PlannerRFT: Reinforcing Diffusion Planners through Closed-Loop and Sample-Efficient Fine-Tuning

Supplementary Material

A. Discussions

Towards a better understanding of this work, we supplement intuitive questions that may raise.

Q1. *What makes PlannerRFT effective?* PlannerRFT's effectiveness stems from three key factors:

Enhance Lateral movement. The expert trajectory distribution is dominated by straight-driving maneuvers, causing IL planners to underfit lateral skills such as lane changes and obstacle avoidance. PlannerRFT introduces structured lateral perturbations through guided denoising and reinforces them with reward-oriented optimization, enabling lane changes and obstacle-avoidance behaviors in complex scenes, as shown in Fig. A6 and Fig. 4.

Plan-Motion Alignment. IL planners mimic expert trajectories without considering the downstream controller, leading to execution failures in narrow or high-precision maneuvers. PlannerRFT evaluates executed closed-loop trajectories via the simulator's vehicle dynamics and updates the planner with execution-level rewards such as collision and off-road penalties. This closed-loop correction bridges the gap between planning and execution and markedly improves maneuver precision and controller feasibility in challenging environments, as shown in Fig. A7.

Trial-and-error Rollouts. Real traffic is dynamic, and blindly reproducing expert behavior can be unsafe. PlannerRFT leverages trial-and-error closed-loop RL to adapt to dynamic traffic, markedly improving interaction capability, as shown in Fig. A8 and Fig. A9, which also explains its strong gains on reactive-traffic benchmarks. We further observe that RL helps mitigate the causal-confusion issues inherent in IL, as shown in Fig. A10.

Q2. *Why adopt a dual-branch optimization (PPO and GRPO), and how is training stability maintained?* The exploration policy adjusts guidance scales at every simulation step and directly affects long-horizon closed-loop behavior, making PPO suitable for online learning. In contrast, the DiT generator outputs high-dimensional multi-step trajectories that are better optimized offline with group-based updates, where GRPO provides efficient training. Training stability is further ensured by applying policy-guided denoising around a fixed reference trajectory rather than the evolving DiT outputs, and the well-trained reference trajectory distribution prevents collapse and yields steadily improving rewards, as shown in Fig. 6.

Q3. *What are potential applications and future directions with the PlannerRFT framework and the nuMax simulator?*

Algorithm 1 Guided Denoising for RL Sampling

Require: Current observation o_t, scene encoder E_{scene}, route encoder E_{navi}, reference DiT D_{ref}, fine-tuned DiT D_θ, exploration policy π_ϕ, GRPO group size G_{grpo}

// Step 1: Scenario encoding.
$F_{\text{scene}} \leftarrow E_{\text{scene}}(o_t)$
$F_{\text{navi}} \leftarrow E_{\text{navi}}(o_t)$
// Step 2: Get reference trajectory.
$x_S^{\text{ref}} \leftarrow z,\ z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $i = 1$ to S **do** ▷ $S = 5$ DDIM steps
  $s \leftarrow s_i$ ▷ VP-SDE timestep
  $x_{s-1}^{\text{ref}} \leftarrow D_{\text{ref}}(x_s^{\text{ref}}, s, F_{\text{scene}}, F_{\text{navi}})$
end for
$x^{\text{ref}} \leftarrow x_0^{\text{ref}}$
// Step 3: Get adaptive exploration direction.
$(a_{\text{lat.}}, b_{\text{lat.}}, a_{\text{lon.}}, b_{\text{lon}}) \leftarrow \pi_\phi\left(x^{\text{ref}}, F_{\text{scene}}, F_{\text{navi}}\right)$
// Step 4: Sample multi-modal guidance scales.
for $k = 1$ to G_{grpo} **do**
  $\eta_{\text{lat}}^{(k)} \sim \text{Beta}(a_{\text{lat}}, b_{\text{lat}}), \quad \eta_{\text{lon}}^{(k)} \sim \text{Beta}(a_{\text{lon}}, b_{\text{lon}})$
end for
// Step 5: Guided denoising.
for $k = 1$ to G_{grpo} **do**
  $x_S^{(k)} \leftarrow z,\ z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  for $i = 1$ to S **do**
    $s \leftarrow s_i$
    $x_{s-1}^{(k)} \leftarrow D_\theta(x_s^{(k)}, s, F_{\text{scene}}, F_{\text{navi}}, \eta_{\text{lat}}^{(k)}, \eta_{\text{lon}}^{(k)}, x^{\text{ref}})$ ▷ classifier-guided denoising following Eq. (5)
  end for
  $x^{(k)} \leftarrow x_0^{(k)}$
end for
return $\{x^{(k)}\}_{k=1}^{G_{\text{grpo}}}$ ▷ Multi-modal trajectory samples

Model: Build upon a simple diffusion planner, PlannerRFT can enhance the multi-modality and adaptive sampling in RL. We freeze the encoder and fine-tune only the trajectory DiT, which suggests a potential ability to act as a unified decoder for different input modalities, such as sensor-based E2E planners or language-conditioned VLM/VLA planners.

Simulator: To support our training pipeline, we develop nuMax, a fast online RL simulator designed to facilitate academic research on the nuPlan benchmark. Additional implementation details, limitations, and future development plans are provided in Sec. C.

B. Implementation Details of PlannerRFT

Training Details. Fig. A2 show the training pipeline for the PlannerRFT framework. For the commonly used online RL, the training pipeline involves two steps: (1) RL sampling and (2) policy update.

For RL sampling, PlannerRFT adopt the policy-guide denoising to generate multi-modal and scenario-adaptive trajectocy group. Algorithm 1 outlines the details of the policy-guide denoising process. We adopt a 5 steps DDIM sampling during training and inference for computational efficiency and exploration stochasticity.

$$x_{s-1} = \sqrt{\alpha_{s-1}}\,\hat{x}_s^0 + \sqrt{1-\alpha_{s-1}-\sigma_s^2}\,\epsilon_\theta(x_s, s) + \sigma_s z \tag{A1}$$

$$\epsilon_\theta(x_s, s) = \frac{x_s - \sqrt{\alpha_s}\,\hat{x}_s^0}{\sqrt{1-\alpha_s}} \tag{A2}$$

$$\sigma_s = \eta \cdot \sqrt{\frac{1-\alpha_{s-1}}{1-\alpha_s}}, \tag{A3}$$

where s is the denoising timestep, $\hat{x}_s^0$ is the model-predicted clean trajectory at timestep s, σ_s controls the stochasticity of DDIM sampling, and $z \sim \mathcal{N}(0, I)$ denotes standard Gaussian noise. Specifically, we set $\eta = 1$ during RL training to encourage stochastic exploration, and $\eta = 0$ during evaluation for deterministic sampling.

