target lane)

(f) Hard-end short lane change

Figure 4: EF polygon presets for six driving scenarios. The geometric label checks entry into the corridor and allows at most one exit through the designated edge. An optional label-construction check requires the trajectory to remain inside at a specified time. The hard-end short lane-change preset (bottom right) limits the requested maneuver to a short spatial window. Spatial coordinates are in meters.

EF request encoding. The request is encoded as 50 values: one scene type, 16 interleaved $(x, y, \text{enterable})$ vertex triples, and one region-expansion flag. The 49-dimensional region block, which includes the expansion flag, is projected separately from the scene type. Vertices 0–7 trace the left boundary forward, and vertices 8–15 trace the right boundary backward. The designated forward exit is therefore edge 7, connecting vertices 7 and 8 under zero-based indexing. For a lane-change request that starts outside the polygon, both endpoints of the first entry edge must be marked enterable; a trajectory that starts inside automatically passes this entry check. The feature has no separate exit-index or time scalar. The optional time gate in Eq. (7) is applied when constructing labels.

J.2 Training and Visualization

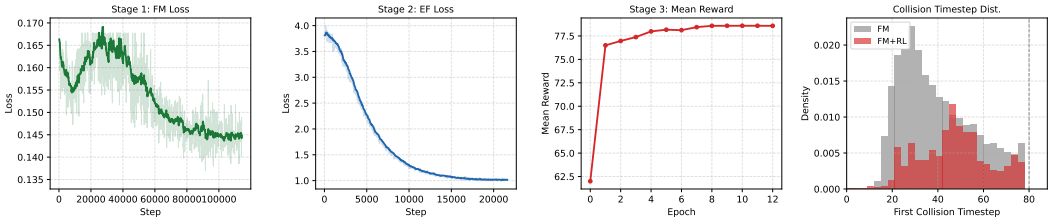


Figure 5: Training diagnostics. From left to right: Stage 1 FM loss, Stage 2 total loss, Stage 3 mean reward, and first-collision time-step histograms. The reward curve contains 13 saved evaluations, whose indices differ from the epoch labels in Table 12. The histograms use a separate 10,000-scene diagnostic set; each model evaluates 2,398 candidates per scene, corresponding to 23,980,000 candidate evaluations, with a common count scale. They show collision indices 0–78, omitting index 79 and the no-collision value 81; index 80 is unused. They do not represent the main evaluation set in Table 2.

Input perturbations. The vocabulary defines a Voronoi partition (Figure 6). For each selected anchor a in the $d = 2T$ dimensional training representation, we use midpoint nearest-neighbor tests to construct an approximate neighbor set $\mathcal{B}(a)$ and set the proposal scale. Let $d_{\min}(a)$ denote the distance to the nearest other anchor in the full vocabulary, and $d_{\max}(a)$ the largest distance to a neighbor in $\mathcal{B}(a)$. The radii are

$$R_{\max} = \sqrt{d}, \qquad R_{\text{in}} = \min\{0.49 d_{\min}(a), R_{\max}\},$$
$$R_{\text{prop}} = \max\{R_{\text{in}}, \min\{0.5 d_{\max}(a), R_{\max}\}\}.$$

Sample independent $\xi \sim \mathcal{N}(0, I_d)$ and $U \sim \text{Uniform}(0, 1)$, then propose $y = a + R_{\text{prop}} U^{1/d} \xi / \|\xi\|_2$, which is uniform within the proposal ball. Accept y only if an exact Euclidean 1-NN query over the full fixed vocabulary returns a; otherwise retry, up to eight attempts. If every attempt fails, sample uniformly from the ball of radius R_{in}. The approximate set $\mathcal{B}(a)$ therefore determines the proposal scale, while the full-vocabulary query verifies cell membership. Missing approximate neighbors cannot weaken this acceptance check. The fallback also remains in the source cell because $R_{\text{in}} \le d_{\min}(a)/2$. This guarantee applies only to the perturbed start, not to all interpolated x_t or to driving safety. The proposal radius approximates the cell's scale rather than its boundary, and finite rejection sampling with fallback is not uniform over the entire cell. The figure illustrates these geometric regions, not empirical sample densities; Figure 7 shows trajectory examples from the local-noise comparison.

