ReconDreamer-RL: Enhancing Reinforcement Learning via Diffusion-based Reconstruction

Chaojun Ni[1,2*], Guosheng Zhao[2*], Xiaofeng Wang[2*], Zheng Zhu[2* †], Wenkang Qin[2], Xinze Chen[2], Guanghong Jia[3], Guan Huang[2], and Wenjun Mei[1†]

[1]Peking University, Beijing, China
[2]GigaAI, Beijing, China
[3]Tsinghua University, Beijing, China
[*]These authors contributed equally to this work.
[†] Corresponding authors: zhengzhu@ieee.org, mei@pku.edu.cn.
Project Page: https://recondreamer-rl.github.io/

Abstract. Reinforcement learning for end-to-end autonomous driving in closed-loop simulations is gaining increasing attention, yet most simulators differ substantially from real-world conditions, leading to a significant sim2real gap. Recent methods use scene reconstruction to build photorealistic simulators, improving sensor realism but remaining constrained by the training data distribution, which limits their ability to render novel trajectories and corner cases. To address this, we propose *ReconDreamer-RL*, a framework that integrates video diffusion priors into scene reconstruction for reinforcement learning, enabling more realistic and diverse closed-loop autonomous driving training. Specifically, we introduce ReconSimulator, which combines video diffusion priors for appearance modeling with a kinematic model for physical modeling, reconstructing driving scenarios from real-world data and narrowing the sim2real gap. To cover more corner cases, we propose the Dynamic Adversary Agent (DAA), which adjusts surrounding vehicles' trajectories relative to the ego vehicle to generate challenging scenarios such as cut-ins. Finally, we introduce the Cousin Trajectory Generator (CTG) to mitigate the bias of training trajectories toward simple straight-line motion. Experiments show that *ReconDreamer-RL* improves end-to-end autonomous driving training and outperforms imitation learning methods with a 5× reduction in Collision Ratio.

Keywords: Autonomous Driving · Reinforcement Learning · Scene Reconstruction · Video Diffusion Models · Sim-to-Real

1 INTRODUCTION

End-to-end training of autonomous driving models [21, 28, 30, 32, 46, 49, 57, 59] through reinforcement learning in closed-loop simulation environments attracts growing interest. Compared to imitation learning, which relies solely on expert-collected demonstrations, closed-loop reinforcement learning enables the model

to interact with the environment, enhancing its robustness and adaptability across diverse scenarios.

Despite these advantages, existing methods still face significant challenges. One key issue is creating realistic driving environments capable of effective interactions with autonomous driving policies. Game-engine-based simulators [1, 67] lack sensor-level realism, whereas real-world closed-loop training is costly and risky. To overcome these limitations, recent approaches [12, 13, 15, 31, 50, 56] employ scene reconstruction methods to build photorealistic digital twins of real-world scenarios. However, these reconstruction-based methods are constrained by their training data, typically generating high-quality sensor outputs only within regions covered by recorded camera trajectories. Additionally, these methods fail to adequately account for corner cases such as sudden braking, as such rare behaviors are typically absent from the reconstruction data.

In this paper, we introduce *ReconDreamer-RL*, a framework that integrates the video diffusion priors [48,63] into scene reconstruction to enhance end-to-end autonomous driving training by reinforcement learning. The framework consists of three components: the ReconSimulator, the Dynamic Adversary Agent (DAA), and the Cousin Trajectory Generator (CTG). *ReconDreamer-RL* enhances the training process in two key stages: (1) The imitation learning stage uses behavior cloning for plan initialization. (2) The reinforcement learning stage optimizes the policy through closed-loop trial-and-error interactions with the environment. Specifically, ReconSimulator employs 3D Gaussian Splatting (3DGS) to reconstruct the driving scene and incorporates the video diffusion prior for appearance modeling. Meanwhile, a kinematic model is used to ensure the validity of vehicles' trajectories during ego-vehicle movement and scene editing. Then, in the first stage, we propose that DAA enrich corner case scenarios by controlling surrounding vehicles to generate challenging situations such as sudden lane changes. Meanwhile, the CTG enhances the diversity of sensor-collected data by synthesizing new trajectories from expert trajectories, addressing the data bias toward straight-line driving. In the second stage, the autonomous driving policy is trained within ReconSimulator via reinforcement learning, while the DAA continues to dynamically alter surrounding vehicle trajectories to create corner cases.

Through evaluation in the closed-loop 3DGS environment, we demonstrate that the end-to-end driving policy trained with the *ReconDreamer-RL* framework performs better, especially in corner cases, with a 5× reduction in Collision Ratio compared to imitation learning methods.

The main contributions of this paper are summarized as follows:

- We introduce *ReconDreamer-RL*, the framework that enhances end-to-end autonomous driving training by scene reconstruction with diffusion prior. It uses ReconSimulator to create realistic and freely explorable environments for reinforcement learning by enhancing appearance and physical modeling, reducing the sim2real gap.
- We propose DAA to auto-generate corner case instances to improve scenario coverage. Additionally, the CTG is introduced to address data distribution

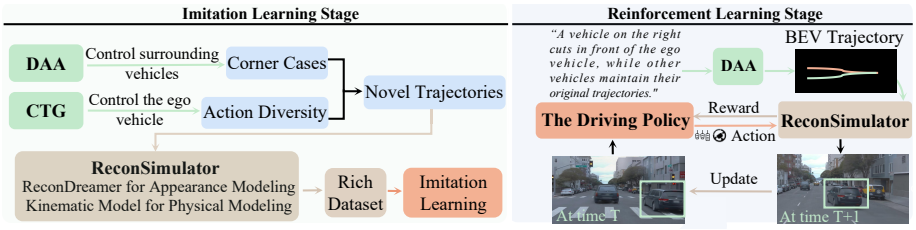

Fig. 1: In *ReconDreamer-RL*, ReconSimulator improves appearance modeling by ReconDreamer and incorporates physical modeling to reconstruct driving scenes. In the imitation learning stage, DAA generates corner-case scenario trajectories, while CTG diversifies the ego vehicle's actions and uses ReconSimulator to render sensor data for training the policy. In the reinforcement learning stage, the policy is trained in a closed-loop environment, interacting with DAA-controlled vehicles.

challenges by enriching sensor-collected expert trajectories, ensuring more balanced and comprehensive training data. This joint design further promotes model generalization and resilience.

– We validate the effectiveness of *ReconDreamer-RL* in a closed-loop 3DGS environment. *ReconDreamer-RL* performs stronger in challenging closed-loop evaluations, achieving a 5× reduction in collision rate compared to imitation learning methods.

