FIELDDRIVE: FIELD-GUIDED ONE-STEP TRAJECTORY GENERATION AND REWARD OPTIMIZATION FOR AUTONOMOUS DRIVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative models have been introduced as trajectory decoders for autonomous driving to model the conditional multimodal distribution of reasonable driving behaviors in complex traffic scenes. However, existing generative decoders typically rely on multiple denoising or flow-integration steps, resulting in high inference overhead. Meanwhile, existing reinforcement learning methods are mainly designed around multi-step generation chains and are structurally mismatched with one-step generators. Combining one-step multimodal trajectory decoding with the optimization capability of reinforcement learning remains a central challenge in generative autonomous driving planning. To address this challenge, we propose FieldDrive, a Drift-based one-step trajectory decoder with a corresponding reinforcement learning method. We cluster a trajectory vocabulary and select high-scoring trajectories to construct a feasible trajectory distribution, which guides the generated trajectory distribution by the Drift Field Vector. We train a critic network to provide a Reward Field Vector that directs the model toward high-reward regions. Theoretically, we show that the Drift Field Vector and Reward Field Vector share a common origin: both arise as optimization directions of distributional energy function. Through a two-stage training paradigm, the model explores different distributions of reasonable trajectories and progressively converges to high-reward regions through reinforcement learning, while enabling efficient decoding with a single forward pass at inference time. Experiments on NAVSIM v1 and v2 demonstrate that FieldDrive enables efficient and high-quality trajectory decoding.

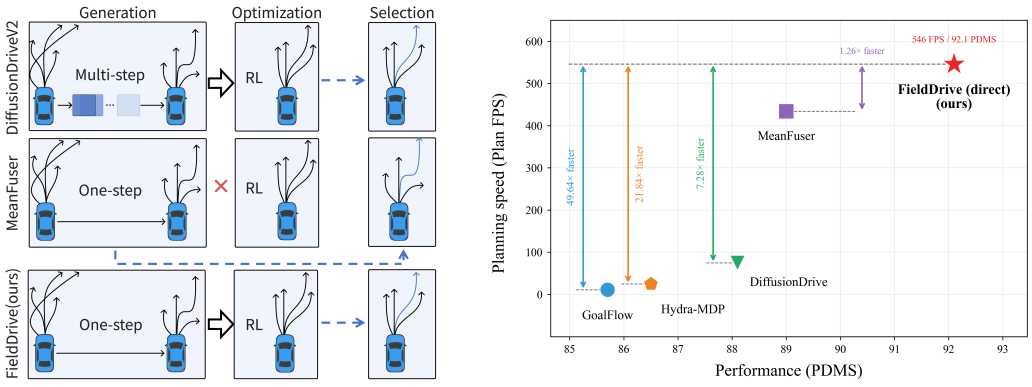
1 INTRODUCTION

End-to-end autonomous driving aims to generate future ego trajectories from sensor observations, navigation commands, and ego state within a unified learning framework. Planning-oriented joint modeling integrates perception and prediction with planning so that upstream representations are learned for the driving objective(Hu et al., 2023).Complex traffic scenes often allow multiple reasonable driving behaviors, and the same scene may correspond to different speed choices, maneuver choices, and local avoidance behaviors. The planning model therefore needs to sufficiently explore the feasible trajectory space and finally produce a trajectory that best satisfies the planning objective (Liao et al., 2025; Xing et al., 2025).

Recent studies have introduced generative models as trajectory decoders for autonomous driving to learn scene-conditioned distributions of feasible trajectories. These methods mainly follow two paradigms: (1) diffusion-based planners, which iteratively transform an initial noise sample into a trajectory through conditional denoising (Chi et al., 2025; Liao et al., 2025; Zheng et al., 2025); and (2) flow-matching planners, which learn a continuous vector field that transports noise toward the trajectory distribution under scene or goal conditions (Xing et al., 2025; Trivedi et al., 2026). Owing to their ability to model trajectory distributions, generative models can adequately represent multiple feasible driving modes.

Recent studies have explored improving the efficiency and optimization capability of generative trajectory decoders. DiffusionDriveV2 (Zou et al., 2025) incorporates reinforcement learning into

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.



(a) Trajectory generation (b) Planning speed

Figure 1: Diffusion/FlowMatching require multiple inference steps to transform noise into a trajectory, while one-step denoising lacks a corresponding reinforcement-learning method. FieldDrive fills this gap and enables efficient and high-quality trajectory decoding.

the generative planning process, further shifting generated trajectories toward high-reward regions. MeanFuser (Wang et al., 2026) introduces one-step denoising into autonomous driving and substantially accelerates trajectory decoding. Nevertheless, diffusion-based planners inherently require multiple denoising steps because their generation process is implemented as an iterative reverse process. One-step flow-based methods such as MeanFlow (Geng et al., 2026) improve decoding efficiency, but their deterministic formulation does not explicitly expose the Markovian transition structure or stochastic differential process underlying multi-step generative models. Consequently, they do not directly provide the action probabilities required by policy-gradient methods such as GRPO (Shao et al., 2024) and PPO (Schulman et al., 2017). This creates a fundamental gap between fast one-step decoding and reinforcement-learning method.