For policy update, PlannerRFT consists of two learnable modules: the exploration policy and the fine-tuned DiT, which have different optimization goals and are trained with different optimization losses.

We use the PPO loss to update the exploration policy, aiming to maximize the long-term cumulative reward in closed-loop planning.

$$\mathcal{L}_{\text{PPO}}(\phi) = \mathbb{E}_t\Big[\mathcal{L}_{\text{clip}}(\phi) - c_v\,(V_\phi(s_t) - V_t^{\text{target}})^2 + c_e\,\mathcal{H}(\pi_\phi(\cdot|o_t))\Big] \tag{A4}$$

$$\mathcal{L}_{\text{clip}}(\phi) = \min\Big(r_t(\phi)A_t,\ \text{clip}\big(r_t(\phi), 1-\epsilon, 1+\epsilon\big)A_t\Big) \tag{A5}$$

$$r_t(\phi) = \frac{\pi_\phi(\eta_t \mid o_t)}{\pi_{\phi\text{old}}(\eta_t \mid o_t)}, \tag{A6}$$

where $\mathcal{L}_{\text{clip}}$ is the clipped policy objective, $r_t(\phi)$ is the importance sampling ratio, A_t is the advantage estimate, $V_\phi(s_t)$ is the value prediction, $\mathcal{H}(\pi_\phi)$ denotes the entropy bonus, and c_v, c_e are the value and entropy coefficients. These hyperparameters are summarized in Tab. A1.

We use the GRPO loss to update the fine-tuned DiT, aiming to maximize the reward within the prediction horizon at the current timestep. Following DPPO [36], each conditional step in the diffusion chain is a Gaussian policy:

$$\pi_\theta(x_{s-1} \mid x_s) = \mathcal{N}\Big(x_{s-1};\ \mu_\theta(x_s, s),\ \sigma_s^2 I\Big) \tag{A7}$$

Table A1. **Hyperparameters for PlannerRFT.**

	Hyperparameter	Value
Guidance	Max. Lateral Offset $\lambda_{\text{lat.}}$	2.5 (m)
	Max. Longitudinal Offset $\lambda_{\text{lon.}}$	25 (%)
PPO	Samples	40M
	Initial Learning Rate	2.5×10^{-4}
	Learning Rate Schedule	Cosine decay
	Number of Envs.	128
	Env. Steps per Iteration	32
	Batch Size	4096
	Mini-batch Size	4096
	Steps per Epoch	1
	Epochs	4
	Value Coefficient c_v	0.5
	Entropy Coefficient c_e	0.01
	Discount Factor	0.99
	GAE λ	0.95
	Clip Range ϵ	0.2
	Max Gradient Norm	0.5
GRPO	Initial Learning rate	2.5×10^{-4}
	Learning Rate Schedule	Cosine decay
	Group Size G_{grpo}	8
	Mini-batch Size	4096
	Steps per Epoch	6
	Epochs	1
	Denoising Discount Factor γ	0.8
	BC loss weight c_b	0.4

Table A2. **Comparison of different inference types.** "Diffusion Planner$_{\text{DPM}}$" is the official 10-step DPM-solver version of Diffusion Planner [49]. "w/ guid." denotes inference with guided denoising, where the guidance scale is set to the mean of the Beta distribution. "w/o guid." denotes inference without guidance.

Model	Steps	Latency (ms)	Val14-NR	Val14-R
Diffusion Planner$_{\text{DPM}}$	10	86.43	89.87	82.80
PlannerRFT w/ guid.	10	75.48	89.83	83.93
PlannerRFT w/o guid.	5	**34.27**	**89.96**	**84.46**

$$\mu_\theta(x_s, s) = \sqrt{\alpha_{s-1}}\,\hat{x}_s^0 + \sqrt{1-\alpha_{s-1}-\sigma_s^2}\,\epsilon_\theta(x_s, s), \tag{A8}$$

where $\mu_\theta(x_s, s)$ is the deterministic update term in DDIM, and $\sigma_s^2 I$ controls the sampling stochasticity. Therefore, the optimization objective is to adjust the denoising process such that the conditional policy $\pi_\theta(x_{s-1} \mid x_s)$ is shifted toward trajectories with higher expected rewards:

$$\mathcal{L}_G = -\frac{1}{G_{grpo}}\sum_{k=1}^{G_{grpo}}\frac{1}{S}\sum_{s=1}^{S}\gamma^{s-1}\log\pi_\theta\Big(x_{s-1}^{(k)} \mid x_s^{(k)}\Big)\,\hat{A}_k - c_b\,\frac{1}{G_{grpo}}\sum_{k=1}^{G_{grpo}}\frac{1}{S}\sum_{t=1}^{S}\log\pi_\theta\Big(\tilde{x}_{s-1}^{(k)} \mid \tilde{x}_s^{(k)}\Big), \tag{A9}$$

