R^{2}DRIVE: REINFORCEMENT-LEARNED REACTIVE AGENTS IN A CLOSED-LOOP 3D DRIVING SIMULATOR

Anonymous authors
Paper under double-blind review

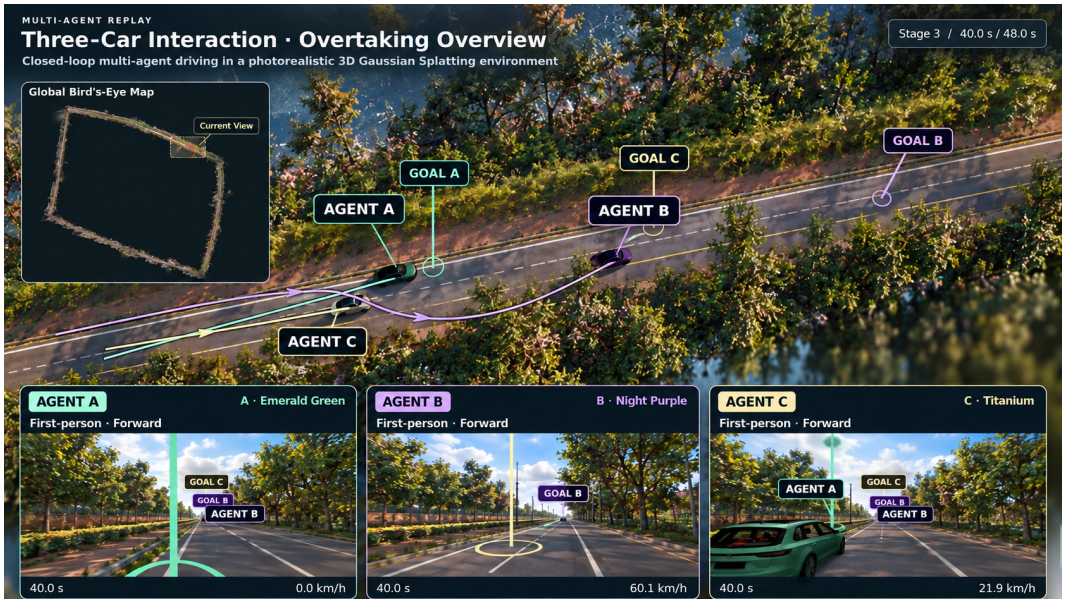


Figure 1: **R^{2}Drive overview.** An interactive circuit reconstructed from real-world scans, showing vehicles A/B/C and their goals (elevated view) with the corresponding first-person views. We host three complete circuits totalling more than 2 km of drivable roadway, all built so that the agent dynamics and simulation environment connect scene-level motion to local observations. All circuits are hosted online for interactive browser-based exploration, and Fig. 6 (Appendix C) shows what that looks like in practice; the reconstruction pipeline is detailed in Appendix B.

ABSTRACT

We present R^{2}Drive, a closed-loop multi-agent dynamics framework for sensor-level autonomous driving simulation with reactive surrounding vehicles. R^{2}Drive couples style-specific RL traffic policies with a 3D Gaussian simulator for adaptive behavior and closed-loop planning. The simulator preserves geometric fidelity (34.6 dB PSNR) while supporting distinct safety- and efficiency-oriented policy variants. Each reconstructed circuit is playable in a browser at 12–30 FPS over 1.5–2.0M Gaussians, enabling manual driving, shared-policy multi-agent replay, or an external planner receiving rendered RGB observations. On Waymo, the adapted agents reach 4% collision and 86% goal achievement; on nuPlan, the safety variant transfers zero-shot to 7.6% collision and 98.8% goal achievement. These results bridge photorealistic reconstruction and programmable multi-agent dynamics for closed-loop planner validation.

1 INTRODUCTION

Autonomous driving validation demands simulation environments that reconcile photorealistic reconstruction with adaptive multi-agent interactions while preserving physical consistency. Current

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

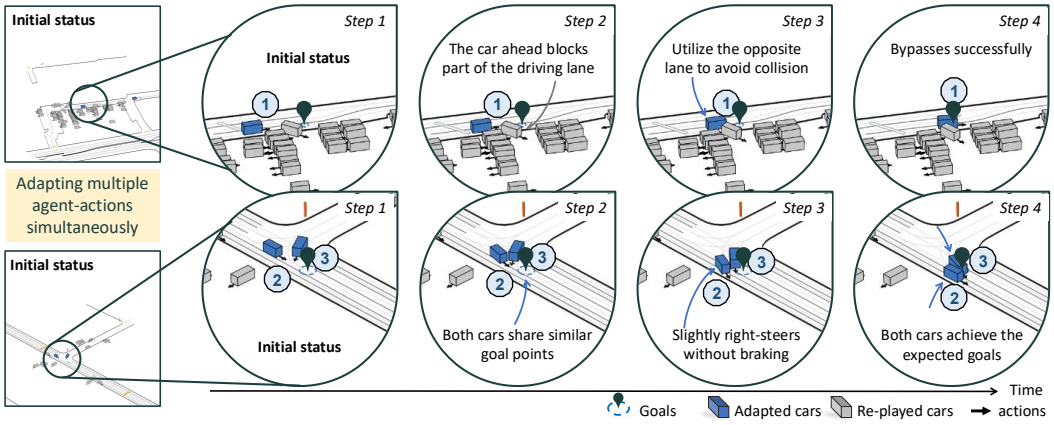


Figure 2: **Bird's-eye-view dynamics** (Waymo Open Sim Agents Challenge (Montali et al., 2023) scenarios). **Top:** at a parking-lot entrance, the adapted agent under the efficiency policy (ϕ^{eff}) invades the opposite lane to avoid a collision. **Bottom:** at a merging intersection, agents coordinate lane changes through adaptive acceleration. Goals are anchored to future waypoints sampled from the recorded trajectories at fixed intervals, preserving the original traffic layout.

solutions trade one against the other. Synthetic simulators such as CARLA (Dosovitskiy et al., 2017) lack real-world geometric fidelity because of procedural content generation; video diffusion models such as MagicDrive (Gao et al., 2024) and DriveArena (Yang et al., 2024b) sacrifice physical consistency for visual quality; and neural reconstruction systems such as StreetGaussians (Yan et al., 2024b) achieve strong 3D fidelity through decomposed static-dynamic representations, yet their fixed agent dynamics cannot simulate the alternative traffic scenarios that planner validation requires.