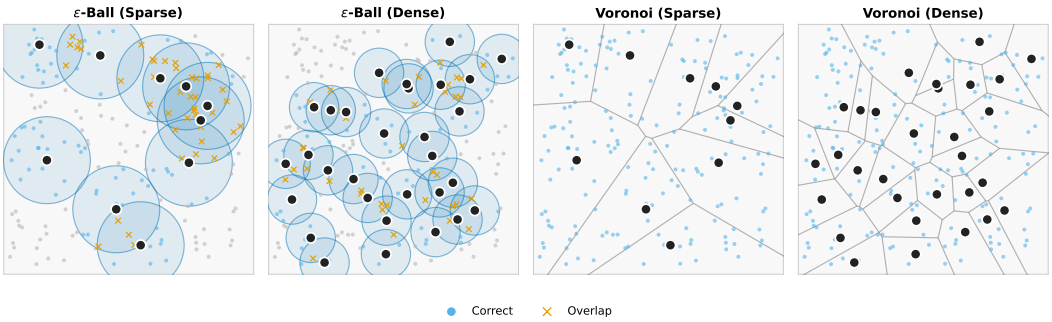
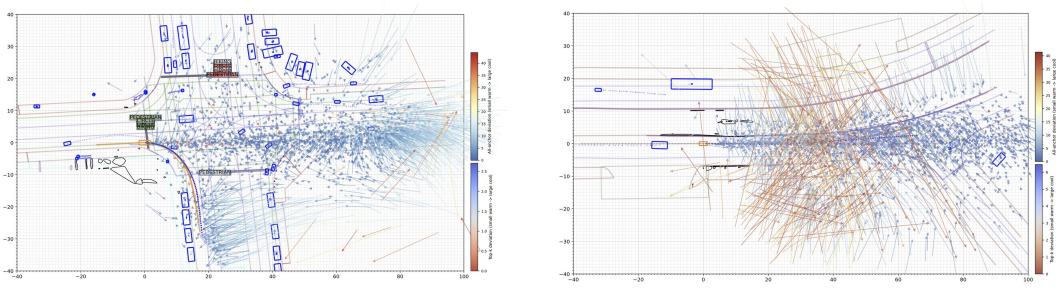


Figure 6: ϵ-ball regions and Voronoi partitions for dense and sparse anchors. Blue points have a unique assignment, orange crosses have multiple assignments, and gray points are uncovered; white-bordered circles mark anchors. ϵ-balls leave overlaps and gaps, whereas nearest-anchor assignment, with consistent tie-breaking, provides complete, non-overlapping coverage in both anchor regimes.



(a) With local input noise

(b) Without local input noise

Figure 7: Stage 1 examples from a separate local-noise comparison. The left configuration uses ball perturbations and produces distinct trajectory modes; the right omits these perturbations and produces more concentrated outputs. These qualitative examples do not isolate the effect of the Voronoi acceptance rule or quantify mode coverage. Spatial coordinates are in meters.