Recently, Drift (Deng et al., 2026) was proposed to formulate generation directly as a distribution-matching process. Drift enables one-step generation while naturally supporting reward guided optimization. Motivated by these properties, we proposed **FieldDrive**, a drift style autonomous driving planning model and its corresponding RL method. We first apply Drift to end-to-end trajectory decoding. Specifically, we cluster trajectory candidates into a trajectory vocabulary and select high-scoring trajectories under each scene condition to construct a scene-conditioned feasible trajectory distribution. The Drift Field Vector combines attraction toward positive samples with repulsion from generated samples to pull the generated trajectory distribution toward the feasible region. We then analyze the theoretical origin of Drift Field Vector and propose the Reward Field Vector as the trajectory distribution optimization direction. From the perspective of distributional energy functions, the Drift Field Vector can be interpreted as the normalized direction that reduces the maximum mean discrepancy between distributions, whereas the Reward Field Vector corresponds to the direction that increases the expected reward of the generated distribution. Based on this interpretation, we design an actor–critic architecture to estimate the Reward Field Vector and a two-stage training paradigm that first enables the model to explore feasible trajectories and then drives it toward high-reward regions.

We evaluate FieldDrive on the NAVSIM v1 and v2 benchmarks, where it achieves state-of-the-art performance in both direct-decoding and scorer-assisted settings. FieldDrive achieves a PDMS of 92.1 on NAVSIM v1 and an EPDMS of 89.7 on NAVSIM v2. Meanwhile, it directly generates a single final trajectory at a decoding speed of 546 FPS under on a single NVIDIA H20 GPU. These results demonstrate that FieldDrive can simultaneously maintain high-quality trajectory planning and efficient one-step inference, validating the effectiveness of our method.

Our main contributions are as follows:

1. **Drift-based one-step trajectory decoding.** We introduce the Drift one-step generative model into end-to-end autonomous driving, enabling efficient trajectory decoding with a single forward pass. By supervising the generator with multiple high-quality positive tra-

jectories, the model learns to explore a diverse distribution of feasible driving behaviors, providing the exploration basis for subsequent reward-guided optimization.

2. **Field-guided reinforcement learning for Drift.** We establish a unified energy-field view of distribution matching and reward optimization. Based on this view, we derive a Reward Field Vector from the critic's trajectory gradient and combine it with the Drift Field Vector in a two-stage actor-critic training framework. This reinforcement learning procedure further concentrates the explored trajectory distribution in high-reward region without relying on the multi-step likelihood structure.

3. **Efficient direct trajectory planning.** FieldDrive achieves a PDMS of 92.1 on NAVSIM v1 and an EPDMS of 89.7 on NAVSIM v2 in high planning speed. These results demonstrate that FieldDrive effectively combines the exploration capability of generative models with the optimization capability of reinforcement learning, enabling efficient direct trajectory planning.

2 Related Work

2.1 Generative Autonomous Driving Planning

Generative models have attracted substantial attention because they can fit diverse distributions of plausible outcomes rather than produce a single deterministic prediction. Diffusion models formulate generation as a progressive denoising process (Ho et al., 2020; Song et al., 2020b;a), whereas flow-matching methods learn a vector field whose integration transports samples from an initial noise distribution to the target distribution (Lipman et al., 2022; Liu et al., 2022). These modeling paradigms have been increasingly applied to autonomous driving. Diffusion Policy adapts generative modeling to conditional action generation (Chi et al., 2025), and DiffusionDrive and Diffusion Planner extend diffusion-based generation to end-to-end trajectory prediction using truncated diffusion, multimodal anchors, and flexible guidance (Liao et al., 2025; Zheng et al., 2025). GoalFlow applies goal-conditioned flow matching to multimodal driving trajectories, while Adaptive Time Step Flow Matching and MeanFuser improve the speed–quality trade-off through adaptive or one-step generation (Xing et al., 2025; Trivedi et al., 2026; Wang et al., 2026). Nevertheless, one-step flow decoders remain deterministic endpoint maps and do not expose the stochastic transition structure required by many policy-gradient methods. Drift instead evolves the pushforward distribution during training toward a target distribution and naturally supports one-step inference (Deng et al., 2026); this distribution-level field provides a direct basis for reward-guided optimization, motivating our FieldDrive formulation.

2.2 Reinforcement Learning for Generative Models

Existing reinforcement-learning methods for generative models mainly derive policy updates from intermediate sampling processes. DDPO models diffusion denoising as a Markov decision process and applies PPO-style updates to the product of per-step transition probabilities, an idea further explored for diffusion action policies by DPPO and for autonomous driving by Gen-Drive and DiffusionDriveV2 (Black et al., 2024; Ren et al., 2025; Huang et al., 2025; Zou et al., 2025). For flow models, Flow-GRPO converts the deterministic ODE into a marginal-preserving SDE and then applies GRPO (Liu et al., 2025; Shao et al., 2024), which still requires a multi-step rollout and transition accounting. DiffusionNFT avoids reverse-chain likelihood estimation by contrasting positive and negative generations through a forward flow-matching objective (Zheng et al., 2026), while DIPOLE constructs reward-maximizing and reward-minimizing diffusion policies and combines their scores to control policy improvement (Liang et al., 2026). These approaches either explicitly model a Markov chain or SDE, or rely on structural properties of multi-step diffusion/flow distributions and paired positive–negative policies. Developing effective reinforcement-learning methods for one-step denoising models remains an open problem.