What is needed is all three at once: precise geometric alignment, physically plausible multi-agent interaction, and closed-loop compatibility with perception-planning architectures. Existing methods address only isolated aspects: policy-driven simulators such as LimSim (Wen et al., 2023) enable multi-agent behavior through cloning yet do not generalize, GPUDrive (Kazemkhani et al., 2024) favors speed over behavioral flexibility, and replay systems support planner integration without the behavioral diversity edge cases demand.

We present R^{2}Drive, a closed-loop multi-agent dynamics framework that resolves this tension through three technical innovations. First, a reinforcement learning adaptor enables customizable vehicle behavior, such as safety-prioritized (ϕ^{safe}) or efficiency-driven (ϕ^{eff}) policies, reaching a 4% collision rate on the Waymo Motion Dataset (Fig. 4) and transferring zero-shot to nuPlan, where ϕ^{safe} lowers the collision rate to 7.6% from GPUDrive's 9.1% (Table 1). Second, the R^{2}Drive simulator integrates 3D scene reconstruction with reactive agent dynamics, reaching 34.6 dB PSNR with a LiDAR RMSE of 1.802 (Table 2). Third, the closed-loop evaluation stack couples rendered-observation planning with planner post-training under reactive-agent dynamics, improving VAD and UniAD (Table 4) and, with our adapted agents, raising the zero-shot nuPlan goal-achievement rate to 98.8% (Table 1).

Across diverse urban environments, R^{2}Drive thus improves collision avoidance and goal achievement while preserving geometric fidelity, enabling planner validation against adaptive traffic configurations.

Sec. 3 presents the R^{2}Drive framework and Sec. 4 its concrete parameterization, followed by experiments in Sec. 5.

2 Related Work

Traffic Flow Simulation. Existing driving simulators face a realism–physics trade-off: video-based approaches such as DriveArena (Yang et al., 2024b) generate photorealistic scenes through latent diffusion but lack LiDAR consistency and collision constraints. Synthetic platforms such as

CARLA (Dosovitskiy et al., 2017) and MetaDrive (Li et al., 2023) enforce precise agent control through predefined road networks yet struggle to replicate real-world elevation profiles. Recent attempts such as OASim (Yan et al., 2024a) model 2D-plane interactions but fail to capture 3D phenomena like banked curves, while GPU-accelerated systems (Kazemkhani et al., 2024) prioritize rendering speed over physical accuracy. Behavior cloning methods including LimSim (Wen et al., 2023) reproduce recorded traffic patterns but cannot adapt to novel scenarios, and interaction-focused frameworks such as InterSim (Sun et al., 2022) emphasize historical trajectory analysis over future state prediction.

Closed-Loop Street Simulation and Agent Policies. 3D simulation environments differ mainly in scene representation: PVG (Chen et al., 2026) uses a dynamic Gaussian representation, while OmniRe (Chen et al., 2025) and StreetGaussians (Yan et al., 2024b) use Gaussian-based representations for photorealistic reconstruction. StreetCrafter (Yan et al., 2025) and ReconDreamer (Ni et al., 2024) support dynamic scene editing but not responsive agent dynamics, and planner architectures such as UniAD (Hu et al., 2023), VAD (Jiang et al., 2023), and RAD (Gao et al., 2025) are evaluated in static environments. On the policy side, CleanRL (Huang et al., 2022) provides standardized PPO (Schulman et al., 2017) implementations but targets single agents, DPPO (Ren et al., 2025) combines diffusion models with policy gradients for smoother action sampling, and GRPO (Shao et al., 2024) replaces the learned value baseline with a group-normalized advantage estimate. R^{2}Drive couples a Gaussian-based renderer with reactive policies using a shared PPO architecture; different behavior styles correspond to policy variants trained with style-specific preference-modulated rewards, while agents using the same variant share weights during multi-agent rollouts. We compare against GRPO in Sec. 5.3.

3 METHODOLOGY

3.1 R^{2}DRIVE: CLOSED-LOOP MULTI-AGENT DYNAMICS ON STREETS

The framework unifies static infrastructure and dynamic agents through two complementary representations. Static components $\mathbf{L}$ provide persistent geometric context, while dynamic agents $\mathbf{D}(t) = \{\mathbf{d}_1, \ldots, \mathbf{d}_k\}$ capture temporal traffic states in world coordinates. The static polyline ensemble maintains both invariant and time-varying elements, i.e., $\mathbf{L}(t) = \mathbf{L}_{\text{static}} \cup \mathbf{L}_{\text{dynamic}}(t)$. Dynamic agent evolution then follows $\mathbf{D}(t + \Delta t) = f\left(\mathbf{D}(t), \mathbf{L}, \Delta t\right)$, where $f(\cdot)$ computes next-step agent behaviors based on current infrastructure interactions and neighboring entity motions. This evolution operator may be instantiated as recorded real-world trajectories, behavioral policies, or parametric prediction models.

The representation ensures persistent road structure through $\mathbf{L}_{\text{static}}$ while adapting to traffic control changes via $\mathbf{L}_{\text{dynamic}}(t)$. Agent dynamics remain physically constrained through world-grounded transformations that respect both infrastructure geometry and inter-agent relationships.