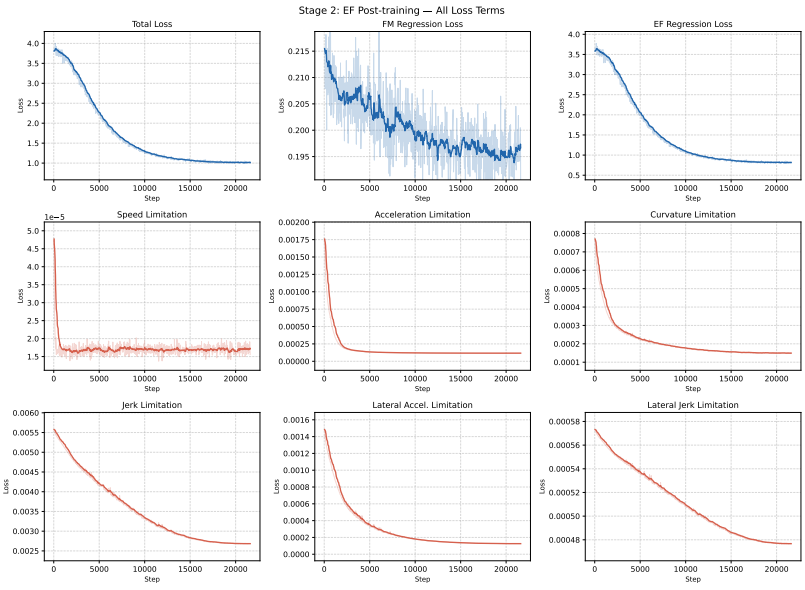


Figure 8: Stage 2 EF post-training losses over 21.6K steps using 7.5M samples. **Top row** (blue): total, FM regression, and EF regression losses. **Bottom two rows** (red): six kinematic losses for violations of speed, acceleration, curvature, jerk, lateral acceleration, and lateral jerk limits.

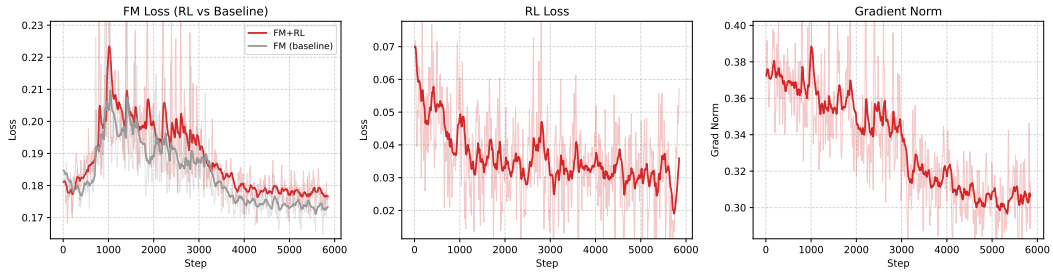


Figure 9: Stage 3 training over 5.8K steps. **Left**: FM regression loss for the RL-fine-tuned model (red) and the frozen pre-Stage 3 baseline (gray); the two remain close. **Center**: RL surrogate loss ($\mathcal{L}_{\text{RL}}$). **Right**: gradient norm. Stage 3 combines FM regression with the black-box static collision reward and enables kinematic losses only in the final epoch.

K Expectation of the Anchor-Guided Estimator

For the current query, write $\lambda = \sigma_i$. The following analysis conditions on its retrieved neighbors, directions, and scale, treating them as fixed and excluding derivatives through their dependence on the query. Fix a scene s, query $x = a_i$, and nonzero neighbor directions $u_j = u_{ij}$. Assume that the reward is bounded and measurable. For $h_j(q) = R(s, x + qu_j)$ and the Gaussian density $\phi_\lambda(v) = (2\pi\lambda^2)^{-1/2} \exp(-v^2/(2\lambda^2))$, write

$$\mathcal{Q}_{j,\lambda}(\alpha) = \int_{\mathbb{R}} h_j(q)\phi_\lambda(q - \alpha)\, dq.$$

Differentiating the Gaussian kernel is justified by boundedness of h_j and integrability of the kernel derivative. At $\alpha = 0$,

$$\mathcal{Q}'_{j,\lambda}(0) = \int h_j(q)\frac{q}{\lambda^2}\phi_\lambda(q)\, dq = \frac{1}{\lambda}\mathbb{E}_z[h_j(\lambda z)z] = \frac{1}{\lambda}\mathbb{E}_z[(h_j(\lambda z) - h_j(0))z].$$

This proves Eq. (15) without differentiating the collision oracle. The factor z arises from differentiating the Gaussian kernel and weights the reward difference by the signed radial sample.