2 Related Work

2.1 Driving Scene Reconstruction with Diffusion Prior

3DGS has become a prominent technique in 3D scene reconstruction. Various works [6, 7, 20, 27, 53] have applied 3DGS to autonomous driving scenarios. However, these methods exhibit significant performance degradation when rendering trajectories that deviate from the training distribution. Although some methods [53, 66] leverage 3DGS with decomposed foreground and background layers to ensure accurate foreground localization, the background still suffers from substantial artifacts under novel viewpoints, for example, blurred lane markings. This poses a challenge for end-to-end autonomous driving policies, which rely heavily on background cues such as lane boundaries for reliable decision-making, making these methods unsuitable for closed-loop evaluation or as reinforcement learning simulators. To address this issue, some studies leverage video diffusion models [2, 38] to enhance the representation of driving scenes. DriveDreamer4D [61] is the first to leverage the video diffusion prior to generate novel trajectory videos for training reconstruction models. ReconDreamer [35] further improves reconstruction by integrating the video diffusion model with online artifact correction, enabling better rendering of large maneuvers. ReconDreamer++ [62] enhances rendering quality by reducing domain gaps and refining the ground surface. However, these works focus on improving scene representation. In contrast, we leverage the video diffusion prior to improve the

simulator's appearance modeling and introduce physical modeling, enhancing end-to-end autonomous driving training by reinforcement learning.

In terms of scene generation methods, DriveDreamer-2 [63] allows users to create corner case scenarios more effectively. However, its generation method is relatively inefficient and unsuitable for reinforcement learning training. In contrast, DiffScene [51] introduces a diffusion-based framework that efficiently generates high-quality, safety-critical driving scenarios, significantly enhancing the evaluation of autonomous vehicles.

2.2 End-to-End Autonomous Driving

Recent advancements in end-to-end autonomous driving algorithms showcase the immense potential of learning-based planning, where raw sensor inputs are directly mapped to control outputs. UniAD [18] integrates various perception tasks [8, 10, 11, 33, 52, 60, 68] to enhance planning performance. VAD [22] focuses on improving trajectory planning by utilizing compact vectorized scene representations. Furthermore, VADv2 [5] extends this by introducing a framework that models probability distributions over planning vocabularies. However, methods based on imitation learning rely heavily on expert demonstrations and suffer from poor generalization. To address this issue, some approaches [4, 16, 23, 26, 41, 58] utilize deep reinforcement learning [24,34,36,39,40,42,43] for training end-to-end autonomous driving models. Nevertheless, these methods often encounter limitations, either due to simulation environments lacking photorealistic fidelity [14] or reliance on open-loop data. Recent work, RAD [15], introduces a reinforcement learning framework specifically designed for training autonomous driving agents in photorealistic 3DGS environments. However, RAD [15] still suffers from inadequate supervision for policy learning due to limitations in rendering novel views and a lack of corner-case scenarios in reconstructed data, leading to limited generalization and a sim-to-real gap.

3 Method

3.1 Preliminary

ReconDreamer [35] integrates the video diffusion prior to improve the performance of reconstruction methods in handling large maneuvers. The core of ReconDreamer is DriveRestorer, designed to restore artifacts in rendered videos through an online restoration process, which mitigates distortions and enhances consistency across frames. It is fine-tuned based on the video diffusion models [48, 63] and uses a diffusion loss as follows:

$$\mathcal{L}_{\mathcal{R}} = \mathbb{E}_{\boldsymbol{z},\epsilon\sim\mathcal{N}(0,1),t}\left[\left\|\epsilon_t - \epsilon_\theta\left(\boldsymbol{z}_t, t, \boldsymbol{c}\right)\right\|_2^2\right], \tag{1}$$

where ϵ_t is the random noise at time step t, ϵ_θ is the denoising network used to predict the noise, $\boldsymbol{z}t$ is the noisy latent variable at step t, and $\boldsymbol{c}$ represents

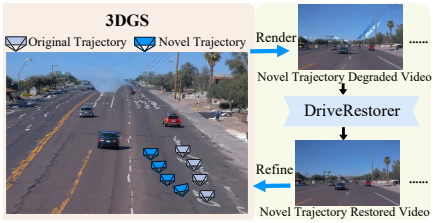

Fig. 2: The process of integrating the diffusion prior for appearance modeling. During the reconstruction of driving scenes, we first render novel trajectory view videos. These rendered videos are then processed by the DriveRestorer to enhance their visual quality, and the restored results are used to further optimize the reconstruction model. This iterative process continues until the reconstruction model converges.

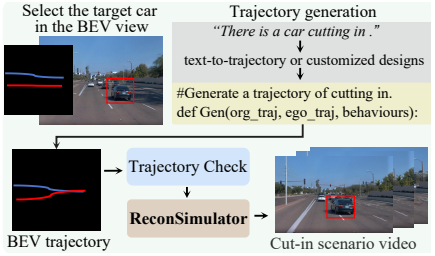

Fig. 3: The pipeline of the DAA. DAA identifies the target vehicles based on their distances to the ego car from the BEV view, where the blue line represents the ego car's trajectory and the red line represents the target vehicle. Then, DAA generates novel trajectories based on the specified interactive behavior. The generated trajectories are checked, and feasible ones are rendered using ReconSimulator.

the control conditions, which include the degraded video $\hat{V}$novel, 3D bounding boxes, and HDMaps, thus incorporating both low-level visual cues and high-level structural information to guide accurate video restoration.

During the inference stage, DriveRestorer freezes its network parameters and restores novel trajectory renderings. The inference process is expressed as:

$$V_{\text{novel}} = \mathcal{R}(\hat{V}_{\text{novel}}, s), \tag{2}$$

where $\mathcal{R}$ represents the DriveRestorer model, and s denotes the structural conditions corresponding to the degraded video $\hat{V}_{\text{novel}}$ (such as 3D bounding boxes and HDMaps). In *ReconDreamer-RL*, we leverage DriveRestorer to enhance scene rendering quality, enabling free ego car navigation and scene editing, as detailed in the ReconSimulator Section.

3.2 Overview of the *ReconDreamer-RL* framework.

Existing end-to-end autonomous driving algorithms are typically trained based on imitation learning, leading to poor generalization. To address this, RAD [15] trains the policy in 3DGS-based simulations, which provide photorealistic sensor data. However, it still faces challenges in rendering novel views and covering diverse corner cases.

We introduce the *ReconDreamer-RL* pipeline in Fig. 1. ReconSimulator first reconstructs the driving scene, providing high-fidelity sensor data for ego vehicle navigation by integrating video diffusion priors and physical modeling, ensuring more accurate and realistic environmental representations. In the imitation learning stage, DAA generates corner cases specifically designed for imitation

learning, while CTG improves driving behavior through a variety of diverse and challenging actions. In the reinforcement learning stage, the policy is trained in closed-loop environments, with DAA generating new corner case trajectories to increase difficulty and enhance the robustness of the policy.