3 PRELIMINARIES

3.1 PROBLEM DEFINITION

Given camera observations I_{cam}, navigation commands L_{nav}, and ego state S_{ego} at the current time, an end-to-end planner must output a trajectory $\tau = \{(x_t, y_t, \theta_t)\}_{t=1}^{H}$ over H future time steps, where (x_t, y_t) and θ_t denote position and heading in the current ego coordinate frame, respectively. I_{cam} contains images from four cameras facing front, rear, left, and right; L_{nav} is the navigation-command encoding; and S_{ego} contains the ego vehicle's position, velocity, and acceleration.

We use a pretrained visual encoder to extract scene features from camera observations (Oquab et al., 2023), and encode the navigation command and ego state into a condition token:

$$F_{\text{scene}} = \text{Enc}_{\text{cam}}(I_{\text{cam}}), \qquad e_{\text{ego}} = \text{Linear}_{\text{ego}}([S_{\text{ego}}; L_{\text{nav}}]). \tag{1}$$

Here, F_{scene} consists of 64 scene registers. Let $c = (F_{\text{scene}}, e_{\text{ego}})$ denote the planning condition. The components of z are independently sampled from the standard normal distribution. The conditional generator directly maps initial noise z to the final trajectory represented by (x, y, θ) at each future time step:

$$z \sim p_z = \mathcal{N}(0, 1), \qquad \tau = G_\theta(c, z) = \{(x_t, y_t, \theta_t)\}_{t=1}^{H}. \tag{2}$$

3.2 DRIFT ONE-STEP GENERATIVE MODELS

Drift uses training-time sample interactions to move the generated distribution toward a reference distribution, incorporating distribution evolution into generator parameter updates so that inference produces a sample with a single forward pass (Deng et al., 2026). For notational simplicity, we fix the condition c in this subsection and denote the current generated distribution by Q and the reference distribution by P. Reference samples $y \sim P$ provide attraction, while generated samples $u' \sim Q$ provide repulsion. A basic Drift construction defines the Drift field vector through kernel-weighted mean displacements:

$$V_D(u; P, Q) = \frac{\mathbb{E}_{y\sim P}[k(u, y)(y - u)]}{Z_P(u)} - \frac{\mathbb{E}_{u'\sim Q}[k(u, u')(u' - u)]}{Z_Q(u)}. \tag{3}$$

Here, k is a positive-valued similarity kernel, and Z_P and Z_Q normalize the kernel weights of reference and generated samples, respectively. The first term draws generated samples toward the reference distribution, while the second provides repulsion between generated samples. When $P = Q$, the two terms cancel, giving $V_D(u; P, P) = 0$, which corresponds to field equilibrium at distribution matching. In practice, sample sets estimate this direction, potentially using multiple kernel scales and affinity normalization between samples. Thus, the field vector adjusts the distribution during training; directly outputs result in a 1-NFE paradigm during inference.

4 METHOD

In this section, we take the perspective of energy-induced field vectors and derive the Drift Field Vector from a Gaussian-kernel MMD energy, explaining its origin and rationale. Following the same approach, we derive a Reward Field Vector from a reward potential to enable reinforcement learning compatible with Drift. By constructing multiple positive samples and designing a critic network, we obtain the Drift Field Vector and Reward Field Vector directly in trajectory space to guide generated trajectory updates. Two-stage training enables the model to explore feasible trajectories conditioned on a scene and converge to high-scoring region.

4.1 GENERATOR OPTIMIZATION GUIDED BY DISTRIBUTION ENERGY FUNCTIONS

For a fixed scene condition c, let $Q_\theta(\cdot \mid c)$ denote the conditional trajectory distribution induced by the generator. We optimize a distributional energy $\mathcal{E}(Q)$ through its first variation: the induced field vector is $V_{\mathcal{E}}(x) = -\nabla_x \, \delta\mathcal{E}/\delta Q(x)$, which transports generated samples toward lower energy. The Drift field therefore guides the generated distribution toward the reference distribution.