STATIC COMPONENT REPRESENTATION

Road infrastructure is represented by scene-specific 4D keypoints, where each keypoint $\mathbf{p}_i$ combines geometric coordinates with a unique identifier:

$$\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}, \qquad \mathbf{p}_i = [i, x_i, y_i, z_i]^\top. \tag{1}$$

These index-anchored coordinates form raw geometric primitives without semantic interpretation. Semantic polyline structures emerge through ordered index sequences in the keypoint collection:

$$\mathbf{L}_m = \{\omega_m, \mathbf{v}_m, \tau_m\}, \qquad \mathbf{v}_m = [i_k, \ldots, i_{k+l}] \subseteq \mathbb{Z}^+, \tag{2}$$

where ω_m denotes semantic class, *e.g.*, a lane boundary or traffic-light stop line, $\mathbf{v}_m$ specifies consecutive keypoint indices from $\mathbf{P}$, and τ_m optionally represents the temporal status of traffic lights associated with the polyline. Each index i_k in $\mathbf{v}_m$ references the complete 4D vector $\mathbf{p}_{i_k} = [i_k, x_{i_k}, y_{i_k}, z_{i_k}]^\top$, preserving both geometric and topological information. This representation separates atomic geometry storage in $\mathbf{P}$ from semantic interpretation in $\mathbf{L}_m$ while enabling dynamic behavior through τ_m state transitions.

Dynamic Component Modeling

Dynamic entities, including vehicles, pedestrians, and cyclists, are represented by oriented 3D bounding boxes with comprehensive state vectors. The complete dynamic ensemble is defined as

$$\mathbf{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_k\}, \qquad \mathbf{d}_j = [\mathbf{t}_j, \mathbf{s}_j, R_j, \mathbf{v}_j]^\top. \tag{3}$$

Each agent state vector contains $\mathbf{t}_j = [t_j^x, t_j^y, t_j^z]^\top \in \mathbb{R}^3$ (3D centroid position), $\mathbf{s}_j \in \mathbb{R}^3$ (height, width, and length), and $\mathbf{v}_j \in \mathbb{R}^3$ (velocity). The ensemble size k varies over time as agents enter and exit the scene. Velocity is estimated by finite differencing between consecutive positions, $\mathbf{v}_j(t) = (\mathbf{t}_j(t) - \mathbf{t}_j(t - \Delta t))\,/\Delta t$. Each agent is transformed by its centroid translation $\mathbf{t}_j$ together with a standard yaw rotation about the z-axis parameterized by θ_j. Thus, $\mathbf{D}(t)$ captures all agents' states at time t, preserving elevation differences through t_j^z while constraining motion to the horizontal plane via $\theta_j \in SO(2)$. The z-coordinate t_j^z remains approximately constant for each agent but varies across types such as vehicles and traffic signals.

3.2 Adapting Behavior by Reinforcement Learning

Our framework employs Proximal Policy Optimization (PPO) to adapt driving behaviors through customized reward functions, enabling flexible agent-specific behavior modeling. To ensure human-like behavioral priors and accelerate convergence, we initialize the policy with imitation learning on recorded expert trajectories. Unlike diffusion-based approaches such as VBD (Huang et al., 2024) that generate multi-agent dynamics through denoising processes, our method applies PPO at each timestep for target agents with individualized reward functions. This per-agent temporal application ensures flexible behavior adaptation while maintaining real-time responsiveness.

Architecture

The policy is implemented as a feed-forward late-fusion actor-critic network operating on the unified representation from Sec. 3.1. For each target agent j, the network ingests three complementary inputs: (1) static road components $\mathbf{L}_m$ defined in (2), (2) neighboring dynamic agent states $\mathbf{d}_i \in \mathbf{D}$ from (3), and (3) the target agent's own state $\mathbf{d}_j$ together with its navigation goal $\mathbf{g}_j \in \mathbb{R}^3$. Lightweight MLP encoders map road and agent features into latent embeddings, and max pooling aggregates the variable-size road and neighboring-agent sets into compact summaries $\mathbf{E}_{\text{road}}$ and $\mathbf{E}_{\text{obj}}^j$.

These pooled embeddings are fused with the target agent's state and goal, then processed by a shared multi-layer perceptron to produce a joint representation:

$$\mathbf{h}_j = \text{MLP}_{\text{shared}}\left([\mathbf{E}_{\text{obj}}^j; \mathbf{E}_{\text{road}}; \mathbf{d}_j; \mathbf{g}_j]\right), \tag{4}$$

where $\mathbf{E}_{\text{obj}}^j$ denotes pooled embeddings of agents other than j. The shared representation is passed to actor and critic heads with temporal smoothing, producing stable control outputs while maintaining motion continuity through delta-pose constraints. The concrete architecture of the shared network is specified in Sec. 4.

Training

The discrete action space comprises 7 acceleration/braking levels and 13 steering factors, yielding 91 control combinations. Training uses PPO with parallel rollout collection and minibatch updates over 128 worlds, each containing one scene and up to 128 controlled agents. The optimization pipeline alternates among temporal rollouts that collect interaction data, generalized advantage estimation (Schulman et al., 2017) to compute advantages, and policy-network updates. In our implementation, training takes about two days on 8×A100 GPUs, and all reported closed-loop experiments use seed 42.

Reward Design

To optimize the policy parameters θ, we define a composite reward function R that balances navigation efficiency and safety constraints. Following the standard GPUDrive formulation (Kazemkhani et al., 2024), the instantaneous reward for agent j is

$$R_j = w_c \cdot \mathbb{I}_{\text{coll}} + w_g \cdot \mathbb{I}_{\text{goal}} + w_o \cdot \mathbb{I}_{\text{off-road}}, \tag{5}$$



Figure 3: **Interactive 4D rollout** (Waymo multi-view camera setup). From a recorded scene, the ego vehicle is displaced 0.5 m to the left and surrounding agents under the safety-prioritized policy (ϕ^{safe}) react over the next 1.1 s, with each timestep re-rendered from the evolving 3D Gaussian scene. **Top:** original recorded rollout. **Bottom:** reactive rollout, where nearby agents respond to the shifted ego trajectory while rendering stays temporally and geometrically consistent.