3.3 ReconSimulator

The ideal reinforcement learning environment for autonomous driving should minimize the sim2real gap. However, existing 3DGS-based methods merely reconstruct data, failing to ensure realistic appearances along novel trajectories. Meanwhile, it is essential to ensure that the trajectories of all vehicles adhere to physical constraints. Therefore, we propose ReconSimulator to address appearance and physical modeling challenges. In terms of appearance, we integrate the diffusion prior to enhance scene representation for high-quality sensor data rendering on novel trajectories. For physical modeling, kinematic modeling is used to ensure the physical feasibility of vehicle trajectories. Next, we delve into the details of the appearance and physical modeling.

Appearance Modeling. To ensure high-quality rendering and scene editing, we first use 3DGS to reconstruct the scene and render novel trajectories. Then, DriveRestorer [35] mitigates artifacts in the rendered videos, and the results are used to fine-tune the reconstruction model. This enables the final model to produce high-quality rendering from diverse viewpoints. The overall process is illustrated in Fig. 2. Meanwhile, we model the static background and moving vehicles separately, which enables the modification of trajectories and appearances for all vehicles. The background model is represented by $G_{\text{Background},w}$ in the world coordinate system. For each moving object v, represented by Gaussians $G^v_{\text{Rigid},l}$ in the local coordinate system l, these Gaussians must be transformed into the background coordinate system during the rendering process:

$$G^v_{\text{Rigid},w}(t) = M^v_t \cdot G^v_{\text{Rigid},l} + S^v_t, \tag{3}$$

where M^v_t and S^v_t represent the rotation matrix and translation vector that describe the pose of object v at time t. This formulation ensures that moving vehicles can be accurately placed in relation to the static background as well as other dynamic agents during view rendering.

Physical Modeling. To maintain the authenticity of vehicle trajectories, a kinematic model is used to govern the vehicle's motion. Specifically, the vehicle's pose at time t in the world coordinate system is $W_t = [R_t \mid P_t] \in SE(3)$, where R_t is the rotation matrix and P_t is the position. At each time step, the pose is updated using linear velocity v_t and steering angle δ_t in a kinematic bicycle model:

$$P_{t+1} = P_t + v_t \cdot \Delta t \cdot \hat{d}_t, \tag{4}$$

where $\hat{d}_t$ is the forward direction vector derived from the rotation matrix R_t. The vehicle orientation is updated by applying a rotation about the vertical axis:

$$R_{t+1} = \text{Rot}_z(\Delta\theta_t) \cdot R_t, \tag{5}$$

Fig. 4: Examples of Dynamic Adversary Agent (DAA) controlling surrounding vehicles to simulate cut-in scenarios.

where the incremental rotation angle $\Delta\theta_t$ is calculated based on the bicycle model as:

$$\Delta\theta_t = \frac{v_t}{L}\tan(\delta_t)\Delta t. \tag{6}$$

The L denotes the wheelbase length of the vehicle. The rotation matrix $\mathrm{Rot}_z(\Delta\theta_t)$ is explicitly expressed as:

$$\mathrm{Rot}_z(\Delta\theta_t) = \begin{bmatrix} \cos(\Delta\theta_t) & \sin(\Delta\theta_t) & 0 \\ -\sin(\Delta\theta_t) & \cos(\Delta\theta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

We define distinct kinematic parameters for each vehicle category and, during every trajectory update, perform explicit feasibility checks to ensure the resulting motion remains physically plausible and consistent with the vehicle's operational limits. These checks enforce constraints such as maximum steering angle, steering-rate bounds, and allowable velocity (and acceleration) ranges, preventing unrealistic maneuvers and guaranteeing that updated trajectories can be executed by the corresponding vehicle model.

3.4 Dynamic Adversary Agent

Existing end-to-end autonomous driving models are typically trained with limited exposure to challenging corner cases. To tackle this limitation, we propose the Dynamic Adversary Agent (DAA), which generates highly realistic and diverse corner case interactions. The overview of DAA is provided in Fig. 3, and as illustrated in Fig. 4, it can effectively generate complex cut-in scenarios.

DAA first identifies suitable interactive target vehicles from the Bird's-Eye View (BEV) perspective, based on the distances to the ego vehicle and the specified interactive behavior $\mathcal{B}$. Different interactive behaviors correspond to different distance settings, which refer to traditional traffic flow methods [25,47]. Then, a new trajectory is generated for the selected target vehicle by modifying its original trajectory T_{target} based on both the ego vehicle's trajectory T_{ego} and $\mathcal{B}$. The new trajectory T'_{target} can be represented as:

$$T'_{\text{target}} = f(T_{\text{ego}}, T_{\text{target}}, \mathcal{B}), \tag{8}$$

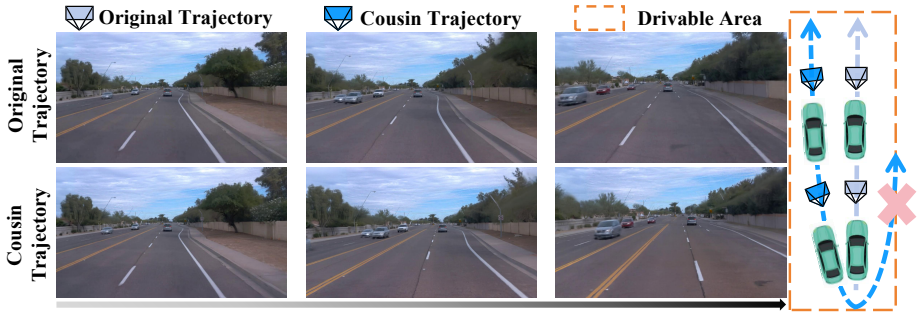

Fig. 5: Cousin Trajectory Generator (CTG) generates cousin trajectories and performs trajectory checks to eliminate unreasonable trajectories (e.g., the pink cross marks), and finally renders the corresponding sensor data in the ReconSimulator.

where $f(\cdot)$ represents the trajectory generation function, which can be implemented using methods such as text-to-trajectory [63] and customized based on specific requirements. Then, the generated trajectory is checked for feasibility. First, the vehicle trajectory T'_{target} should remain within the drivable region, and avoid collisions with other vehicles (collisions with the ego-vehicle are allowed):

$$\|T'_{\text{target}} - o_j\| \geq d_{\min}, \quad \forall j \in \{1, \ldots, M\}, \tag{9}$$

where $d_{\min}$ is the minimum distance between different agents $\{o_j\}_{j=1}^{M}$. Then, the trajectory is further checked to meet the constraints of the kinematic model and converted into the BEV perspective to verify whether it satisfies the interactive behavior $\mathcal{B}$.

DAA can be used in both imitation learning and reinforcement learning stages. In the first stage, DAA generates corner case scenarios by controlling surrounding vehicles and generating the ego vehicle's trajectory to avoid collisions, using a similar method mentioned above. The ReconSimulator then renders offline autonomous driving data based on the trajectories for imitation learning. In the second stage, the reinforcement learning policy controls the actions of the ego vehicle, while DAA controls the trajectory of the target vehicle in ReconSimulator to create corner cases. To enhance robustness, DAA has a certain probability of fine-tuning the trajectories (e.g., adjusting the target vehicle's speed) instead of directly reusing them from the first stage.