Lemma 1. Let P and Q be probability distributions on the trajectory space, and define the distributional energy by the half-squared Gaussian-kernel MMD (Gretton et al., 2012):

$$\mathcal{E}_{\mathrm{MMD}}(Q) = \frac{1}{2}\mathbb{E}_{x,x'\sim Q}\left[k_\sigma(x,x')\right] + \frac{1}{2}\mathbb{E}_{y,y'\sim P}\left[k_\sigma(y,y')\right] - \mathbb{E}_{x\sim Q,y\sim P}\left[k_\sigma(x,y)\right], \tag{4}$$

where $k_\sigma(x,y) = \exp(-\|x-y\|_2^2/(2\sigma^2))$. The induced field is

$$V_{\mathrm{MMD}}(x) = \frac{1}{\sigma^2}\left\{\mathbb{E}_{y\sim P}\left[k_\sigma(x,y)(y-x)\right] - \mathbb{E}_{x'\sim Q}\left[k_\sigma(x,x')(x'-x)\right]\right\}. \tag{5}$$

This vector consists of an attraction term induced by positive samples and a repulsion term induced by generated samples. It therefore provides the unnormalized counterpart of the Drift Field Vector used in practice. For $x = G_\theta(c,z)$, the parameter gradient satisfies

$$\nabla_\theta \mathcal{E}_{\mathrm{MMD}}(Q_\theta) = -\mathbb{E}_z[J_\theta(z)^\mathsf{T} V_{\mathrm{MMD}}(x)]. \tag{6}$$

Following this approach, treating a distributional reward function as an energy function naturally yields a corresponding reward field vector. The proof is provided in Appendix A.1.

Lemma 2. For a fixed differentiable reward function R_ϕ, define the negative expected reward energy as

$$\mathcal{E}_R(Q) = -\mathbb{E}_{x\sim Q} R_\phi(x). \tag{7}$$

Its field vector is

$$V_R(x) = \nabla_x R_\phi(x). \tag{8}$$

The proof is provided in Appendix A.2. Thus, the reward function's gradient with respect to its input trajectory can serve as a reward-improvement direction for the trajectory distribution.

Lemma 3. For $x = g_\theta(z)$, an energy functional $\mathcal{E}(Q)$, and its corresponding field vector $V(x)$, construct the loss function $\mathcal{L}_{\mathrm{vec}}(\theta)$ as follows:

$$\mathcal{L}_{\mathrm{vec}}(\theta) = \frac{1}{2}\mathbb{E}_{x\sim g_\theta(z)}\left\|x - \mathrm{sg}[x + \eta V(x)]\right\|_2^2. \tag{9}$$

After stopping gradients through the entire target, the gradient of this loss with respect to the network parameters θ has the same direction as the parameter gradient of the energy function $\mathcal{E}(Q)$:

$$\nabla_\theta \mathcal{L}_{\mathrm{vec}} = -\eta \mathbb{E}_{x\sim g_\theta(z)}\left[J_\theta(z)^\mathsf{T} V(x)\right] = \eta \nabla_\theta \mathcal{E}(Q_\theta). \tag{10}$$

The proof is provided in Appendix A.3. Thus, in practical training, we can optimize the trajectory decoder by constructing field vectors and computing $\mathcal{L}_{\mathrm{vec}}(\theta)$.

4.2 Constructing Field Vectors

We cluster expert trajectories from the training set to obtain a vocabulary $\mathcal{B} = \{y_j\}_{j=1}^{4096}$ containing 4096 representative trajectories. For each training scene c, a planning evaluator computes the PDMS of vocabulary trajectories; we select 32 high-PDMS vocabulary trajectories in the expert trajectory's neighborhood and add the scene's ground-truth expert trajectory y_{GT}. These 33 trajectories receive equal weights to form an empirical scene-conditioned positive-sample distribution. The vocabulary supplies candidate reasonable driving behaviors in the expert neighborhood, scene-based scoring filters trajectories suitable for the current environment, and GT preserves human expert behavior. Given the current positive-sample set $Y = \{y_j\}_{j=1}^M$ and generated-sample set $X = \{x_\ell\}_{\ell=1}^N$, where $x_\ell = G_\theta(c, z_\ell)$, we follow the Drift formulation (Deng et al., 2026). Let k_h denote the unnormalized kernel at scale h, and let $Z_i^{+,h}$ and $Z_i^{-,h}$ denote the corresponding positive- and generated-sample normalization factors defined in Appendix A.4. The Drift Field Vector for generated trajectory i is

$$\widehat{V}_{D,i} = \sum_{h\in\mathcal{H}}\left[\frac{1}{Z_i^{+,h}}\frac{1}{M}\sum_{j=1}^M k_h(x_i,y_j)(y_j - x_i) - \frac{1}{Z_i^{-,h}}\frac{1}{N}\sum_{\ell=1}^N k_h(x_i,x_\ell)(x_\ell - x_i)\right]. \tag{11}$$

Here, i indexes the current generated sample, j and ℓ iterate over positive and generated samples, respectively, and $\mathcal{H}$ is the set of kernel scales. The first term attracts the current distribution toward the positive-sample distribution, while the second repels it from the generated-sample distribution.