Meaning and ascent of the local surrogate. For this appendix derivation, we rename the query a_i as x and the retrieved directions u_{ij} as u_j. Eq. (16) can then be written as $H_x(\Delta) = N^{-1}\sum_j \mathcal{Q}_{j,\lambda}(u_j^\top \Delta)$. The chain rule and the kernel identity above give

$$\nabla_\Delta H_x(0) = \frac{1}{N}\sum_j \mathcal{Q}'_{j,\lambda}(0)u_j = \mathbb{E}[\hat{g} \mid s, x, \{u_j\}].$$

Convolving a bounded reward with a one-dimensional Gaussian gives a smooth function. With the query and direction pool fixed, H_x is therefore smooth in Δ. Let $h = \nabla H_x(0)$. Then

$$H_x(\eta h) = H_x(0) + \eta\|h\|^2 + O(\eta^2).$$

Thus, following the expected direction in trajectory space locally increases H_x whenever $h \neq 0$ and $\eta > 0$ is sufficiently small. Individual noisy updates need not improve it monotonically.

Connection to parameter updates. Fix a training batch $\mathcal{B}$ and reference parameters θ_0. Write $a_i(\theta) = T_\theta(x_{0,i}, s_i)$ and $x_i = a_i(\theta_0)$, and hold each query's retrieved neighbors, directions, and scale fixed. The corresponding parameter-space surrogate for this batch is

$$\mathcal{H}_{\theta_0,\mathcal{B}}(\theta) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} H_{x_i}\big(a_i(\theta) - x_i\big).$$

At a differentiability point of the decoder, the chain rule gives the exact conditional identity

$$-\mathbb{E}_z[\nabla_\theta \mathcal{L}_{\text{RL}}(\theta_0)] = \frac{1}{|\mathcal{B}|}\sum_i D_\theta a_i(\theta_0)^\top \nabla H_{x_i}(0) = \nabla_\theta \mathcal{H}_{\theta_0,\mathcal{B}}(\theta_0).$$

The stop-gradient implementation therefore estimates the gradient of this fixed batch surrogate, accounting for the shared parameters. As the reference parameters change, the surrogate also changes; the identity does not establish one global objective for the entire training process. Let $g = \nabla \mathcal{H}_{\theta_0,\mathcal{B}}(\theta_0)$ and $\ell = \nabla \mathcal{L}_{\mathrm{FM}}(\theta_0)$, where the latter includes the retained kinematic term. For an ordinary gradient step, the expected combined direction is $d = \beta g - \ell$, and the first-order change in the surrogate is

$$g^\top d = \beta \|g\|^2 - g^\top \ell.$$

A strict first-order gain requires positive alignment between the surrogate gradient and the combined update direction. A finite-step improvement bound in expectation additionally requires smoothness, a sufficiently small step, and controlled estimator variance.

Define the reward-survival event $C_t(y) = \{R(s,y) > t\}$ for $t = 0, \ldots, T$. For $t < T$, it means no collision through predicted-waypoint index t. The sentinel convention gives $C_T = C_{T-1}$, both denoting safety over all T waypoints; T is not an additional checked waypoint. The integer collision-step reward in Eq. (12) admits the tail-sum representation

$$R(s,y) = \sum_{t=0}^{T} \mathbf{1}\{C_t(y)\}, \qquad H_x(\Delta) = \frac{1}{N} \sum_{j=1}^{N} \sum_{t=0}^{T} \Pr_z\{C_t(x + (u_j^\top \Delta + \lambda z) u_j)\}.$$

Thus, H_x sums collision-free probabilities over the checked waypoints, with one additional full-horizon term from the sentinel convention, so it also rewards delaying a collision. The full-horizon collision-free output fraction under this oracle is $J_{\text{safe}}(\theta) = \mathbb{E}_{s,x_0} \mathbf{1}\{C_T(T_\theta(x_0, s))\}$. Increasing the collision-step surrogate does not necessarily increase the probability of remaining collision-free over the full horizon. If the reward were instead $S(s,y) = \mathbf{1}\{C_T(y)\}$, the same estimator identity would hold, and the local surrogate would equal the average collision-free candidate probability. This alternative illustrates the identity; it is not the reward used in our implementation.