3.5 Cousin Trajectory Generator

To address cold-start issues in reinforcement learning [19,37], behavior cloning is an effective pretraining strategy but requires large-scale real-world data, which are often biased toward simple scenarios [9,44,65]. We propose the Cousin Trajectory Generator (CTG) to enhance action diversity, thereby creating the CousinnuScenes dataset with diverse actions.

For trajectory extension, we generate new trajectories for the ego vehicle, such as lane changes and sharp turns, following a process similar to that described in Equation 8. Subsequently, a trajectory check is conducted to evaluate the validity of the generated trajectories, ensuring they remain physically plausible and adhere to the vehicle's operational limitations, including steering angle and velocity, and verifying collision avoidance from the BEV perspective. Meanwhile, to better leverage expert trajectories from rare scenarios like U-turns, we interpolate the expert data for more detailed driving information. Given expert trajectories $\mathbf{X}_{\text{ego}} = \{X_{\text{ego}}^{t_1}, X_{\text{ego}}^{t_2}, \ldots, X_{\text{ego}}^{t_n}\}$, where each $X_{\text{ego}}^{t_i}$ is the ego vehicle's position at time t_i, we perform linear interpolation between consecutive time steps t_i and t_{i+1}. For an interpolated point X_{ego}^{t} at time t, where $t_i \leq t \leq t_{i+1}$, the formula is:

$$X_{\text{ego}}^{t} = X_{\text{ego}}^{t_i} + \frac{t - t_i}{t_{i+1} - t_i} \left(X_{\text{ego}}^{t_{i+1}} - X_{\text{ego}}^{t_i} \right). \tag{10}$$

The time t is expressed as:

$$t = t_i + k \cdot \Delta t, \quad k \in \{1, 2, \ldots, m\}, \tag{11}$$

where $\Delta t = \frac{t_{i+1} - t_i}{m+1}$ and m is the number of interpolation points between two time steps. For each interpolated trajectory point X_{ego}^{t}, we accordingly adjust the positions of surrounding vehicles to ensure the interpolated trajectories maintain a realistic spatial relationship and interactions.

After generating the interpolation and extension trajectories, we use the ReconSimulator to obtain the corresponding sensor data, which is then used for imitation training. We provide examples of trajectory extension in Fig.5, illustrating two cases of ego-vehicle lane-changing maneuvers. Moreover, as shown in Fig.6, Cousin-nuScenes created by CTG contains 4× more non-straight-line driving maneuvers than the nuScenes [3] dataset, thereby enriching the diversity of motion patterns and better covering long-tail behaviors that are underrepresented in the original data.

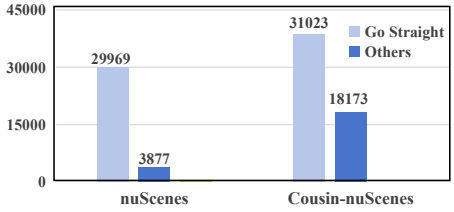

Fig. 6: CTG creates Cousin-nuScenes with more diverse actions than nuScenes.

4 Experiments

In this section, we present our experimental setup, detailing the implementation specifics, the baseline used for evaluation, and the evaluation metrics. Subsequently, we present both quantitative and qualitative results, highlighting the enhanced performance achieved by *ReconDreamer-RL*. Finally, we perform ablation studies to assess the individual contributions of each component.

Table 1: Comparison of different methods on various metrics.

Method	CR ↓	DCR↓	SCR↓	DR↓	PDR↓	HDR↓
VAD [22]	0.386	0.234	0.152	0.163	0.103	0.060
GenAD [64]	0.333	0.190	0.143	0.146	0.093	0.053
VADv2 [5]	0.290	0.162	0.128	0.154	0.107	0.047
RAD [15]	0.238	0.143	0.095	0.084	0.057	0.027
ReconDreamer-RL	**0.077**	**0.048**	**0.029**	**0.040**	**0.027**	**0.013**

4.1 Experimental Setup

Implementation Details. We evaluate the effectiveness of the *ReconDreamer-RL* framework in enhancing end-to-end policy training by applying it to RAD [15]. RAD takes multi-view camera images as input, converts them into scene-level representations, and predicts a probabilistic distribution over driving actions. It consists of an image encoder, a BEV encoder, a map head, an agent head, and a planning head. The BEV encoder transforms perspective-view features into BEV representations [29, 54]. The map head predicts vectorized road elements, the agent head estimates the states and future trajectories of surrounding dynamic agents, and the planning head outputs the final action distribution.

The overall training pipeline contains two stages: imitation learning and reinforcement-learning fine-tuning. In the imitation-learning stage, we first strengthen the perception modules using ground-truth annotations of map elements and agent motions. After this perception warm-up, the BEV encoder, map head, and agent head are frozen, and supervised behavior cloning is used to initialize the planning policy. The imitation data are collected from three sources: expert demonstrations from nuScenes [3], corner cases generated by the Dynamic Adversary Agent (DAA), and Cousin-nuScenes data generated by the Cousin Trajectory Generator (CTG). During behavior cloning, only the image encoder and planning head are updated.

In the reinforcement-learning stage, we further optimize the policy in closed-loop 3DGS environments reconstructed from nuScenes [3]. As shown in Fig. 7, we launch N parallel ReconSimulator workers, each initialized with a randomly sampled scenario. The current policy controls the ego vehicle and continuously interacts with the environment to collect rollouts. Each rollout contains trajectories of the form $(s_t, a_t, r_{t+1}, s_{t+1}, \ldots)$, where s_t denotes the state at time step t, a_t is the executed action, r_{t+1} is the received reward, and s_{t+1} is the next state. These rollouts are stored and used to update the policy with Proximal Policy Optimization (PPO) [40].

To improve stability and prevent the policy from drifting away from human-like driving behavior, we alternate PPO-based reinforcement learning with imitation-learning updates. The imitation updates use the same three data sources as in the behavior-cloning stage, namely nuScenes demonstrations, DAA-generated corner cases, and Cousin-nuScenes data. After a fixed number of optimization steps, the updated policy is synchronized to all ReconSimulator workers for sub-

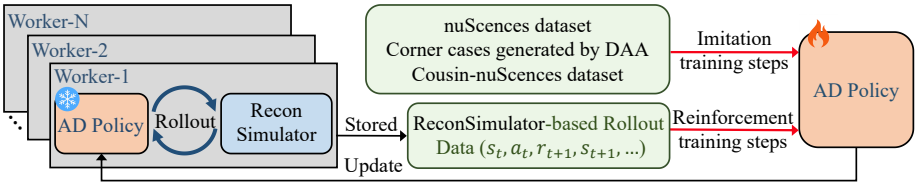

Fig. 7: Training procedure of the reinforcement-learning stage. We launch N parallel ReconSimulator workers, each initialized with a randomly sampled scenario. The current policy interacts with these environments to collect rollouts, which are stored for PPO updates. To stabilize training, PPO-based reinforcement learning is alternated with imitation-learning updates.