where γ is the denoising discount factor. Following ReCog-

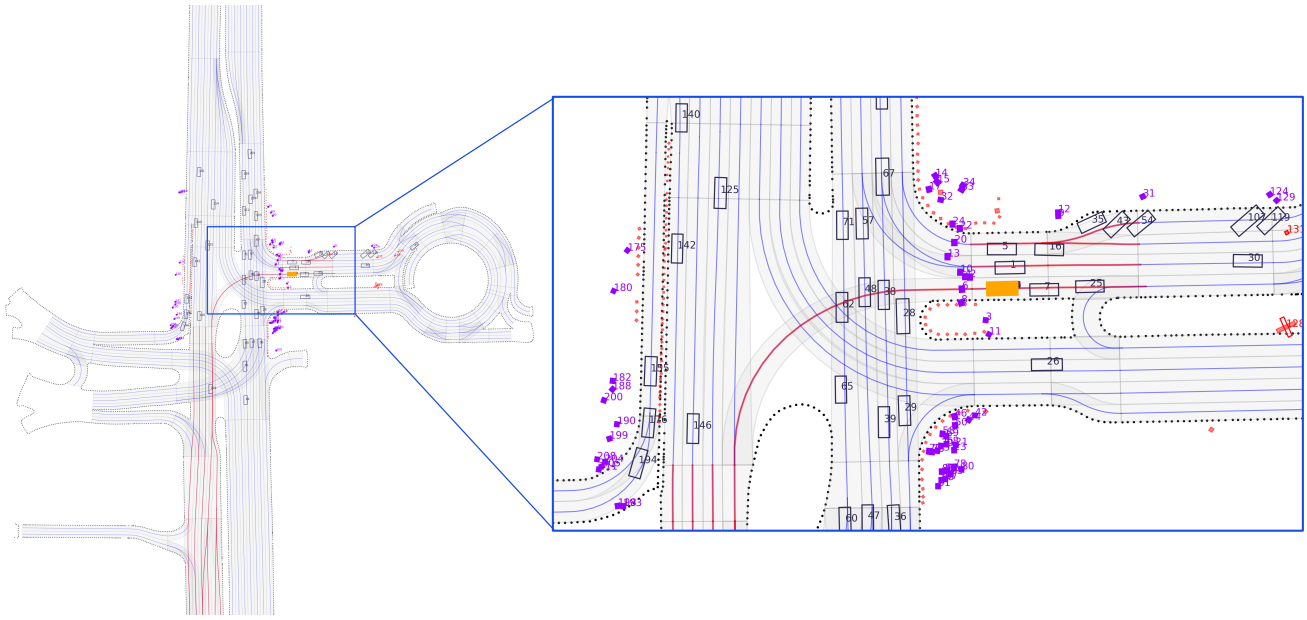

Figure A1. **Visualization of a Cached Scenario.** We cache scene elements within a 200 m radius of the ego vehicle, including lanes, dynamic agents, static obstacles, and navigation routes. Lane polygons are drawn in gray with blue centerlines, while navigation routes are highlighted in red. Surrounding vehicles are drawn as black rectangles, pedestrians and cyclists in purple, and static objects in red.

Drive [23], we incorporate a behavior cloning loss to prevent policy collapse during exploration, and c_b denotes the weight of the behavior cloning term.

Inference Details. For inference, we adopt the same 5-step DDIM sampler as in the training phase, without guided denoising or reliance on the reference planner. Table A2 compares different inference settings in terms of denoising steps, latency, and closed-loop performance on the Val14 benchmark. PlannerRFT with guided denoising requires twice as many denoising steps because each step depends on the reference planner's trajectory. This additional dependency also results in nearly double the inference latency compared to the unguided version. Performance-wise, the guided version is slightly worse than PlannerRFT without guidance. Exploration guidance provides a directional prior that enables sampling around the intended exploration direction, yielding both positive and negative trajectory examples that help refine the learned distribution. Under a limited training budget, however, the model may not fully capture an accurate distribution mean, particularly in fine-grained maneuvering scenarios, leading to slight performance degradation. These effects collectively highlight the sampling efficiency of PlannerRFT.

C. Implementation Details of nuMax

We develop nuMax, a JAX-based, GPU-parallel simulator built upon Waymax [10], to support large-scale closed-loop training on nuPlan. nuMax achieves significantly higher simulation speed compared to the official nuPlan simulator by re-designing the data representation, scene update pipeline, and agent dynamics entirely around JAX's functional. Below, we summarize the implementation details.

Scenario Cache. Efficient high-throughput data loading is crucial for large-scale training in closed-loop simulation. In the official nuPlan dataset, scenario recordings are stored in an SQLite[3] database and HD maps reside in a GeoPandas[4] dataframe, requiring the simulator to query both sources at every simulation step to retrieve lane geometry, dynamic agents, and scene context. This stepwise database access limits the simulation throughput. Consequently, nuMax pre-caches the training scenario based on ScenarioMax[5], a high-performance toolkit for autonomous vehicle scenario-based testing and dataset conversion. Specifically, for temporal context, we extract a fixed window consisting of 20 past frames, the current frame, and 150 future frames at a sampling interval of 0.1 s. For spatial context, we crop all scene elements within a 200 m radius centered on the ego vehicle, including lanes, dynamic agents, static obstacles, and navigation routes. An example visualization of the scenario cache is shown in Fig. A1. All processed data are serialized into TFRecord[6] files, enabling fast sequential I/O, efficient GPU loading, and full compatibility with JAX's parallelized execution model, thereby eliminating runtime

[3] https://sqlite.org
[4] https://geopandas.org
[5] https://github.com/valeoai/ScenarioMax
[6] https://www.tensorflow.org/tutorials/load_data/tfrecord

database queries and supporting high-throughput simulation in nuMax.

Tracker and Scorer. Reliable vehicle motion tracking is essential for robust closed-loop simulation. Waymax adopts a vehicle controller built on Perfect Control, but we found it occasionally understeers in sharp-turn scenarios. In nuMax, we replace the perfect-control controller with the two-stage motion controller from the official nuPlan-devkit[7] , which consists of an LQR tracker and a kinematic bicycle model. Our implementation follows the controller used in PDM-Closed[8] , which extends the controller to support batched trajectory inputs, enabling nuMax to track an entire group of candidate trajectories simultaneously during rollouts. We reimplement the two-stage controller in JAX and integrate it into nuMax's GPU-parallel simulation pipeline, where XLA[9] compilation further accelerates tracking and improves overall computational efficiency.

Comprehensive and principled reward evaluation is essential for effective reinforcement learning policy optimization. Building upon the metrics provided by Waymax, we further incorporate the official nuPlan scoring framework. In particular, our scorer includes both terminal penalties and soft penalties, enabling a more complete assessment of driving quality and safety during closed-loop rollouts.

For terminal penalties, once the ego violates any terminal condition, the simulation is immediately terminated and the reward is set to zero. The terminal penalties include:

- **Collision (`Col`)**: if the ego vehicle collides with surrounding vehicles, pedestrians, cyclists, or static objects.
- **Off road (`DAC`)**: if the ego vehicle drove off the drivable area.

For soft penalties, the ego aims to minimize these penalties while avoiding any terminal violations. The soft penalties include:

- **Wrong direction (`WD`)**: if the ego vehicle drives against the designated lane direction.
- **Time to collision (`TTC`)**: if the ego vehicle violates the time-to-collision (TTC) safety threshold.
- **Comfort (`C`)**: if the ego vehicle exhibits excessive longitudinal/lateral acceleration, jerk, or steering rate.
- **Ego Progress (`EP`)**: measured as the normalized ratio between the ego's accumulated route progress and that of the expert.
- **Speeding (`Speed`)**: if the ego vehicle exceeds the speed limit of the current lane or route segment.