Conditional variance. If $|R(s,y) - R(s,x)| \le B$ for all queried y, independent radial draws give

$$\mathbb{E}\|\hat{g}_i - \mathbb{E}\hat{g}_i\|_2^2 \le \frac{B^2}{N\lambda^2}.$$

Here all expectations are conditional on the fixed query and direction pool. For the reward in Eq. (12), the conservative bound $B = T + 1$ also covers collisions at step zero. For a composite reward with additional kinematic penalties, the expectation identity applies under the stated boundedness assumption, but the collision-probability interpretation and the choice $B = T + 1$ do not transfer. A bound for the actual composite reward must be established separately. Using m independent draws per direction changes the denominator to $Nm\lambda^2$ and requires $Nm + 1$ oracle queries.

Smooth-reward special case. If $R(s,\cdot)$ has a globally bounded third derivative with operator norm at most L_3, Taylor expansion gives

$$\mathbb{E}\hat{g}_i = G_i \nabla R(s, a_i) + b_i, \qquad G_i = \frac{1}{N} \sum_j u_{ij} u_{ij}^\top, \qquad \|b_i\|_2 \le \tfrac{1}{2} L_3 \lambda^2.$$

The factor $1/2$ follows from $\mathbb{E}[z^4]/6 = 1/2$. G_i is positive semidefinite with trace one and rank at most N; it is generally not an orthogonal projector. The rank bound follows because its range lies in $\mathrm{span}\{u_{ij}\}_{j=1}^{N}$. This is a restriction on the query-local output-space signal, not on all effects of a shared-parameter network update. This Taylor statement does not apply to the raw collision-step reward.

Scope of the result. Eq. (16) defines a scalar objective for a fixed query. Updating the query and retrieving new neighbors changes this local objective, and the resulting objectives need not correspond to one globally smoothed function. Moreover, probing each direction separately need not capture the interactions in the combined trajectory $x + \Delta$. Ascent of the local surrogate alone therefore guarantees neither an increase in the true expected reward nor a larger safe-output fraction. Global ZO-SGD convergence would require a separate argument.

Exploration subspace. For a fixed query, the estimate lies in the span of the retrieved directions, whose dimension is at most 16 when $N = 16$ in the 160-dimensional trajectory space. Orthogonal directions receive no direct signal from this query. The following feasibility analysis concerns speed and acceleration constraints; it does not establish obstacle avoidance or feasibility for every model-generated query.

Directional feasibility probability. Convex combinations of trajectories satisfying the same convex finite-difference constraints are feasible, but signed Gaussian extrapolations need not remain in their convex hull. More generally, for any query x and constraint set $\mathcal{F}$, a certified interval $\{x + qu_j : L_j \le q \le U_j\} \subseteq \mathcal{F}$ gives

$$\Pr\{x + \lambda z u_j \in \mathcal{F}\} \ge \Phi(U_j/\lambda) - \Phi(L_j/\lambda). \tag{24}$$

Here Φ is the standard normal CDF. Equality holds when $[L_j, U_j]$ is the entire feasible intersection with the line; an empty intersection has probability zero. When $L_j \le 0 \le U_j$, write $L_j = -r_j^-$ and $U_j = r_j^+$ for nonnegative radii to obtain a violation bound $\Phi(-r_j^-/\lambda) + \Phi(-r_j^+/\lambda)$. The bound follows because $L_j \le \lambda z \le U_j$ implies feasibility. Averaging over directions and queries bounds the expected fraction of feasible candidates, but individual batches may differ from that expectation.

For position trajectories with uniform step h, internal speed and acceleration constraints take the form

$$\|D_1 y\|_{2,\infty} \le 25, \qquad \|D_2 y\|_{2,\infty} \le 4.5,$$

where D_1 and D_2 include h^{-1} and h^{-2}, respectively. Each time-indexed constraint along $y = x + qu_j$ is a scalar quadratic inequality $\|b_t + qc_t\|^2 \le c_{\max}^2$. Intersecting its feasible intervals over time and constraint type gives $[L_j, U_j]$ exactly, or an empty set. No isotropy or Gaussian assumption about the anchor directions is needed; the Gaussian is the radial sampling construction. The argument also applies to third-difference jerk constraints if those definitions are adopted, but jerk is not evaluated here.