Table 2: Comparison of rendering speeds (FPS) among reinforcement learning simulators—EmerNeRF [55], RAD-3DGS [15], and ReconSimulator—measured under the same evaluation setting.

Simulator	EmerNeRF	RAD-3DGS	ReconSimulator
Speed (FPS)	0.21	135	125

Table 3: Metrics in cut-in scenarios.

Method	CR↓	DCR↓	SCR↓
VAD [22]	0.449	0.293	0.156
GenAD [64]	0.379	0.234	0.145
VADv2 [5]	0.436	0.276	0.160
RAD [15]	0.317	0.210	0.107
ReconDreamer-RL	**0.089**	**0.053**	**0.036**

sequent data collection. In this stage, the BEV encoder, map head, and agent head remain frozen, while the image encoder and planning head are fine-tuned. We further analyze the influence of the ratio between reinforcement-learning and imitation-learning updates on the final performance.

Baseline. In comparative analysis, we select representative end-to-end autonomous driving methods. For imitation learning, we choose VAD [22], GenAD [64], and VADv2 [5]. For reinforcement learning, we use RAD [15], which is trained in closed-loop 3DGS environments.

Evaluation Metrics. Following the methodology described in RAD [15], we assess policy performance using six metrics. Dynamic Collision Ratio (DCR) and Static Collision Ratio (SCR) measure collisions with dynamic and static obstacles, respectively, and their sum is reported as the Collision Ratio (CR) to reflect overall obstacle avoidance. Positional Deviation Ratio (PDR) and Heading Deviation Ratio (HDR) quantify deviations from the expert trajectory in position and heading, and are summed as the Deviation Ratio (DR). For comfort, we report Longitudinal Jerk (Long.Jerk) and Lateral Jerk (Lat.Jerk), which measure changes in acceleration along the longitudinal and lateral axes and indicate driving smoothness. To evaluate the realism of the reinforcement learning environment, we adopt three metrics from DriveDreamer4D [61]: Novel Trajectory Agent IoU (NTA-IoU) for foreground quality, Novel Trajectory Lane IoU (NTL-IoU) for lane-marking quality, and Fréchet Inception Distance (FID) [17] for overall rendering fidelity.

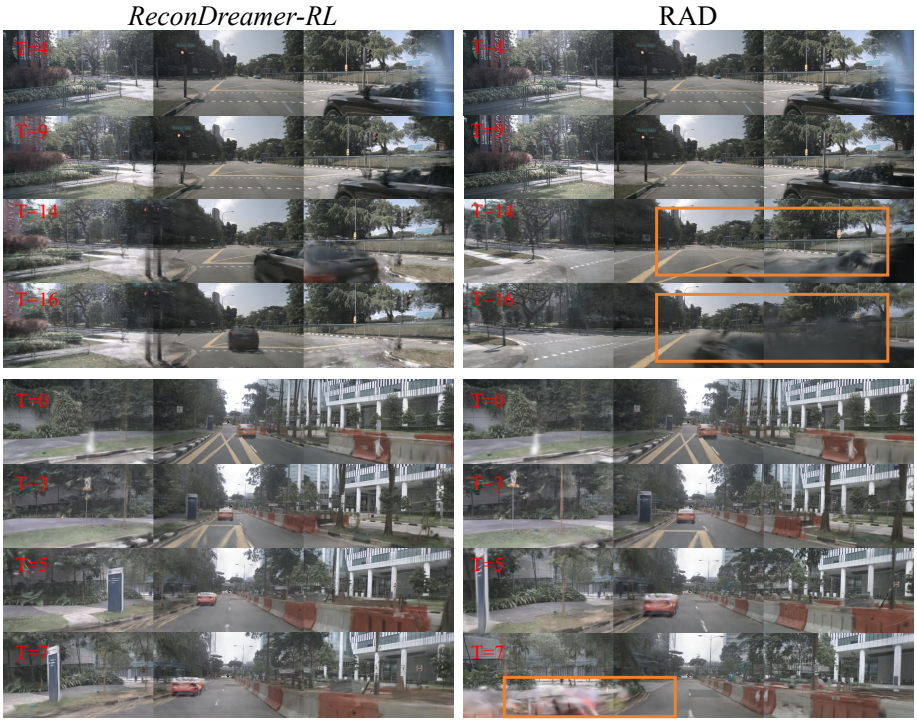

Fig. 8: Comparison of different methods in challenging corner cases, with collisions highlighted by orange boxes.

4.2 Main Results

Quantitative Results. *ReconDreamer-RL* enhances the performance of RAD [15] across all evaluation metrics, as shown in Tab. 1. Traditional imitation learning-based methods, such as VAD [22] and VADv2 [5], struggle with closed-loop evaluation tasks due to the gap between training and closed-loop inference. RAD [15] enhancing and training within a 3DGS-based simulator. However, due to limited corner case coverage and poor novel view rendering ability of 3DGS, RAD exhibits a higher collision rate in challenging evaluation benchmark. However, *ReconDreamer-RL* leverages diffusion prior and data augmentation to enhance end-to-end autonomous driving, achieving a 3× reduction in Collision Ratio.

Corner Case Results. As shown in Tab. 3, we compare the collision metrics of different methods in corner cases, focusing on cut-in situations. Imitation learning-based methods exhibit a high Dynamic Collision Ratio in cut-in corner cases, as such scenarios are completely absent from the training data. RAD [15] improves the model's ability to handle corner cases through reinforcement learning, but still lacks the necessary data to train on such situations. In contrast, *ReconDreamer-RL* improves the policy's performance through a two-stage process, achieving 404.5% improvement in Collision Ratio over imitation methods.

Table 4: Impact of diffusion prior within ReconSimulator on novel view rendering quality across lane shift and lane change.

Method	Lane Shift @ 3m			Lane Shift @ 6m			Lane Change		
	NTA-IoU ↑	NTL-IoU ↑	FID ↓	NTA-IoU ↑	NTL-IoU ↑	FID ↓	NTA-IoU ↑	NTL-IoU ↑	FID ↓
w/o Video Diffusion Prior	0.224	49.67	160.23	0.148	44.89	256.42	0.215	46.23	178.34
w/ Video Diffusion Prior	**0.342**	**50.12**	**130.48**	**0.325**	**49.72**	**125.43**	**0.337**	**50.01**	**122.67**

Table 5: Comfort analysis in unedited 3DGS environments.

Method	Long. Jerk↓	Lat. Jerk↓
VAD [22]	5.645	0.597
GenAD [64]	12.490	0.432
VADv2 [5]	8.124	0.435
RAD [15]	**4.237**	0.191
ReconDreamer-RL	4.264	**0.189**

Table 6: Comfort analysis in unedited in cut-in scenarios.