Each component score lies within $[0, 1]$, and the final reward is obtained by weighted aggregation of all metrics:

$$r_t = (\texttt{Col} \cdot \texttt{DAC} \cdot \texttt{WD}) \times \left(\frac{w_1 \texttt{TTC} + w_2 \texttt{EP} + w_3 \texttt{C} + w_4 \texttt{Speed}}{\sum_i w_i} \right) \in [0, 1], \tag{A10}$$

where we adopt the official nuPlan weights: $w_1 = w_2 = 5.0$, $w_3 = 2.0$, and $w_4 = 4.0$. For the open-loop scorer used for GRPO trajectory evaluation, we adopt the same metric terms, but replace the terminal-penalty computation with the survival formulation defined in Eq. (6).

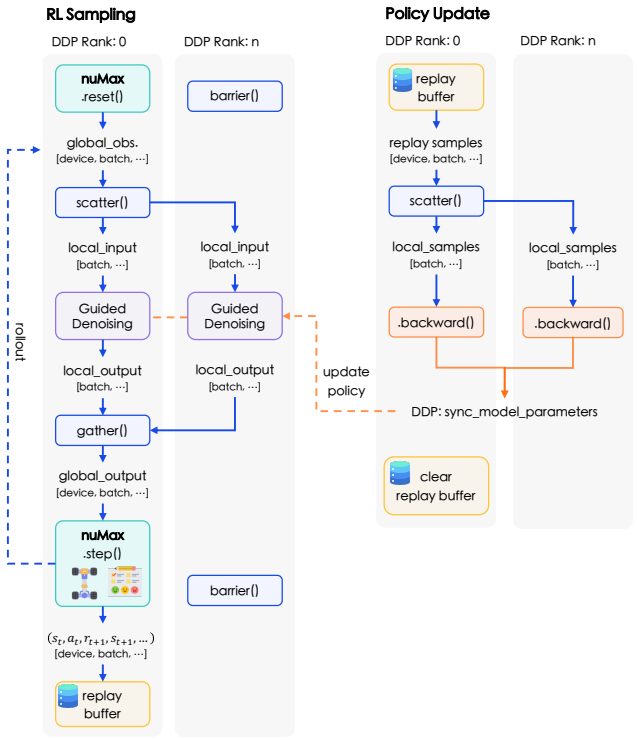

Figure A2. **Distributed RL Training Pipeline.** Policy inference and learning run in PyTorch DDP, while environment simulation is executed in JAX on rank-0 to avoid XLA conflicts. Observations and replay samples are scattered to all ranks for guided denoising and gradient updates, and planned trajectories are gathered back to rank-0 for simulation, with synchronization at every step.

Distributed RL Training Pipeline. For the reinforcement learning pipeline, we refer to V-Max [1], a JAX-based high-performance framework built on the Brax[10] engine, which integrates simulation pipelines, observation wrappers, and evaluation metrics. However, most diffusion-based planners widely adopted in the community are implemented in PyTorch, and re-implementing the entire model stack in JAX would be both costly and incompatible with existing pretrained models. To preserve compatibility and facilitate

[7] https://github.com/motional/nuplan-devkit
[8] https://github.com/autonomousvision/tuplan_garage
[9] https://openxla.org/xla
[10] https://github.com/google/brax

Table A3. **Closed-loop Planning Results on nuPlan Val14, Test14-hard, and Test14 benchmarks.** "Diffusion Planner$_{\texttt{DPM}}$" employs the official 10-step DPM-solver. "Diffusion Planner$_{\texttt{DDIM}}$" employs a 5-step DDIM sampler, identical to that used in PlannerRFT.

Type	Planner	Val14		Test14-hard		Test14-random	
		NR	R	NR	R	NR	R
Expert	Log-replay	93.53	80.32	85.96	68.80	94.03	75.86
Rule	IDM	75.60	77.33	56.15	62.26	70.39	74.42
	PDM-Closed	92.84	92.12	65.08	75.19	90.05	91.63
Learning	PDM-Open	53.53	54.24	33.51	35.83	52.81	57.23
	GameFormer	13.32	8.69	7.08	6.69	11.36	9.31
	PlanTF	84.27	76.95	69.70	61.61	85.62	79.58
	PLUTO	88.89	78.11	70.03	59.74	89.90	78.62
	Diffusion Planner$_{\texttt{DPM}}$	89.87	82.80	75.99	69.22	89.19	82.93
	Diffusion Planner$_{\texttt{DDIM}}$	89.81	82.94	76.01	68.18	89.14	82.63
	Flow Planner	**90.43**	83.31	76.47	70.42	89.88	82.93
	PlannerRFT(Ours)	89.96(+0.15)	**84.46**(+1.52)	**77.16**(+1.15)	**72.21**(+4.03)	**90.76**(+1.62)	**85.80**(+3.17)

Table A4. **Ablation on Guidance Type.** Results are reported on the Test14-random reactive benchmark.

	Guidance Choices		Closed-loop metrics ↑						
Training Type	lateral	longitudinal	Collisions	TTC	Drivable	Comfort	Progress	Speed	**R-score**
IL Pretrain$_{\texttt{DDIM}}$	✗	✗	86.58	79.05	94.48	86.03	76.99	97.20	68.18
PlannerRFT	✗	✓	87.50	81.62	92.65	86.76	**77.54**	97.99	69.59
	✓	✗	87.31	80.88	94.85	**87.50**	76.38	97.32	70.18
	✓	✓	**88.97**	**84.93**	**95.59**	85.66	77.17	**98.03**	**72.21**

broader community adoption, we therefore retain the policy model in PyTorch. Based on these considerations, we design a hybrid distributed RL training pipeline that couples JAX-based simulation with PyTorch Distributed Data Parallel (DDP) for policy inference and learning.

Fig. A2 illustrates our hybrid distributed RL training pipeline. All JAX-based simulation is executed on rank-0 to avoid XLA device conflicts and duplicate backend initialization that would arise from launching independent JAX runtimes on each PyTorch DDP rank. We distribute the observations and replay samples to all DDP ranks, and aggregate the planned trajectories back to rank 0 for simulation. In addition, we ensure that all ranks are synchronized before each simulation step.

Limitations of nuMax. nuMax currently inherits two key limitations. First, due to XLA's static-shape constraints, supporting other model input representations would require additional post-processing or re-caching, highlighting the need for a general scenario cache interface. Second, surrounding-vehicle simulation is log-replay rather than IDM, as the latter slows down training; improving the efficiency of the IDM traffic simulation remains an important direction for future optimization.