where $\mathbb{I}_{\{\cdot\}} \in \{0, 1\}$ are indicator functions for collision, goal achievement, and off-road events. The weight values adopted in all reported experiments are listed in Sec. 4. By fixing w_g and w_o while adjusting the sensitivity of the collision signal via ϕ, our model captures diverse driving temperaments, ranging from conservative safety-first behavior to efficiency-driven aggressive maneuvers. Policy optimization minimizes a composite loss that balances action selection, value prediction, and exploration:

$$\mathcal{L} = -\mathbb{E}_t \Big[\underbrace{\log \pi_\theta(a_t|s_t)\, \hat{A}_t}_{\text{policy}} - \alpha_1 \underbrace{\big(V_\theta(s_t) - V_{\text{target}}\big)^2}_{\text{value}} - \alpha_2 \underbrace{\mathcal{H}(\pi_\theta(\cdot|s_t))}_{\text{entropy}} \Big], \tag{6}$$

which is minimized, so the policy term is the negative log-likelihood weighted by the advantage estimate $\hat{A}_t$ obtained from generalized advantage estimation (Schulman et al., 2017), while the value and entropy terms enter with the opposite sign. Here α_1 weights the value loss and α_2 weights the entropy term. The behavior parameter ϕ_j modulates the collision-distance threshold during training, producing style-specific policy variants. In practice, the *safe* variant is trained with a more conservative distance threshold to prioritize collision avoidance, whereas the *efficient* variant is trained to maximize progress without overspeeding. This mechanism enables complex multi-agent simulation in which individual responses reflect the selected style-specific risk profile together with the global navigation goal, and the policies are optimized with Adam (Kingma & Ba, 2014).

3.3 R^{2}Drive Simulator: Reactive Multi-Agent Dynamics

The simulation extends base dynamics through agent-specific response models,

$$\mathbf{d}_j(t + \Delta t) = f_{\text{react}}\left(\mathbf{L}, \mathbf{D}(t), \Delta t; \phi_j\right), \tag{7}$$

where ϕ_j indexes the style-specific policy variant assigned to agent j. At each timestep, the observation is the tuple $\mathbf{O}(t) = \{\mathbf{L}, \mathbf{D}(t)\}$, which combines the static map context with the current dynamic agent states.

We employ 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) as an efficient and differentiable representation of dynamic 3D scenes. The R^{2}Drive simulator grounds agent dynamics in real-world geometry by projecting agents to the road surface and aligning the ego vehicle with the local road tangent, while representing the scene as a union of a static background $\mathcal{G}_s$ and dynamic objects $\{\mathcal{O}_k\}_{k=1}^{K}$. Dynamic-object Gaussians are transformed into world coordinates through differentiable rigid-body motion, and the scene parameters are optimized with the photometric rendering loss

$$\mathcal{L}_{\text{render}} = \sum_{t=1}^{T} \sum_{m=1}^{M} \|C_t^m - \hat{C}_t^m\|_2^2, \tag{8}$$

where C_t^m and $\hat{C}_t^m$ are the rendered and ground-truth images for camera m at timestep t.

The simulator updates ego-camera poses through the vehicle trajectory while optimizing dynamic-object poses with lightweight networks driven by rendering gradients, following Nerfstudio (Tancik et al., 2023). This compact formulation suffices here because the main contribution lies in coupling differentiable street-scale rendering with reactive multi-agent behavior, rather than in a new Gaussian parameterization or camera optimization scheme.

3.4 R^{2}DRIVE PLANNER: EGO-CENTRIC PLANNING

Ego-centric planning in R^{2}Drive can use a lightweight reference vision–action (VA) planner on rendered observations. It maps $\mathbf{X}(t) = \{\mathcal{I}_{\text{RGB}}(t), \mathbf{x}_t\}$ to $a_{\text{ego}}(t)$, where the 30-D $\mathbf{x}_t$ encodes ego motion, scan-line ground geometry, and goal-relative distance/direction; no vectorized BEV intermediate is used. Historical agent states $\mathbf{D}(t_{-H:0})$ initialize the simulator ($H = 25$, 2.5 s at 10Hz). Surrounding agents follow (7), while ego actions use a kinematic bicycle model, yielding $\mathbf{D}(t + \Delta t) = f_{\text{R}^2\text{Drive}}(\mathbf{D}(t), \pi_\theta(\mathbf{X}(t)))$. Details appear in Sec. 4.1 and Appendix E.

This cycle evaluates planning robustness under evolving traffic. Because the planner is grounded in rendered RGB plus compact state rather than a vectorized map, rendering artifacts and occlusion affect the full perception-to-action path. Preference-adapted agents (Sec. 3.2) test diverse behaviors, with per-step propagation summarized in Algorithm 1 of Appendix A.

In Sec. 5.3, the same closed-loop stack post-trains VAD (Jiang et al., 2023) and UniAD (Hu et al., 2023) under identical reactive-agent dynamics; only the ego planner is swapped.

4 IMPLEMENTATION

The concrete parameterization used for every result in Sec. 5 is summarized here; additional network and training details are provided in Appendix D, while the reconstruction and rendering pipeline is documented in Appendix B.

4.1 POLICY NETWORK AND EGO PLANNER

The shared multi-layer perceptron in (4) has two hidden layers of dimension 256, followed by a linear projection to $\mathbf{h}_j$, with two-layer actor and critic heads. The separate vision–action ego planner (Fig. 7, Appendix E) fuses a rendered RGB frame I_t with a 30-dimensional state vector x_t (ego motion, scan-line ground-profile features, and goal-relative distance/direction) into a 384-D representation. Its actor predicts discrete driving actions and its critic predicts a scalar value used only during training. The planner therefore remains exposed to rendering fidelity while sharing the reactive agents' PPO objective (Sec. 3.2), not their input modality. This planner drives the interactive setting in Fig. 1.

4.2 REWARD CONFIGURATION

The instantaneous reward in (5) uses $w_c = -0.5$ for the collision penalty, $w_g = +1.0$ for reaching the anchored future waypoint, and $w_o = -0.5$ for off-road violations, for every policy variant reported in Sec. 5.1. Behavioral variation across ϕ comes from the style-specific collision-distance threshold $d_{\text{min}}(\phi_j)$ used during training, not from re-tuning these reward weights.