Method	Long. Jerk↓	Lat. Jerk↓
VAD [22]	6.714	0.784
GenAD [64]	14.216	0.548
VADv2 [5]	9.245	0.499
RAD [15]	4.912	0.316
ReconDreamer-RL	**3.832**	**0.276**

Rendering Speed. Rendering speed is crucial for reinforcement learning environments. As shown in Tab. 2, ReconSimulator achieves an efficient rendering speed of 125 FPS, which is significantly faster than EmerNeRF's 0.21 FPS and comparable to RAD-3DGS's 135 FPS. This high rendering speed enables ReconSimulator to satisfy the demanding requirements of reinforcement learning.

Qualitative Results. As shown in Fig. 8, we compare *ReconDreamer-RL* and RAD [15] under two corner cases. The first scenario involves a vehicle on the right of the ego car quickly cutting in. In this case, *ReconDreamer-RL* successfully avoids the collision and maintains a safe distance, while RAD [15] fails to react in time, leading to a potential collision due to inadequate planning. The second scenario features a vehicle on the right performing a cut-in, followed by a sudden brake. Although RAD [15] attempts to avoid the collision by shifting lanes, it fails to control its speed and steering angle properly, *ReconDreamer-RL* not only safely changes lanes but also decelerates smoothly, ensuring that the ego vehicle avoids the collision while maintaining a stable and controlled trajectory.

Comfort Results. As shown in Table 5, we compare the comfort analysis in unedited 3DGS environments. We observe that *ReconDreamer-RL* performs similarly to RAD [15] in terms of Long. Jerk and Lat. Jerk metrics, and significantly outperforms algorithms that rely solely on imitation learning. This is because *ReconDreamer-RL* employs a strategy that alternates between imitation learning and reinforcement learning during training, as detailed in the supplementary material. This approach allows *ReconDreamer-RL* to not only improve safety but also better align with human driving behavior. Additionally, we compare comfort metrics in cut-in scenarios, evaluating only the scenes where no collisions occur across all algorithms. We find that *ReconDreamer-RL* consistently delivers the highest performance in these scenarios, as it learns more effective collision avoidance strategies from a large set of corner cases, whereas other methods lack such data, leading to instability when encountering these corner cases.

Table 7: Computational overhead and generalization performance of ReconSimulator.

Model	NTA-IoU ↑	NTL-IoU ↑	FID ↓	Extra Diffusion Training Time (h)	Single Reconstruction Time (h)
RAD-3DGS [15]	0.219	48.72	149.50	–	1.5
ReconSimulator w/o finetune	0.287	49.75	144.31	–	1.8
ReconSimulator w/ finetune	0.317	**49.84**	127.10	3.0	1.8
ReconSimulator re-trained	**0.325**	49.72	**125.43**	12.0	1.8

4.3 Ablation Studies

ReconSimulator. As shown in Tab. 4, we conduct ablation experiments on ReconSimulator to demonstrate the impact of the diffusion prior on novel view rendering quality. Specifically, without ReconSimulator, the NTA-IoU and NTL-IoU drop by 54.46% and 9.71%, highlighting the importance of ReconSimulator. while the FID metric increases by 104.43% for a 6m lane shift, indicating significant degradation in the appearance of surrounding vehicles and lane markings. This suggests that the absence of ReconSimulator affects the visual consistency of the rendered scenes. Meanwhile, in rows 2, 3, and 4 of Tab. 8, we use RAD-3DGS to generate DAA and CTG data, instead of ReconSimulator, to train the RAD policy, while keeping the rendering process unchanged. Despite the lower rendering quality of RAD-3DGS, performance still improves due to the absence of corner cases present in the original data. As shown in rows 6, 7, and 8 of Tab. 8, using ReconSimulator to generate CTG and DAA data results in more significant performance gains. We also compare ReconSimulator with traditional 3DGS, which separates foreground and background. As shown in the red box of Fig. 9, ReconSimulator yields sharper lane markings and a cleaner background.

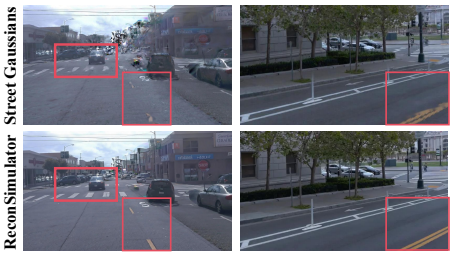

Fig. 9: Comparison of ReconSimulator and Street Gaussians in rendering novel views.

The Cost of integrating diffusion prior and the generalization ability of DriveRestorer . To analyze the cost of integrating the diffusion prior and the generalization ability of DriveRestorer, we compare different strategies for the ReconSimulator. First, we train a DriveRestorer model on the Waymo dataset [45] and apply it directly to the ReconSimulator for reconstructing the nuScenes dataset [3], assessing both computational overhead and generalization performance. As shown in Tab. 7, when the ReconSimulator utilizes the DriveRestorer model trained on the Waymo dataset for reconstruction, it still significantly improves reconstruction on nuScenes, demonstrating strong cross-dataset generalization. This generalization substantially reduces the cost of integrating the diffusion prior. Moreover, fine-tuning DriveRestorer on a target

Table 8: Ablation study of *ReconDreamer-RL*, evaluating the effectiveness of each module. When ReconSimulator is not used, we directly employ the RAD-3DGS [15] method to reconstruct the scenes.

ReconSimulator	DAA	CTG	CR↓	DCR↓	SCR↓	DR↓	PDR↓	HDR↓
			0.238	0.143	0.095	0.084	0.057	0.027
	✓		0.167	0.102	0.065	0.076	0.052	0.024
		✓	0.191	0.121	0.070	0.068	0.046	0.022
	✓	✓	0.142	0.082	0.060	0.063	0.043	0.020
✓			0.172	0.103	0.069	0.073	0.053	0.020
✓	✓		0.117	0.069	0.048	0.067	0.050	0.017
✓		✓	0.143	0.086	0.057	0.053	0.040	0.013
✓	✓	✓	**0.077**	**0.048**	**0.029**	**0.040**	**0.027**	**0.013**

dataset is relatively inexpensive, requiring only 3 hours, whereas training from scratch takes 12 hours, which remains reasonable. Additionally, this overhead is incurred only once for reconstructing all the scenes in the target dataset. During the scene reconstruction process, the integration of the diffusion prior introduces minimal overhead, adding only about 0.3 hours per scene.

Impact of the DAA. The DAA helps reduce collision occurrences by generating corner case scenarios for training. As shown in Tab. 8, incorporating DAA decreases the Collision Ratio from 0.172 to 0.117, demonstrating its effectiveness in improving model safety and reliability.