D. Additional Ablation Studies

Additional planning results in Test14-Random. Table A3 reports the closed-loop performance on the Test14-random benchmark, which contains 261 randomly selected scenarios from the nuPlan Planning Challenge. As shown in Tab. A3, the 5-step DDIM sampler and the ODE-based DPM-solver yield nearly identical results, ensuring a fair comparison. PlannerRFT achieves the best performance in both non-reactive (NR) and reactive (R) settings in Test14-random, improving over the pretrained planner by +1.62 (NR) and +3.17 (R).

Ablation on Guidance Choices. We evaluate the effectiveness of lateral and longitudinal guidance by enabling them individually in the policy-guided denoising process. As shown in Tab. A4, lateral guidance improves performance in Drivable and Comfort, as it enhances sharp-turn performance and produces smoother lateral maneuvers. In contrast, longitudinal guidance performs better in terms of Collisions, TTC, Progress, and Speed, as these behaviors can be controlled through acceleration and deceleration. Combining both forms of guidance results in the best performance, highlighting the complementary effect of lateral and longitudinal exploration in optimizing closed-loop planning.

Ablation on Group Size. We evaluate the impact of group size on performance by testing three different values of

Table A5. **Ablation on Weaker base models.**

Epoch	Exploration	Val14		Test14-hard		Test14-random	
		NR	R	NR	R	NR	R
500	IL	89.81	82.94	76.01	68.18	89.14	82.63
100	IL	84.63	73.30	71.21	59.49	85.61	75.30
	RFT w/o	83.13	74.53	72.68	59.77	85.87	76.61
	RFT w/	86.43	79.00	74.16	67.47	88.15	80.86

G_{grpo} in the Test14-hard benchmark. As shown in Tab. A6, using a group size of 4 results in suboptimal performance compared to larger group sizes. When the group size is increased to 8 or 12, performance improves, with scores stabilizing around 72.21 (R-score) and 77.16 (NR-score). We choose a group size of 8 to strike an optimal balance between performance and computational efficiency.

Ablation on Weaker Base Models. We treat models trained with fewer IL epochs as weaker base models and fine-tune under both settings (w/ and w/o the exploration policy), as shown in Tab. A5. Results show that RFT remains effective even with a weaker base.

E. Additional Qualitative Results

Additional Visualization of Policy-Guided Denoising. We show qualitative comparisons of planned trajectories from Diffusion Planner, DiffusionDrive, and PlannerRFT with policy-guided denoising, as shown in Fig. A3. With policy-guided denoising, PlannerRFT generates a group of multi-modality and scenario-adaptive trajectories for sampling efficient RL training.

Qualitative Results on Safety-Critical Scenarios. We present closed-loop planning results of PlannerRFT on safety-critical scenarios, as shown in Fig. A4 and Fig. A5. These examples demonstrate the planner's enhanced safety awareness, improved maneuver robustness, and better handling of dynamic interactions.

Qualitative Results on Obstacle Avoidance Scenarios. We present the closed-loop planning results for PlannerRFT on obstacle avoidance scenarios, as shown in Fig. A6 and Fig. A7. These examples demonstrate the planner's ability to laterally avoid obstacles and to execute precise maneuvers in narrow spaces.

Qualitative Results in Reactive Traffic. We present the closed-loop planning results for PlannerRFT in reactive traffic, as shown in Fig. A8 and Fig. A9. These examples demonstrate the planner's enhanced decision-making and planning capability in interactive scenarios.

Qualitative Results on Causal-Confusion Scenarios. We present closed-loop planning results on a causal-confusion scenario, as shown in Fig. A10, which illustrate the advantage of RL in mitigating the causal-confusion issues inherent in imitation learning.

Table A6. **Ablation on Group number.** Results are reported on the Test14-random benchmark.

G_{grpo}	Closed-loop metrics ↑					
	R-score	Coll.	Driv.	C.	Prog.	**NR-score**
4	71.24	86.40	94.85	84.93	**78.99**	76.31
8	72.21	**88.97**	**95.59**	**85.66**	77.17	**77.16**
12	**72.29**	**88.97**	95.22	**85.66**	77.62	77.04

F. License of Assets

Data for nuPlan [19] are provided under the CC-BY-NC 4.0 license. Our IL-pretrained model follows the implementation of Diffusion Planner [49]. As the original repository[11] does not provide an explicit license, the referenced code is used solely for academic research and reproducibility purposes, and all rights remain with the original authors. nuMax is a re-implementation of Waymax [10] for non-commercial research, in accordance with the Waymax License Agreement for Non-Commercial Use[12]. Scenario caching in nuMax is developed upon ScenarioMax, which is released under the Apache-2.0 license. Our RL training pipeline references V-Max [1] and Brax, distributed under the MIT License and Apache-2.0 License, respectively. The reinforcement learning algorithms further draw upon DPPO [36] and ReCogDrive [23], which are released under the MIT License and Apache-2.0 License.