4.3 CLOSED-LOOP SIMULATION PROCEDURE

The temporal propagation mechanism operates through the simulator's reactive dynamics engine, whose per-step procedure is given as Algorithm 1 in Appendix A. Fidelity of the resulting rollout is measured by the Jensen–Shannon divergence between recorded and reactive behaviors,

$$D_{\text{JS}}(P_{\text{recorded}} \| P_{\text{react}}), \tag{9}$$

where lower divergence indicates closer agreement with the recorded trajectories.

5 EXPERIMENTS

5.1 MULTI-AGENT DYNAMICS

Our framework enables heterogeneous driving behaviors through style-specific policy variants, contrasting with the fixed dynamics of GPUDrive (Kazemkhani et al., 2024). We evaluate multi-agent dynamics on scenarios from the Waymo Open Sim Agents Challenge (WOSAC) (Montali et al., 2023). On the Waymo Motion Dataset (Sun et al., 2020), R^{2}Drive runs several driving strategies

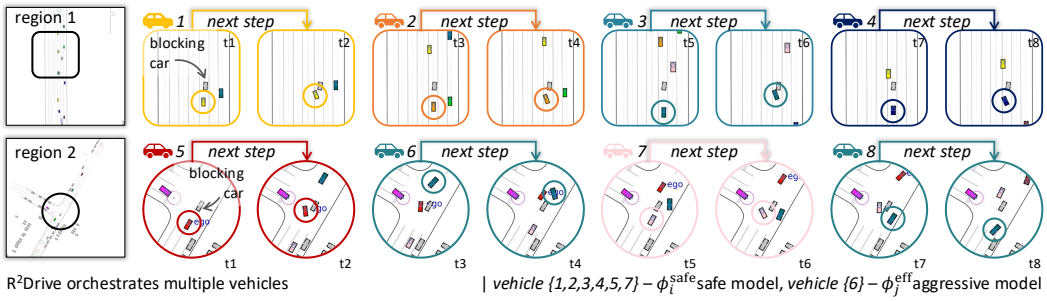


Figure 4: **Collision avoidance** (Waymo Open Sim Agents Challenge (Montali et al., 2023) scenario). A broken-down vehicle blocks the lane; agent 6, running the efficiency policy (ϕ^{eff}), temporarily invades the opposite lane to evade the obstacle while all other agents remain under the safe policy (ϕ^{safe}), sustaining traffic flow without a crash.

within a single scene and sustains collision-free interaction even when an efficiency-driven agent deviates from lane discipline (Fig. 4), reaching a 4% collision rate and an 86% goal-achievement rate on the WOSAC evaluation split, which uses a different scenario distribution and collision criterion than the nuPlan validation set of Table 1 and is therefore not directly comparable to it. This validates its capacity to simulate complex traffic interactions with mixed style-specific behaviors.

nuPlan Dataset

Cross-dataset evaluation on nuPlan (Caesar et al., 2021) demonstrates R^{2}Drive's generalization to unseen regions and street layouts. Table 1 compares policy configurations across 850 challenging scenarios featuring multi-agent congestion and unprotected turns.

Table 1: Multi-agent simulation results on the nuPlan validation set.

Method	Collision Rate ↓	Goal Rate ↑
R^{2}Drive (ϕ^{safe})	**7.6**	98.8
R^{2}Drive (ϕ^{eff})	8.2	**99.5**
GPUDrive (Kazemkhani et al., 2024)	9.1	96.0
LimSim (Wen et al., 2023)	12.3	93.5
DriveGAN (Kim et al., 2021)	15.8	90.1
Waymo Replay	21.7	85.3

Through zero-shot transfer from Waymo training, our safety-optimized configuration (ϕ^{safe}) lowers the collision rate from GPUDrive's 9.1% to 7.6% while improving the goal-achievement rate by 2.8 points (98.8% vs. 96.0%). The efficiency-oriented variant (ϕ^{eff}) reaches a 99.5% goal-achievement rate through strategic rule deviations, such as justified lane invasions, with only a modest increase in collision risk (8.2%). This validates R^{2}Drive's ability to balance safety and efficiency in dense heterogeneous traffic without scenario-specific fine-tuning. R^{2}Drive achieves strong safety–efficiency trade-offs through style-specific policy variants.

5.2 Simulator Validation

To simulate adjusted traffic behavior in a recorded scene, we optimize trajectories that are both efficient and safe, as shown in Fig. 3. Whereas the multi-agent results above use WOSAC scenes, the rendering results here use the Waymo multi-view camera setup: we validate the simulator's photorealistic rendering and geometric consistency on the Waymo Open Dataset (Sun et al., 2020), following OmniRe's experimental protocol (Chen et al., 2025) with three front-view cameras for training. Quantitative evaluation focuses on vehicle reconstruction quality across 1000 urban scenarios, excluding deformable objects. Metrics include PSNR↑ and SSIM↑ for both scene reconstruction and novel-view synthesis.

View Synthesis

The R^{2}Drive simulator achieves state-of-the-art performance with 34.63 dB PSNR for training-view scene reconstruction and 32.75 dB for novel-view synthesis, reported in the left and right blocks of Table 2, respectively. Vehicle-specific metrics also improve, with R^{2}Drive outperforming OmniRe by 0.34 dB PSNR on training views and 0.18 dB on novel views; the advantage is smaller under

Table 2: Reconstruction quality on the Waymo Open Dataset: training-view reconstruction (left block) and novel-view synthesis (right block). Each block reports full-render PSNR/SSIM, vehicle-only PSNR/SSIM, and LiDAR CD/RMSE; the last column marks whether dynamic agents are supported.