Impact of the CTG. In Tab. 8, the benefit of CTG is validated by reductions in CR (from 0.172 to 0.143) and DR (from 0.073 to 0.052), indicating improved trajectory adherence. By addressing the data bias, CTG enhances the action distribution learned during the imitation learning phase.

5 Conclusion

This paper introduces *ReconDreamer-RL*, a novel and comprehensive framework designed to enhance reinforcement learning for end-to-end autonomous driving through the integration of video diffusion priors. It focuses on constructing a more realistic simulator, enriching corner cases, and addressing the issue of training data distribution. Specifically, we introduce ReconSimulator, which combines the video diffusion prior for appearance modeling and a kinematic model for physical dynamics, providing realistic environments for reinforcement learning. Additionally, we develop the DAA, which generates complex traffic scenarios, such as sudden lane changes. To address the issue of training data distribution being mostly on simple straight-line movements, CTG is proposed to enhance trajectory diversity through extension and interpolation. Experimental evaluations indicate that *ReconDreamer-RL* significantly improves the performance of end-to-end autonomous driving models, reducing the Collision Ratio by 5× compared to imitation-based learning methods.

References

1. Automotive, I.: Carmaker. https://www.ipg-automotive.com (2023), accessed: 2025-06-22
2. Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., English, Z., Voleti, V., Letts, A., et al.: Stable video diffusion: Scaling latent video diffusion models to large datasets. arXiv preprint arXiv:2311.15127 (2023)
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2020)
4. Caesar, H., Kabzan, J., Tan, K.S., Fong, W.K., Wolff, E., Lang, A., Fletcher, L., Beijbom, O., Omari, S.: nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. arXiv preprint arXiv:2106.11810 (2021)
5. Chen, S., Jiang, B., Gao, H., Liao, B., Xu, Q., Zhang, Q., Huang, C., Liu, W., Wang, X.: Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. arXiv preprint arXiv:2402.13243 (2024)
6. Chen, Y., Gu, C., Jiang, J., Zhu, X., Zhang, L.: Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. International Journal of Computer Vision (2026)
7. Chen, Z., Yang, J., Huang, J., de Lutio, R., Esturo, J.M., Ivanovic, B., Litany, O., Gojcic, Z., Fidler, S., Pavone, M., et al.: Omnire: Omni urban scene reconstruction. In: The Thirteenth International Conference on Learning Representations (2025)
8. Cheng, J., Chen, Y., Chen, Q.: Pluto: Pushing the limit of imitation learning-based planning for autonomous driving. arXiv preprint arXiv:2404.14327 (2024)
9. Codevilla, F., Santana, E., López, A.M., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2019)
10. Delavari, E., Khalil, A., Kwon, J.: Caril: Confidence-aware regression in imitation learning for autonomous driving. arXiv preprint arXiv:2503.00783 (2025)
11. Di Palo, N., Johns, E.: Keypoint action tokens enable in-context imitation learning in robotics. arXiv preprint arXiv:2403.19578 (2024)
12. Ding, T., Xiang, D., Rivas, P., Dong, L.: Neural pruning for 3d scene reconstruction: Efficient nerf acceleration. arXiv preprint arXiv:2504.00950 (2025)
13. Ding, T.K., Xiang, D., Qi, Y., Yang, Z., Zhao, Z., Sun, T., Feng, P., Wang, H.: Nerf-based defect detection. In: International Conference on Remote Sensing, Mapping, and Image Processing (RSMIP 2025) (2025)
14. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning (2017)
15. Gao, H., Chen, S., Jiang, B., Liao, B., Shi, Y., Guo, X., Pu, Y., Yin, H., Li, X., Zhang, X., et al.: Rad: Training an end-to-end driving policy via large-scale 3dgs-based reinforcement learning. arXiv preprint arXiv:2502.13144 (2025)
16. Gao, H., Zhao, M., Zheng, X., Wang, C., Zhou, L., Wang, Y., Ma, L., Cheng, B., Wu, Z., Li, Y.: An improved hierarchical deep reinforcement learning algorithm for multi-intelligent vehicle lane change. Neurocomputing (2024)
17. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. NeurIPS (2017)

18. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al.: Planning-oriented autonomous driving. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2023)
19. Hua, J., Zeng, L., Li, G., Ju, Z.: Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning. Sensors (2021)
20. Huang, N., Wei, X., Zheng, W., An, P., Lu, M., Zhan, W., Tomizuka, M., Keutzer, K., Zhang, S.: s^3gaussian: Self-supervised street gaussians for autonomous driving. arXiv preprint arXiv:2405.20323 (2024)
21. Huang, Z., Weng, X., Igl, M., Chen, Y., Cao, Y., Ivanovic, B., Pavone, M., Lv, C.: Gen-drive: Enhancing diffusion generative driving policies with reward modeling and reinforcement learning fine-tuning. arXiv preprint arXiv:2410.05582 (2024)
22. Jiang, B., Chen, S., Xu, Q., Liao, B., Chen, J., Zhou, H., Zhang, Q., Liu, W., Huang, C., Wang, X.: Vad: Vectorized scene representation for efficient autonomous driving. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2023)
23. Jiang, B., Chen, S., Zhang, Q., Liu, W., Wang, X.: Alphadrive: Unleashing the power of vlms in autonomous driving via reinforcement learning and reasoning. arXiv preprint arXiv:2503.07608 (2025)
24. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al.: Highly accurate protein structure prediction with alphafold. nature (2021)
25. Kesting, A., Treiber, M.: Traffic flow dynamics: data, models and simulation. No. Book, Whole)(Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
26. Li, Q., Jia, X., Wang, S., Yan, J.: Think2drive: Efficient reinforcement learning by thinking with latent world model for autonomous driving (in carla-v2). In: European Conference on Computer Vision (2024)
27. Li, T., Qiu, Y., Wu, Z., Lindström, C., Su, P., Nießner, M., Li, H.: Mtgs: Multi-traversal gaussian splatting. arXiv preprint arXiv:2503.12552 (2025)
28. Li, Z., Li, K., Wang, S., Lan, S., Yu, Z., Ji, Y., Li, Z., Zhu, Z., Kautz, J., Wu, Z., et al.: Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. arXiv preprint arXiv:2406.06978 (2024)
29. Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Yu, Q., Dai, J.: Bevformer: learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
30. Liao, B., Chen, S., Yin, H., Jiang, B., Wang, C., Yan, S., Zhang, X., Li, X., Zhang, Y., Zhang, Q., et al.: Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. arXiv preprint arXiv:2411.15139 (2024)
31. Lu, H., Xu, T., Zheng, W., Zhang, Y., Zhan, W., Du, D., Tomizuka, M., Keutzer, K., Chen, Y.: Drivingrecon: Large 4d gaussian reconstruction model for autonomous driving. arXiv preprint arXiv:2412.09043 (2024)
32. Lu, Y., Yao, Y., Tu, J., Shao, J., Ma, Y., Zhu, X.: Can lvlms obtain a driver's license? a benchmark towards reliable agi for autonomous driving. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025)
33. Lu, Z., Lu, B., Wang, W., Wang, F.: Differentiable nms via sinkhorn matching for end-to-end fabric defect detection. arXiv preprint arXiv:2505.07040 (2025)
34. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
35. Ni, C., Zhao, G., Wang, X., Zhu, Z., Qin, W., Huang, G., Liu, C., Chen, Y., Wang, Y., Zhang, X., et al.: Recondreamer: Crafting world models for driving scene reconstruction via online restoration. arXiv preprint arXiv:2411.19548 (2024)