[11] https://github.com/ZhengYinan-AIR/Diffusion-Planner

[12] https://github.com/waymo-research/waymax/blob/main/LICENSE

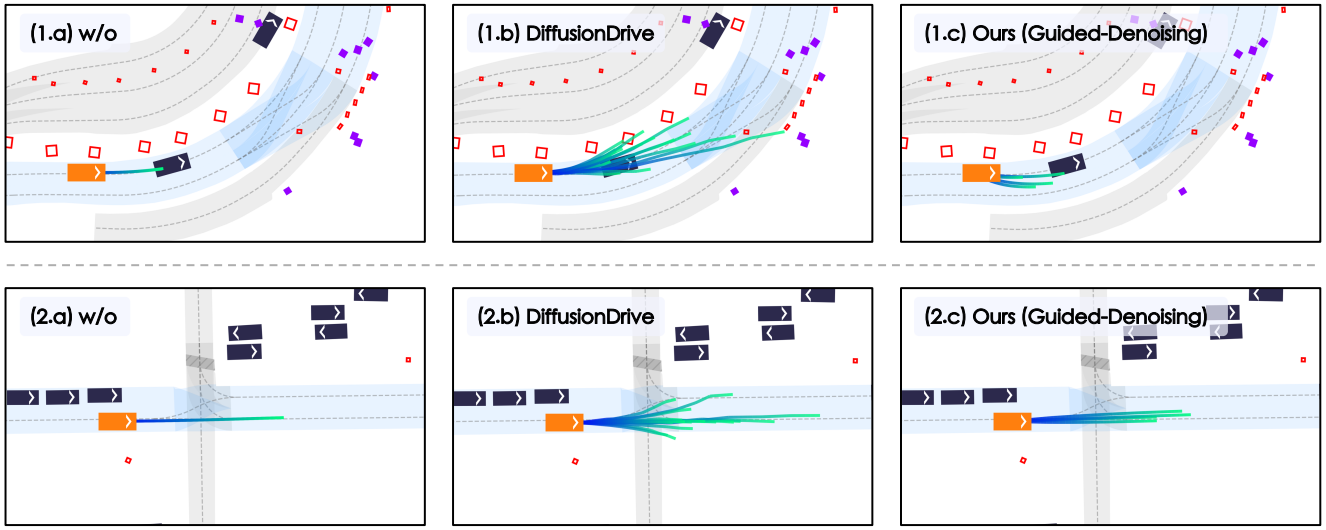

Figure A3. **Visualization of Diffusion Planner (a, w/o guided denoising), DiffusionDrive (b) and PlannerRFT (c, w/ guided denoising).** For Diffusion Planner and PlannerRFT, we resample 4 trajectories, for DiffusionDrive we use 20 anchor noises. We visualize the planned trajectory over a 4 *second* horizon. Note that DiffusionDrive is evaluated on the NAVSIM `navtest` split with camera and LiDAR inputs; for visualization, we render all planners on the same scenario shared between NAVSIM and nuPlan. PlannerRFT demonstrates multi-modal and scenario-adaptive trajectory generation through policy-guided denoising.

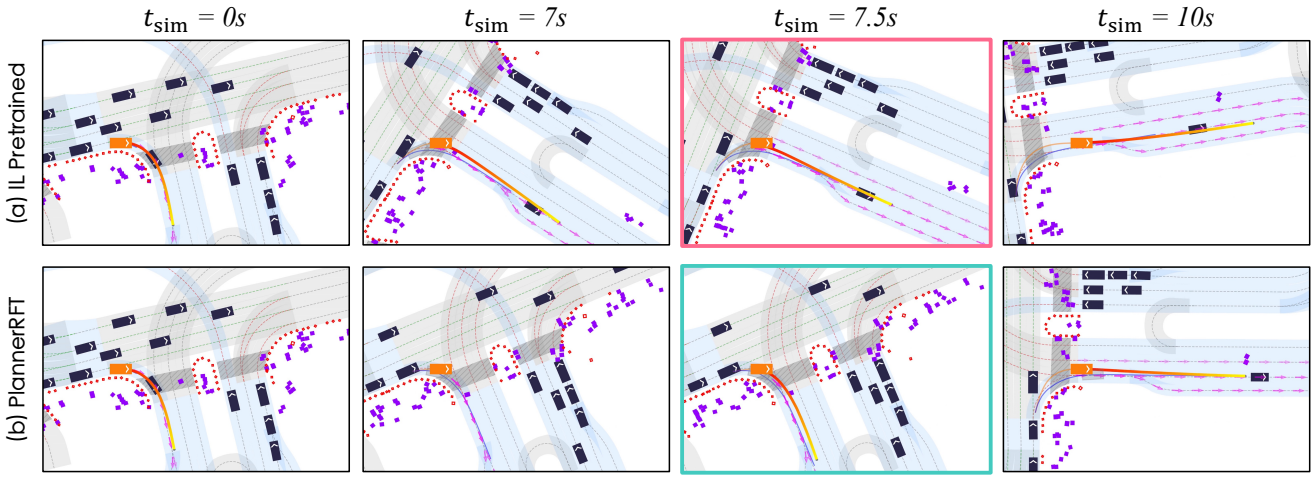

Figure A4. **Intersection Pedestrian Avoidance.** The ego vehicle intends to make a right turn at an intersection while pedestrians are crossing. **(a)** The IL Pretrained planner collision with a pedestrian at $t_{\text{sim}} = 7.5\ s$. **(b)** PlannerRFT waits for all pedestrians to finish crossing and then proceeds with the right turn. In each frame shot, the simulation position and planning trajectory are marked as orange, the ground-truth position and ground-truth trajectory recorded in the driving log are marked as gray and blue, respectively. Surrounding vehicles are marked as black rectangles with white arrows indicating heading. The pedestrians are marked as purple, and the static objects are marked as red. The lane polygons are marked gray and the navigation routes are marked as light blue with centerline arrows.

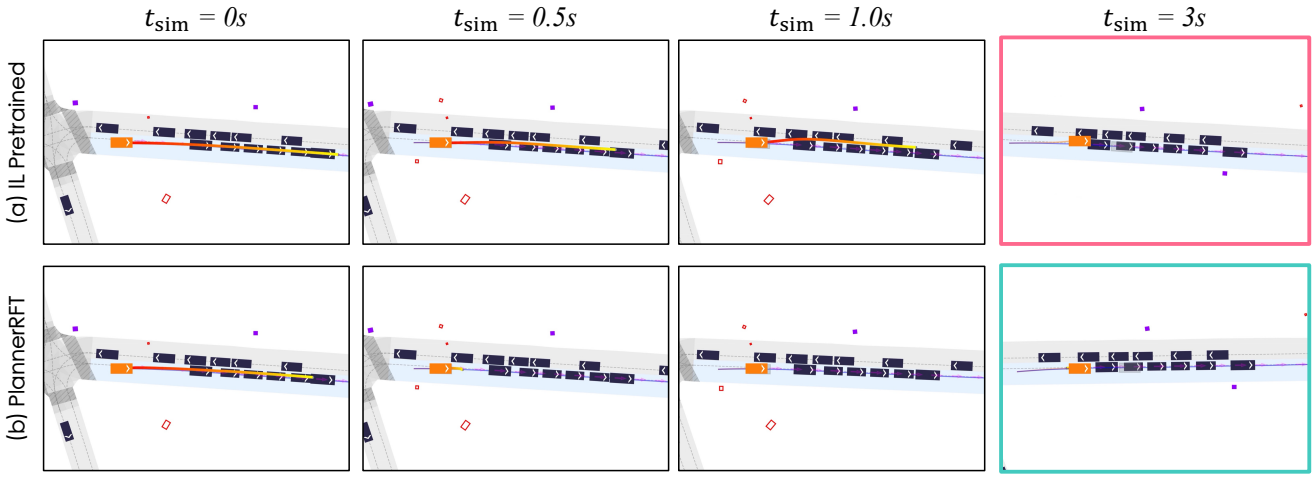

Figure A5. **Emergency Brake in Reactive Traffic.** A safety-critical scenario in which surrounding vehicles governed by the IDM policy enter a deadlock at the initial timestep ($t_{\text{sim}} = 0\ s$), blocking all traffic, while the ego vehicle approaches at high speed as recorded in the log. **(a)** The IL-pretrained planner fails to brake in time and collides with the preceding vehicle. **(b)** PlannerRFT detects the stationary lead vehicle and applies braking at $t_{\text{sim}} = 1\ s$, successfully avoiding the collision.