Method	Training-view reconstruction						Novel-view synthesis						Agents
	Full		Vehicle		LiDAR		Full		Vehicle		LiDAR		
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	CD↓	RMSE↓	PSNR↑	SSIM↑	PSNR↑	SSIM↑	CD↓	RMSE↓	
EmerNeRF (Yang et al., 2024a)	31.93	0.902	24.65	0.723	–	–	29.67	0.883	22.07	0.609	–	–	×
3DGS (Kerbl et al., 2023)	26.00	0.912	16.18	0.425	0.415	2.804	25.57	0.906	16.00	0.407	0.467	2.896	×
DeformGS (Yang et al., 2024c)	28.40	0.929	19.53	0.570	–	–	27.72	0.922	18.91	0.530	–	–	×
PVG (Chen et al., 2026)	32.37	0.937	25.02	0.787	–	–	30.19	0.919	22.28	0.679	–	–	×
StreetGS (Yan et al., 2024b)	29.08	0.936	27.73	0.880	0.274	2.199	28.54	0.928	26.71	0.846	0.286	2.228	×
OmniRe (Chen et al., 2025)	34.25	0.954	28.91	0.892	0.242	1.894	32.57	0.942	27.57	0.858	0.244	1.909	×
R^{2}Drive (ours)	**34.63**	**0.956**	**29.25**	**0.897**	**0.225**	**1.802**	**32.75**	**0.943**	**27.75**	**0.859**	**0.232**	**1.823**	✓

synthesis than under reconstruction but holds in both settings. This gain stems from joint pose optimization and multi-view consistency constraints during neural rendering. The synthesized views also preserve collision-free interactions between surrounding vehicles and the ego vehicle, as shown in the lower row of Fig. 3.

Tracking-Oriented Validation

Beyond photometric and geometric fidelity, we assess whether the simulated outputs remain useful for downstream perception tasks. Following a fixed multi-object tracking protocol on the nuScenes validation set, we compare PVG, StreetGS, OmniRe, and the R^{2}Drive simulator using AMOTA, AMOTP, Recall, and identity switches (IDS). As shown in Table 3, the R^{2}Drive simulator achieves the best tracking accuracy and the fewest identity switches among the compared simulators, indicating that its reactive rendering remains temporally coherent enough for planner-facing perception stacks.

Table 3: Multi-object tracking on the nuScenes validation set.

Simulator	AMOTA↑	AMOTP↓	Recall↑	IDS↓
PVG (Chen et al., 2026)	0.294	1.699	0.308	990
StreetGS (Yan et al., 2024b)	0.302	1.531	0.396	959
OmniRe (Chen et al., 2025)	0.307	1.523	0.409	955
R^{2}Drive (ours)	**0.351**	**1.327**	**0.453**	**920**

Geometric Reconstruction

The R^{2}Drive simulator demonstrates strong geometric fidelity in Table 2, with a Chamfer Distance of 0.225 on training frames, representing a 7.0% improvement over OmniRe (Chen et al., 2025). RMSE follows the same direction at 1.802 versus OmniRe's 1.894, a smaller 4.9% gain because the residual error concentrates in thin structures such as poles and lane markings rather than in the vehicle surfaces that dominate the Chamfer term. Joint optimization of ego and object poses aligns the reconstruction with LiDAR ground truth, particularly on vehicle roofs and wheel arches.

5.3 Closed-Loop Planning

We evaluate separately post-trained VAD (Jiang et al., 2023) and UniAD (Hu et al., 2023) on the Bench2Drive validation set (Jia et al., 2024) under the same R^{2}Drive reactive-agent dynamics. As shown in Table 4, our RL post-training improves both planners over their Bench2Drive baselines. Removing imitation-learning initialization or replacing PPO with GRPO consistently reduces the gain, indicating

Table 4: Closed-loop planning on the Bench2Drive validation set (Jia et al., 2024) for VAD (Jiang et al., 2023) and UniAD (Hu et al., 2023) under identical reactive-agent dynamics. Metrics are Driving Score (DS), Success Rate (SR), Efficiency (Eff), and Comfort (Com).

Setting	VAD				UniAD			
	DS↑	SR↑	Eff	Com	DS↑	SR↑	Eff	Com
Bench2Drive	42.4	15.0	157.9	46.0	45.8	16.4	129.2	43.6
- with our RL	**52.8**	**24.7**	**178.6**	**51.4**	**56.3**	**25.8**	**149.5**	**49.2**
- w/o IL	47.8	18.9	166.8	48.1	50.5	19.8	137.6	45.4
- with GRPO	49.1	20.4	170.3	49.0	52.2	21.3	141.8	46.6

that the improvement comes from the post-training recipe rather than from changing the underlying planner architecture.

The framework runs at 10Hz over 24-frame sequences (2.4 s) on nuScenes (Caesar et al., 2020), with dynamic adaptation applied at each frame. Compared with recorded dynamics, R^{2}Drive raises traffic-flow efficiency by 12% while reducing collision risk by 32%; the trajectories and velocity profiles behind these aggregates are shown in Fig. 5, whose panels (b)–(d) report per-step ego acceleration, velocity, and travel distance.

6 Conclusion

R^{2}Drive combines three components: style-specific RL traffic policies, 3D Gaussian simulation for photorealistic closed-loop rollouts, and a planning stack for rendered-observation planning and planner post-training under reactive-agent dynamics. Together they preserve strong reconstruction quality while enabling customizable traffic interactions.

The system is meant to be driven rather than only read about: each released circuit runs as a live environment in which a human can take the ego vehicle while reactive agents respond, self-play rollouts expose two or three agents acting from one shared policy variant, and an external planner can attach through the control API and receive the same rendered observations a browser user sees. Having all three share one scene parameterization is what keeps them consistent, and it is the property we take to be the contribution.

A current limitation is scale: the circuits cover just over 2 km of roadway and a handful of interaction scenarios, and styles are specified through reward modulation rather than learned from demonstration. Extending the capture area and making the style space learnable from human driving are the natural next steps.

Reproducibility statement

Reproduction details for the model and training setup, rewards, datasets, evaluation, random seed, closed-loop simulation, reconstruction and rendering, policy implementation, and VA planner are provided in Sections 3–5 and Appendices A–E.

AI use statement

Generative AI tools were used to assist with drafting and polishing portions of the manuscript, including language editing, manuscript organization, and LaTeX formatting. All AI-assisted text was reviewed and revised by the authors. The authors verified the technical claims, experimental setup, implementations, numerical results, and conclusions, and take full responsibility for the final content of this work.

REFERENCES

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuPlan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *CVPR ADP3 Workshop*, 2021.

Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic vibration Gaussian: Dynamic urban scene reconstruction and real-time rendering. *International Journal of Computer Vision*, 134(3):83, 2026.

Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, Li Song, and Yue Wang. Omnire: Omni urban scene reconstruction. In *The Thirteenth International Conference on Learning Representations*, 2025.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

Hao Gao, Shaoyu Chen, Bo Jiang, Bencheng Liao, Yiang Shi, Xiaoyang Guo, Yuechuan Pu, Haoran Yin, Xiangyu Li, Xinbang Zhang, Ying Zhang, Wenyu Liu, Qian Zhang, and Xinggang Wang. Rad: Training an end-to-end driving policy via large-scale 3dgs-based reinforcement learning. *arXiv preprint arXiv:2502.13144*, 2025.

Ruiyuan Gao, Kai Chen, Enze Xie, Lanqing Hong, Zhenguo Li, Dit-Yan Yeung, and Qiang Xu. Magicdrive: Street view generation with diverse 3d geometry control. In *International Conference on Learning Representations (ICLR)*, 2024.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Zhiyu Huang, Zixu Zhang, Ameya Vaidya, Yuxiao Chen, Chen Lv, and Jaime Fernández Fisac. Versatile behavior diffusion for generalized traffic agent simulation. *arXiv preprint arXiv:2404.02524*, 2024.

Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. VAD: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Saman Kazemkhani, Aarav Pandya, Daphne Cornelisse, Brennan Shacklett, and Eugene Vinitsky. GPUDrive: Data-driven, multi-agent driving simulation at 1 million FPS. *arXiv preprint arXiv:2408.01584*, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/3d_gaussian_splatting_high.pdf.

Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. DriveGAN: Towards a controllable high-quality neural simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5820–5829, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. MetaDrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3461–3475, 2023. doi: 10.1109/TPAMI.2022.3190471.

Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nicholas Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, Brandyn White, and Dragomir Anguelov. The Waymo open sim agents challenge. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Chaojun Ni, Guosheng Zhao, Xiaofeng Wang, Zheng Zhu, Wenkang Qin, Guan Huang, Chen Liu, Yuyin Chen, Yida Wang, Xueyang Zhang, et al. Recondreamer: Crafting world models for driving scene reconstruction via online restoration. *arXiv preprint arXiv:2411.19548*, 2024.

Yanjun Pan, Qin Lin, Het Shah, and John M Dolan. Safe planning for self-driving via adaptive constrained ilqr. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2377–2383. IEEE, 2020.

Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations*, 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, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2446–2454, 2020.

Qiao Sun, Xin Huang, Brian C. Williams, and Hang Zhao. InterSim: Interactive traffic simulation via explicit relation modeling. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11416–11423. IEEE, 2022.

Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

Licheng Wen, Daocheng Fu, Song Mao, Pinlong Cai, Min Dou, Yikang Li, and Yu Qiao. LimSim: A long-term interactive multi-scenario traffic simulator. In *IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1255–1262, 2023.

Guohang Yan, Jiahao Pi, Jianfei Guo, Zhaotong Luo, Min Dou, Nianchen Deng, Qiusheng Huang, Daocheng Fu, Licheng Wen, Pinlong Cai, Xing Gao, Xinyu Cai, Bo Zhang, Xuemeng Yang, Yeqi Bai, Hongbin Zhou, and Botian Shi. Oasim: an open and adaptive simulator based on neural rendering for autonomous driving, 2024a.

Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street Gaussians: Modeling dynamic urban scenes with gaussian splatting. In *ECCV*, 2024b.

Yunzhi Yan, Zhen Xu, Haotong Lin, Haian Jin, Haoyu Guo, Yida Wang, Kun Zhan, Xianpeng Lang, Hujun Bao, Xiaowei Zhou, and Sida Peng. Streetcrafter: Street view synthesis with controllable video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 822–832, 2025.

Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, and Yue Wang. EmerNeRF: Emergent spatial-temporal scene decomposition via self-supervision. In *The Twelfth International Conference on Learning Representations*, 2024a.

Xuemeng Yang, Licheng Wen, Yukai Ma, Jianbiao Mei, Xin Li, Tiantian Wei, Wenjie Lei, Daocheng Fu, Pinlong Cai, Min Dou, Botian Shi, Liang He, Yong Liu, and Yu Qiao. Drivearena: A closed-loop generative simulation platform for autonomous driving. *arXiv preprint arXiv:2408.00415*, 2024b.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20331–20341, 2024c.