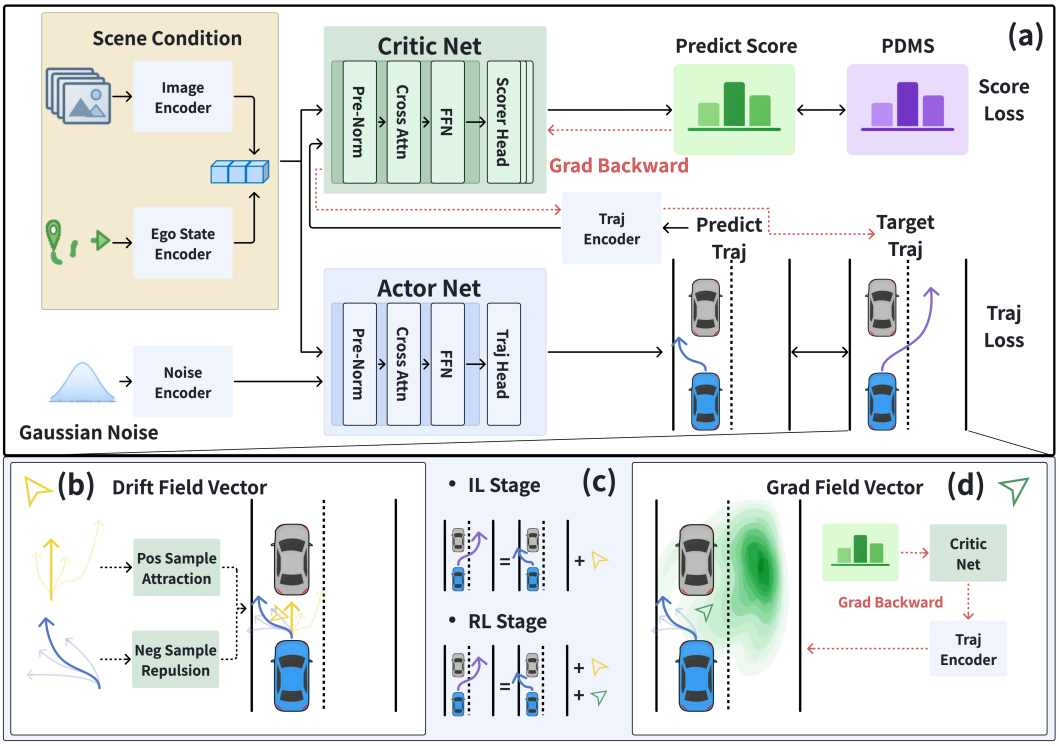


Figure 2: **Overview of FieldDrive.** (a) illustrates the overall FieldDrive architecture; (b) and (d) show the construction of the Drift Field Vector and Reward Field Vector, respectively; (c) highlights the difference in target trajectories between the two training stages.

For the one-step planning process, the immediate action reward r is identical to the final planning reward R. In the autonomous-driving setting, we directly use the PDM Score (PDMS) of a predicted trajectory as its reward. We obtain six actual metrics using the planning evaluator and use them to supervise the critic's predictions. The critic scores generated trajectories and aggregates the predicted metrics into a differentiable reward potential $R_\phi(c, x)$; the logarithmic scoring form and label processing are described in Appendix B.1. The critic is updated online with current generated samples and evaluation feedback, allowing the reward potential to adapt to changes in the generated distribution. With critic parameters and scene features fixed, we differentiate the critic with respect to its input trajectory to obtain the Reward Field Vector:

$$\widehat{V}_{R,i} = \nabla_x R_\phi(c, x)|_{x=x_i} . \tag{12}$$

4.3 Two-Stage Training Paradigm

In the first stage, imitation learning uses the positive-sample distribution formed by 32 high-scoring vocabulary trajectories in the expert neighborhood and GT as its target. For each scene, we sample N generated trajectory encodings $x_i = G_\theta(c, z_i)$, use generated trajectories as negative samples and the 33 trajectories as positive samples, and construct the Drift Field Vector $\widehat{V}_{D,i}$. Following Lemma 3, we train the generator through stop-gradient regression. During training, the metrics predicted by critic are supervised by actual PDMS metrics. For planning metric m, let $\tilde{r}_{im}$ denote the actual evaluation label and $p_{\phi,m}(c, x_i)$ the critic prediction. The first-stage critic loss, actor loss, and total loss are

$$\begin{aligned}
\mathcal{L}_{\text{critic}} &= \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{6} \text{BCE}(p_{\phi,m}(c, \text{sg}[x_i]), \tilde{r}_{im}) , \\
\mathcal{L}^{\text{IL}}_{\text{actor}} &= \frac{1}{N}\sum_{i=1}^{N} \left\| x_i - \text{sg}\left[x_i + \widehat{V}_{D,i}\right] \right\|_2^2 , \\
\mathcal{L}_{\text{IL}} &= \lambda^{\text{IL}}_{\text{traj}}\mathcal{L}^{\text{IL}}_{\text{actor}} + \lambda^{\text{IL}}_{\text{score}}\mathcal{L}_{\text{critic}} .
\end{aligned} \tag{13}$$

Multi-positive supervision teaches the generator a diverse distribution of feasible speed and maneuver choices. The resulting samples provide both exploration coverage and reward feedback for the subsequent reinforcement-learning stage; the effect of this coverage is evaluated in Table 3.