36. Ouyang, R., Li, H., Zhang, Z., Wang, X., Zhu, Z., Huang, G., Wang, X.: Motion-r1: Chain-of-thought reasoning and reinforcement learning for human motion generation. arXiv preprint arXiv:2506.10353 (2025)
37. Rashidinejad, P., Zhu, B., Ma, C., Jiao, J., Russell, S.: Bridging offline reinforcement learning and imitation learning: A tale of pessimism. Advances in Neural Information Processing Systems (2021)
38. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2022)
39. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al.: Mastering atari, go, chess and shogi by planning with a learned model. Nature (2020)
40. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
41. Shalev-Shwartz, S., Shammah, S., Shashua, A.: Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295 (2016)
42. Shao, Z., Wangjia2024bench2drive, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al.: Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300 (2024)
43. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. nature (2016)
44. Sonmez, Y., Krasowski, H., Arcak, M.: Learning to drive by imitating surrounding vehicles. arXiv preprint arXiv:2503.05997 (2025)
45. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2020)
46. Sun, W., Lin, X., Shi, Y., Zhang, C., Wu, H., Zheng, S.: Sparsedrive: End-to-end autonomous driving via sparse scene representation. arXiv preprint arXiv:2405.19620 (2024)
47. Treiber, M., Kesting, A.: Traffic flow dynamics. Springer (2013)
48. Wang, X., Zhu, Z., Huang, G., Chen, X., Zhu, J., Lu, J.: Drivedreamer: Towards real-world-driven world models for autonomous driving. arXiv preprint arXiv:2309.09777 (2023)
49. Weng, X., Ivanovic, B., Wang, Y., Wang, Y., Pavone, M.: Para-drive: Parallelized architecture for real-time autonomous driving. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2024)
50. Wu, Z., Liu, T., Luo, L., Zhong, Z., Chen, J., Xiao, H., Hou, C., Lou, H., Chen, Y., Yang, R., et al.: Mars: An instance-aware, modular and realistic simulator for autonomous driving. In: CAAI International Conference on Artificial Intelligence (2023)
51. Xu, C., Petiushko, A., Zhao, D., Li, B.: Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025)
52. Yan, S., Wang, Y., Zhao, K., Shi, P., Zhao, Z., Zhang, Y., Li, J.: Hemora: Unsupervised heuristic consensus sampling for robust point cloud registration. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2025)

53. Yan, Y., Lin, H., Zhou, C., Wang, W., Sun, H., Zhan, K., Lang, X., Zhou, X., Peng, S.: Street gaussians for modeling dynamic urban scenes. arXiv preprint arXiv:2401.01339 (2024)
54. Yang, C., Chen, Y., Tian, H., Tao, C., Zhu, X., Zhang, Z., Huang, G., Li, H., Qiao, Y., Lu, L., et al.: Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2023)
55. Yang, J., Ivanovic, B., Litany, O., Weng, X., Kim, S.W., Li, B., Che, T., Xu, D., Fidler, S., Pavone, M., et al.: Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. arXiv preprint arXiv:2311.02077 (2023)
56. Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.C., Yang, A.J., Urtasun, R.: Unisim: A neural closed-loop sensor simulator. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2023)
57. Zeng, S., Chang, X., Xie, M., Liu, X., Bai, Y., Pan, Z., Xu, M., Wei, X.: Futuresightdrive: Thinking visually with spatio-temporal cot for autonomous driving. arXiv preprint arXiv:2505.17685 (2025)
58. Zhang, D., Liang, J., Guo, K., Lu, S., Wang, Q., Xiong, R., Miao, Z., Wang, Y.: Carplanner: Consistent auto-regressive trajectory planning for large-scale reinforcement learning in autonomous driving. In: Proceedings of the Computer Vision and Pattern Recognition Conference (2025)
59. Zhang, H., Wang, Z., Wu, Z., Jiang, Y.G.: Diffusionad: Norm-guided one-step denoising diffusion for anomaly detection. arXiv preprint arXiv:2303.08730 (2023)
60. Zhang, J., Zhang, W., Tan, C., Li, X., Sun, Q.: Yolo-ppa based efficient traffic sign detection for cruise control in autonomous driving. arXiv preprint arXiv:2409.03320 (2024)
61. Zhao, G., Ni, C., Wang, X., Zhu, Z., Huang, G., Chen, X., Wang, B., Zhang, Y., Mei, W., Wang, X.: Drivedreamer4d: World models are effective data machines for 4d driving scene representation. arXiv preprint arXiv:2410.13571 (2024)
62. Zhao, G., Wang, X., Ni, C., Zhu, Z., Qin, W., Huang, G., Wang, X.: Recondreamer++: Harmonizing generative and reconstructive models for driving scene representation. arXiv preprint arXiv:2503.18438 (2025)
63. Zhao, G., Wang, X., Zhu, Z., Chen, X., Huang, G., Bao, X., Wang, X.: Drivedreamer-2: Llm-enhanced world models for diverse driving video generation. arXiv preprint arXiv:2403.06845 (2024)
64. Zheng, W., Song, R., Guo, X., Zhang, C., Chen, L.: Genad: Generative end-to-end autonomous driving. In: European Conference on Computer Vision (2024)
65. Zheng, Y., Xia, Z., Zhang, Q., Zhang, T., Lu, B., Huo, X., Han, C., Li, Y., Yu, M., Jin, B., et al.: Preliminary investigation into data scaling laws for imitation learning-based end-to-end autonomous driving. arXiv preprint arXiv:2412.02689 (2024)
66. Zhou, H., Lin, L., Wang, J., Lu, Y., Bai, D., Liu, B., Wang, Y., Geiger, A., Liao, Y.: Hugsim: A real-time, photo-realistic and closed-loop simulator for autonomous driving. arXiv preprint arXiv:2412.01718 (2024)
67. Zhou, L., Song, Y., Gao, Y., Yu, Z., Sodamin, M., Liu, H., Ma, L., Liu, L., Liu, H., Liu, Y., et al.: Garchingsim: An autonomous driving simulator with photorealistic scenes and minimalist workflow. In: 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC) (2023)
68. Zhu, J., Liu, J., Xu, T., Yuan, S., Zhang, Z., Jiang, H., Gu, H., Zhou, R., Liu, S.: Optical wafer defect inspection at the 10 nm technology node and beyond. International Journal of Extreme Manufacturing (2022)