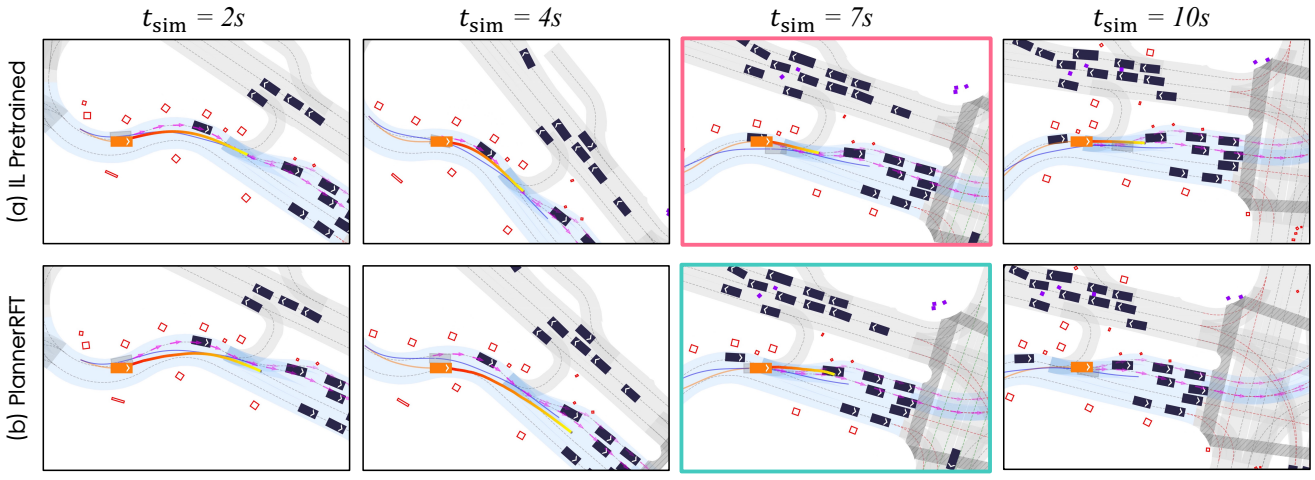

Figure A6. **S-Curve Lane Change.** The ego vehicle starts in the left lane of an S-curve, with a stationary vehicle ahead in the same lane. **(a)** The IL-pretrained planner keeps the lane and collides with the stationary vehicle at $t_{\text{sim}} = 7\ s$. **(b)** PlannerRFT performs a lane change to the right at $t_{\text{sim}} = 4\ s$, bypassing the stationary vehicle.

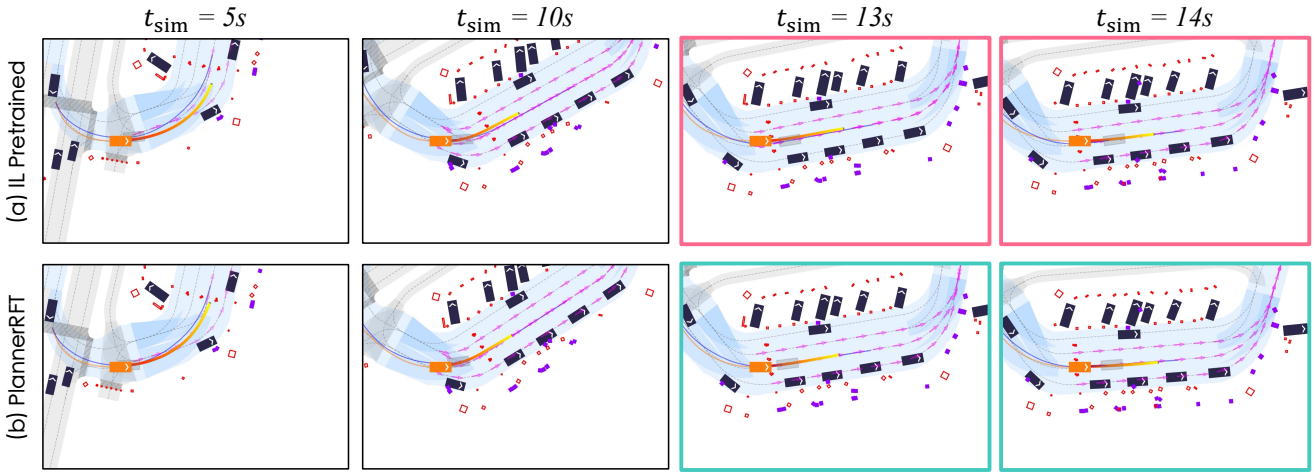

Figure A7. **Traffic-Cone Narrowing.** The ego vehicle is driving on a curved road with traffic cones. **(a)** The IL Pretrained planner based ego vehicle fails to avoid the traffic cone in time, colliding with it at $t_{\text{sim}=13\,s}$. **(b)** PlannerRFT enables the ego vehicle to finely adjust the trajectory, successfully steering the ego vehicle between the two cones.