The second stage initializes the generator from pretraining and adds reward-potential gradients to positive-sample distribution matching. The Drift Field Vector $\widehat{V}_{D,i}$ constrains expert behavior, while the Reward Field Vector $\widehat{V}_{R,i}$ guides trajectories toward high-reward regions. Combining the positive-sample distribution-matching direction and the reward direction in a stop-gradient target yields the second-stage generator loss:

$$
\begin{aligned}
V_i &= \alpha \widehat{V}_{D,i} + \beta \widehat{V}_{R,i}, \\
\mathcal{L}_{\text{actor}}^{\text{RL}} &= \frac{1}{N} \sum_{i=1}^{N} \| x_i - \text{sg}[x_i + V_i] \|_2^2 , \\
\mathcal{L}_{\text{RL}} &= \lambda_{\text{traj}}^{\text{RL}} \mathcal{L}_{\text{actor}}^{\text{RL}} + \lambda_{\text{score}}^{\text{RL}} \mathcal{L}_{\text{critic}}.
\end{aligned}
\tag{14}
$$

The two stages first learn a multimodal feasible distribution and then optimize it with reward feedback. At inference, zero noise z_* produces the final (x, y, θ) trajectory in one pass, while vocabulary filtering and reward-gradient construction remain training-only. Further implementation details are provided in Appendix B.1.

5 Experiments

5.1 Experimental Setup

Implementation Details. We use pretrained DINOv2 ViT-S/14 as the visual backbone. Expert trajectories are clustered into a vocabulary of 4096 representatives; for each training scene, we select 32 high-scoring trajectories in the GT neighborhood and add the GT trajectory to form the positive-sample set. The imitation-learning (IL) stage runs for 40 epochs and learns this scene-conditioned trajectory distribution. The reinforcement-learning (RL) stage is initialized from the IL model and runs for 20 epoch. The planner predicts 8 trajectory points over the next 4 seconds and directly outputs one trajectory with a single decoding pass.

Dataset. We evaluate our model on NAVSIM, a planning-oriented benchmark built on OpenScene, a compact redistribution of nuPlan (Dauner et al., 2024). It provides multi-view camera and LiDAR observations, ego status, high-definition maps, and object annotations at 2 Hz, while emphasizing challenging changes in driving intention. We train on the official `navtrain` split and evaluate on `navtest`. NAVSIM v1 reports the PDM Score (PDMS) from a 4-second non-reactive evaluation, aggregating no-at-fault collisions, drivable-area compliance, time-to-collision, comfort, and ego progress. We also use the Extended PDM Score (EPDMS) introduced for NAVSIM v2 (Cao et al., 2025) on the same `navtest` split for evaluation; compared with PDMS, it additionally evaluates driving-direction compliance, traffic-light compliance, lane keeping, history comfort, and extended comfort. Higher scores indicate better planning performance.

5.2 Main Results

Table 1 reports NAVSIM v1 results. FieldDrive achieves 92.1 PDMS, exceeding the RL-based trajectory-generation methods DiffusionDriveV2 (Zou et al., 2025) and ReCogDrive (Xiong et al., 2026) by 0.9 and 1.3 points, respectively. Table 4 reports the planning speed: with one-NFE direct decoding and no trajectory reconstruction, FieldDrive reaches 546 FPS on a single H20 GPU, which is 1.26× the 434 FPS of MeanFuser (Wang et al., 2026). With an additional scorer, FieldDrive reaches 93.3 PDMS, showing that our method also supports scorer-based selection.

Table 1: **Performance on the NAVSIMv1 navtest benchmark.** The best value in each metric is bold. † denotes methods with candidate scoring during inference.

Method	Backbone	Planning metrics					PDMS ↑
		NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	
TransFuser (Chitta et al., 2022)	ResNet-34	97.7	92.8	92.8	**100**	79.2	84.0
GoalFlow (Xing et al., 2025)	ResNet-34	98.3	93.8	94.3	**100**	79.8	85.7
Hydra-MDP† (Li et al., 2024)	ResNet-34	98.3	96.0	94.6	**100**	78.7	86.5
DiffusionDrive† (Liao et al., 2025)	ResNet-34	98.2	96.2	94.7	**100**	82.2	88.1
MeanFuser (Wang et al., 2026)	ResNet-34	98.6	97.0	95.0	**100**	82.8	89.0
DriveFine† (Dang et al., 2026)	SigLIP-384	98.6	97.9	95.2	99.9	85.5	90.7
ReCogDrive (Xiong et al., 2026)	InternVL3-8B	97.9	97.3	94.9	**100**	87.3	90.8
DiffusionDriveV2† (Zou et al., 2025)	ResNet-34	98.3	97.9	94.8	99.9	87.5	91.2
iPad† (Guo et al., 2026)	ResNet-34	98.6	98.3	94.9	**100**	88.0	91.7
DrivoR† (Kirby et al., 2026)	DINOv2 ViT-S	**98.9**	98.3	**96.2**	**100**	89.1	93.1
FieldDrive (direct)	DINOv2 ViT-S	97.8	98.6	93.4	**100**	90.3	92.1
FieldDrive (with scorer)†	DINOv2 ViT-S	98.1	**98.7**	94.5	**100**	**91.8**	**93.3**

Table 2 reports results using the NAVSIM v2 metrics on the same `navtest` split. FieldDrive achieves 89.7 EPDMS with direct trajectory decoding.

Table 2: **Performance of the NAVSIMv2 navtest benchmark.**

Method	Extended planning metrics									EPDMS ↑
	NC ↑	DAC ↑	DDC ↑	TL ↑	EP ↑	TTC ↑	LK ↑	HC ↑	EC ↑	
Ego Status MLP (Dauner et al., 2024)	93.1	77.9	92.7	99.6	86.0	91.5	89.4	**98.3**	85.4	64.0
TransFuser (Chitta et al., 2022)	96.9	89.9	97.8	99.7	87.1	95.4	92.7	**98.3**	87.2	76.7
Hydra-MDP++ (Li et al., 2024)	97.2	97.5	99.4	99.6	83.1	96.5	94.4	98.2	70.9	81.4
DriveSuprim (Yao et al., 2026)	97.5	96.5	99.4	99.6	88.4	96.6	95.5	98.3	77.0	83.1
DiffusionDriveV2 (Zou et al., 2025)	97.7	96.6	99.2	**99.8**	**88.9**	97.2	96.0	97.8	**91.0**	85.5
MeanFuser (Wang et al., 2026)	98.3	97.2	**99.6**	**99.8**	87.6	97.4	**97.3**	**98.3**	88.2	89.5
FieldDrive (direct)	**98.5**	**99.0**	**99.6**	99.6	88.7	**98.0**	94.3	**98.3**	73.7	**89.7**

5.3 Ablation Studies

Comparion of Positive Trajectory Number. To show that first-stage trajectory diversity is important for second-stage reinforcement learning performance, we use different numbers of vocabulary trajectories as positive samples for first-stage imitation learning, inducing different levels of diversity in generated trajectories. Detailed results are reported in Table 3. As the number of positive samples increases from 1 to 33, generated trajectories pairwise ade increases from 0.65 to 2.25, and models with greater diversity also achieve substantially better final trajectory performance after second-stage reinforcement learning post-training. These results show that first-stage diversity is necessary for reinforcement learning post-training. The table also shows the substantial performance improvement brought by reinforcement learning.

Table 3: **Effect of positive-sample count.** IL diversity reports pairwise ADE/FDE and maximum ADE/FDE to the feasible trajectory set.

Positive set	Count	IL diversity				IL PDMS ↑	RL PDMS ↑
		Pair ADE	Pair FDE	Max ADE	Max FDE		
GT only	1	0.65	1.48	2.16	5.12	**84.65**	90.64
4 voc. traj. + GT	5	0.92	1.86	2.75	5.86	83.97	91.63
16 voc. traj. + GT	17	1.52	2.77	4.56	8.41	81.21	91.70
32 voc. traj. + GT	33	**2.25**	**3.87**	**6.81**	**11.69**	79.34	**92.14**

Table 4: **Decoding speed.**

Method	Plan FPS ↑	PDMS ↑
TransFuser (Chitta et al., 2022)	3934	84.0
Hydra-MDP (Li et al., 2024)	25	86.5
DiffusionDrive (Liao et al., 2025)	75	88.1
GoalFlow (Xing et al., 2025)	11	85.7
MeanFuser (Wang et al., 2026)	434	89.0
FieldDrive (direct)	**546**	**92.1**

Field Vector components ablation. The RL stage combines the Drift Field Vector and Reward Field Vector. Table 5 shows that either Field Vector alone is insufficient: Drift alone mainly preserves the expert region, whereas reward-only updates can leave that region. Their combination achieves 92.14 PDMS. Moreover, experiments in Appendix B.4 verify that the gradient provided by the critic network accurately serves as the Reward Field Vector: updating trajectories along this direction consistently improves their mean rewards.

Scorer-compatible proposal quality. With additional scorer, FieldDrive achieve 93.34 PDMS. We compare FieldDrive and DrivoR after applying the scorer with the same proposal-count protocol. FieldDrive is consistently better from 4 to 64 proposals. The advantage is more pronounced when fewer proposals are used. The results indicates that our model ensures the overall quality of the generated samples.

Table 5: **Field-vector ablation.**

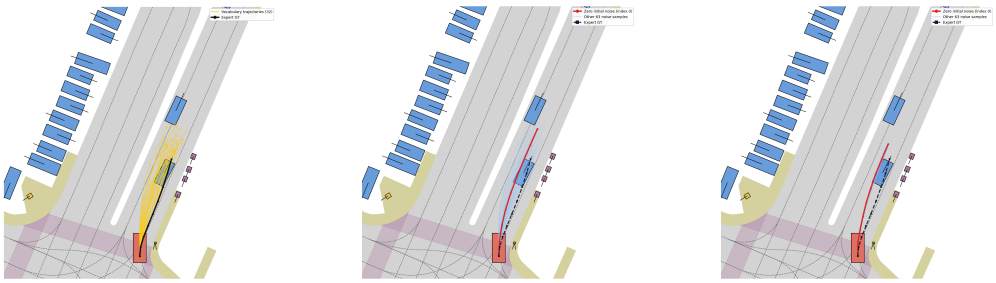
Setting	Drift	Reward	PDMS ↑
Drift only	Yes	No	85.58
Reward only	No	Yes	84.05
Drift + Reward	Yes	Yes	**92.14**