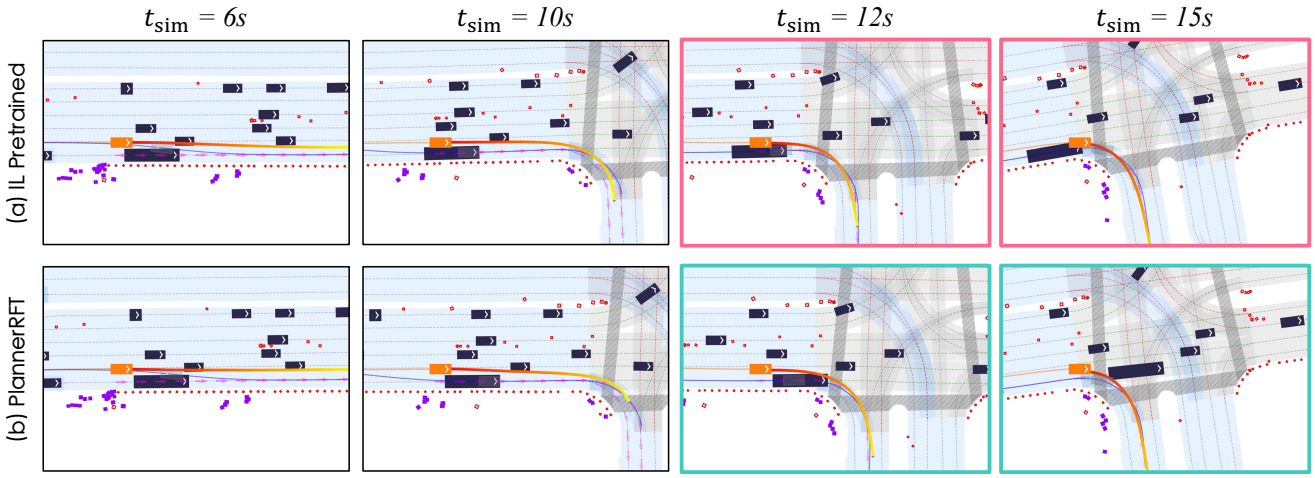

Figure A8. **Blocked Right-Turn in Reactive Traffic.** The ego vehicle intends to turn right at the upcoming intersection. **(a)** The IL Pretrained planner causes the ego vehicle to forcibly change lanes, leading to a collision with a long vehicle on the right at $t_{\text{sim}} = 12\ s$. **(b)** PlannerRFT enables the ego vehicle to consider the long vehicle proceeding straight, hence the ego vehicle decides to delay the lane change, which avoiding a collision.

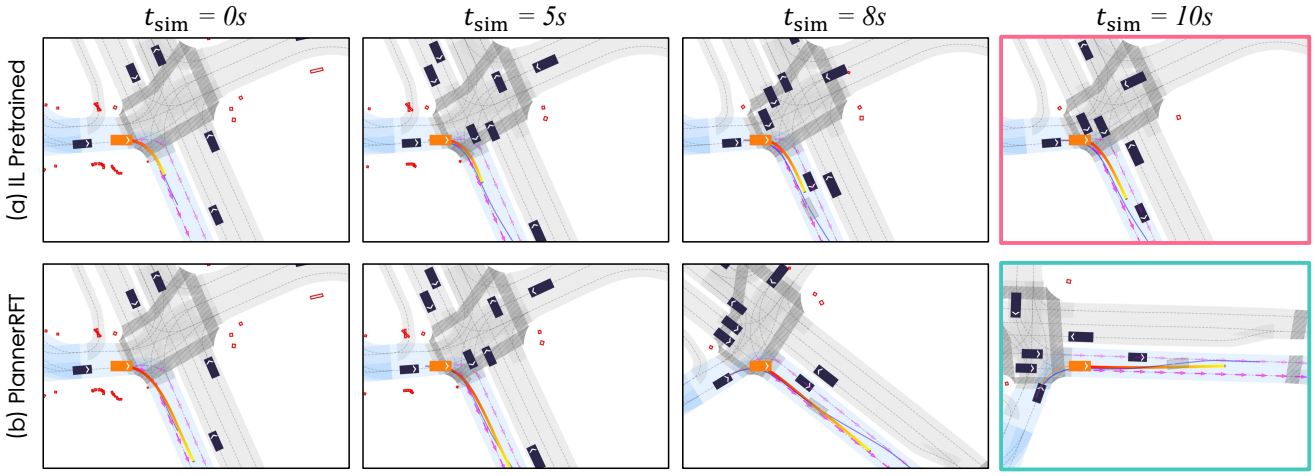

Figure A9. **Unprotected Right-Turn in Reactive Traffic.** The ego vehicle attempts a right turn at an intersection while surrounding vehicles approach from the cross traffic. **(a)** The IL Pretrained planner causes the ego vehicle to hesitate when turning right, ultimately leading to a collision with an oncoming vehicle from the left at $t_{\text{sim}} = 10\ s$. **(b)** PlannerRFT enables the ego vehicle smoothly and successfully completes the right turn before the arrival of the oncoming vehicle from the left.

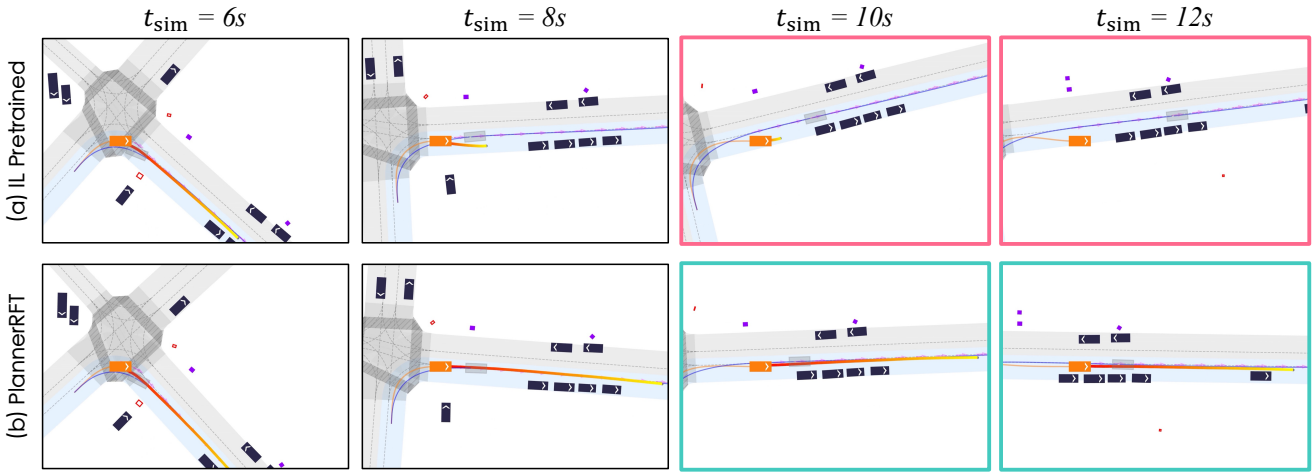

Figure A10. **A Causal-Confusion Scenario** The ego vehicle turns right at an intersection. **(a)** The IL pretrained planner directs the ego vehicle to turn right and then pull over to the side of the road. Off-road at $t_{\text{sim}=10\ s}$. This behavior is likely due to causal confusion: a large number of scenarios in the training data where vehicles turn right and stop to pick up passengers. **(b)** PlannerRFT guides the ego vehicle to turn right and then proceed straight, a maneuver that aligns with common sense and avoids causal confusion.