Table 6: **Scorer compatibility.**

Method	4	8	16	32	64
DrivoR†	89.99	91.55	92.32	92.89	93.12
FieldDrive†	**90.98**$^{+0.99}$	**92.15**$^{+0.60}$	**92.75**$^{+0.43}$	**93.00**$^{+0.11}$	**93.34**$^{+0.22}$

5.4 Qualitative Results

Figure 3 compares the vocabulary positives with trajectories generated after IL and RL. IL covers multiple local speed and maneuver choices, while RL concentrates the generated distribution toward the high-scoring region.



(a) Positive samples. (b) FieldDrive After IL. (c) FieldDrive After RL.

Figure 3: Trajectory distributions of positive samples and the model output.

6 Conclusion

In this work, we propose FieldDrive, an efficient framework for trajectory decoding and optimization. We analyze the origin of the Drift Field Vector and construct a Reward Field Vector accordingly to guide reinforcement learning for a generative model. This approach uses the strong fitting capacity of generative models to balance reinforcement learning's exploration and optimization requirements, replacing multi-step trajectory generation with one-step inference. Experiments on NAVSIM v1 and v2 demonstrate that FieldDrive achieves efficient direct trajectory planning. We hope that our work will further advance the use of generative models in autonomous driving.

References

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 4965–4987, 2024.

Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, et al. Pseudo-simulation for autonomous driving. *arXiv preprint arXiv:2506.04218*, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE transactions on pattern analysis and machine intelligence*, 45(11):12878–12895, 2022.

Chenxu Dang, Sining Ang, Yongkang Li, Haochen Tian, Jie Wang, Guang Li, Hangjun Ye, Jie Ma, Long Chen, and Yan Wang. Drivefine: Refining-augmented masked diffusion vla for efficient and robust driving. In *European Conference on Computer Vision*, pp. 412–429. Springer, 2026.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, et al. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *Advances in Neural Information Processing Systems*, 37:28706–28719, 2024.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13:723–773, 2012.

Ke Guo, Haochen Liu, Xiaojun Wu, Jia Pan, and Chen Lv. ipad: Iterative proposal-centric end-to-end autonomous driving. *IEEE Robotics and Automation Letters*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17853–17862. IEEE, 2023.

Zhiyu Huang, Xinshuo Weng, Maximilian Igl, Yuxiao Chen, Yulong Cao, Boris Ivanovic, Marco Pavone, and Chen Lv. Gen-drive: Enhancing diffusion generative driving policies with reward modeling and reinforcement learning fine-tuning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3445–3451. IEEE, 2025.

Ellington Kirby, Alexandre Boulch, Yihong Xu, Yuan Yin, Gilles Puy, Éloi Zablocki, Andrei Bursuc, Spyros Gidaris, Renaud Marlet, Florent Bartoccioni, et al. Driving on registers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 32058–32069, 2026.

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

Ruiming Liang, Yinan Zheng, Kexin Zheng, Tianyi Tan, Jianxiong Li, Liyuan Mao, Zhihao Wang, Guang Chen, Hangjun Ye, Jingjing Liu, et al. Dichotomous diffusion policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 61669–61690, 2026.

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

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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, 2025. *URL https://arxiv. org/abs/2505.05470*, 2:5, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Ananya Trivedi, Anjian Li, Mohamed Elnoor, Yusuf Umut Ciftci, Avinash Singh, Jovin D'sa, Sangjae Bae, David Isele, Taşkın Padır, and Faizan M Tariq. Adaptive time step flow matching for autonomous driving motion planning. In *2026 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1503–1510. IEEE, 2026.

Junli Wang, Yinan Zheng, Xueyi Liu, Zebin Xing, Pengfei Li, Kun Ma, Hangjun Ye, Guang Chen, Guang Li, Long Chen, et al. Meanfuser: Fast one-step multi-modal trajectory generation and adaptive reconstruction via meanflow for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17884–17893, 2026.

Zebin Xing, Xingyu Zhang, Yang Hu, Bo Jiang, Tong He, Qian Zhang, Xiaoxiao Long, and Wei Yin. Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1602–1611. IEEE, 2025.

Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, Kun Ma, et al. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. In *International Conference on Learning Representations*, volume 2026, pp. 157518–157556, 2026.

Wenhao Yao, Zhenxin Li, Shiyi Lan, Zi Wang, Xinglong Sun, Jose M Alvarez, and Zuxuan Wu. Drivesuprim: Towards precise trajectory selection for end-to-end planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 11910–11918, 2026.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. In *International Conference on Learning Representations*, volume 2026, pp. 134129–134150, 2026.

Yinan Zheng, Ruiming Liang, Kexin Zheng, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Li, Xianyuan Zhan, et al. Diffusion-based planning for autonomous driving with flexible guidance. In *International conference on learning representations*, volume 2025, pp. 37207–37227, 2025.

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