APPENDIX

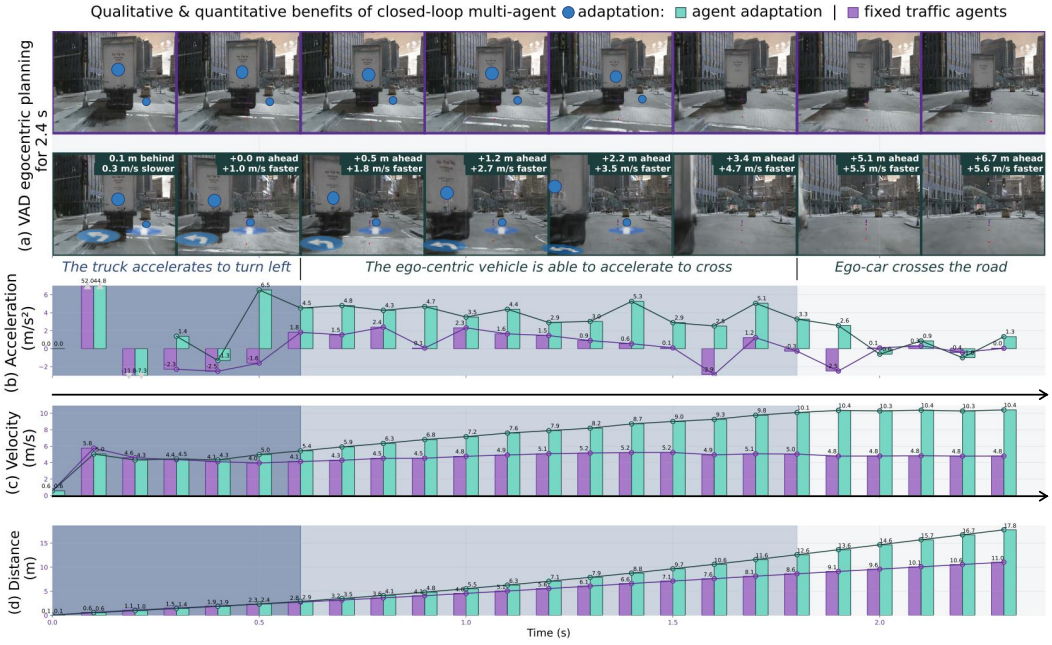


Figure 5: **Closed-loop planning with R^{2}Drive.** (a) VAD (Jiang et al., 2023), after post-training under R^{2}Drive reactive-agent dynamics, is evaluated on nuScenes (Caesar et al., 2020). Compared with fixed recorded dynamics (top), reactive-agent post-training (bottom) exposes VAD to agents that respond to ego intentions: social interactions over 0–0.6 s lead to adaptive ego acceleration (b), yielding higher ego velocity (c) and greater travel distance (d) at safe inter-vehicle margins.

A CLOSED-LOOP SIMULATION PROCEDURE

Algorithm 1 expands the temporal propagation of Sec. 4.3: each step renders observations from the evolving Gaussian scene, advances the ego vehicle through the planner and a kinematic bicycle model, lets the remaining agents respond through their reactive models, and finally projects the joint state back onto the drivable region.

Algorithm 1 Closed-loop simulation in R^{2}Drive.

Input: recorded initial states $\mathbf{D}(0)$, static context $\mathbf{L}$, reconstructed scene Gaussians $\mathcal{G}$, behavior parameters $\{\phi_j\}$, planner π_{planner}, horizon T
Output: closed-loop rollout $\{\mathbf{D}(t)\}_{t=1}^{T}$

for $t = 0$ **to** $T-1$ **do**
  $\mathbf{X}(t) \leftarrow \{\mathcal{I}_{\text{RGB}}(t), \mathbf{x}_t\}$ from $(\mathcal{G}, \mathbf{D}(t))$ ▷ *RGB + 30-D ego/geometric/navigation state*
  $a_{\text{ego}}(t) \leftarrow \pi_{\text{planner}}(\mathbf{X}(t))$ ▷ *ego planning*
  $\mathbf{d}'_{\text{ego}}(t+1) \leftarrow f_{\text{bicycle}}\left(\mathbf{d}_{\text{ego}}(t), a_{\text{ego}}(t)\right)$ ▷ *kinematic bicycle update*
  $\tilde{\mathbf{D}}(t) \leftarrow (\mathbf{D}(t) \setminus \{\mathbf{d}_{\text{ego}}(t)\}) \cup \{\mathbf{d}'_{\text{ego}}(t+1)\}$ ▷ *expose planned ego motion*
  $\mathbf{d}'_j(t+1) \leftarrow f_{\text{react}}\left(\mathbf{L}, \tilde{\mathbf{D}}(t), \Delta t; \phi_j\right)$ **for all** $j \neq$ ego ▷ *reactive agents, (7)*
  $\mathbf{D}(t+1) \leftarrow \text{constrain}_{\text{iLQR}}\left(\{\mathbf{d}'_{\text{ego}}(t+1)\} \cup \bigcup_{j \neq \text{ego}} \{\mathbf{d}'_j(t+1)\}, \mathbf{L}\right)$ ▷ *safety projection (Pan et al., 2020)*
end for

The safety projection in the final step is solved with an iLQR-based constrained optimizer (Pan et al., 2020), which enforces infrastructure compliance against the static component representations of (1).

Behavioral fidelity is reported as the Jensen–Shannon divergence between the recorded and reactive behavior distributions, $D_{\text{JS}}(P_{\text{recorded}} \| P_{\text{react}})$, defined in Sec. 4.3.

B Circuit Construction Pipeline

Sec. 5 reports rendering quality on individual urban scenes, but the circuits released with this paper cover more than 2 km of continuous roadway, which makes pose estimation the dominant source of error. We therefore reconstruct each circuit as a single globally consistent model rather than as a collection of independent scenes. This appendix documents that pipeline, and Fig. 6 in Appendix C shows the result as a user sees it: the released circuits are not static reconstructions but interactive environments, in which a policy can be driven directly from the browser and the surrounding agents respond to it in real time.

B.1 Egocentric Trajectory and Camera Pose Optimization

Each circuit is captured with a vehicle-mounted multi-camera rig whose inertial measurements give an egocentric trajectory prior. Camera poses relative to that trajectory are initialized from a calibrated sparse structure-from-motion stage (metric depth, frontend/backend configuration with a backend window size of 5) and aligned to the dataset coordinate frame by a similarity transform. Frames that remain unregistered are filled in by spherical linear interpolation along the trajectory and flagged explicitly, so that interpolated poses are never mistaken for measured ones during refinement.

Pose refinement proceeds in two stages. Position priors from the inertial trajectory are first enforced through bundle adjustment with a tight position prior standard deviation of 0.001. The refined result then has to pass four geometric gates before it is accepted: alignment error ≤ 0.1, median and 95th-percentile reprojection errors of 2 and 8 pixels, at least 24 reliable trajectories, and rotational consistency within 3° and 8° at the same percentiles. Scenes that fail these gates fall back to fixed-pose re-triangulation rather than entering the joint stage with corrupted geometry.

B.2 Chunked Reconstruction and Boundary Consistency

Because a multi-kilometre capture cannot be optimized in one pass, we partition each circuit into chunks of roughly 5–10 segments and refine them in two alternating passes: intra-chunk optimization, which fits poses and scene parameters within each chunk, and inter-chunk optimization, which aligns neighbouring chunks while holding their interiors fixed. This graduation matters because a pose error of a few centimetres at a chunk boundary disrupts Gaussian geometry on both sides of the seam, and the effect compounds along the circuit.

Gaussians are initialized from a feed-forward scene predictor run with a window of 8 frames; the resulting depth maps are calibrated against co-visible sparse geometry, and additional Gaussian seeds are drawn from keyframes with high co-visibility. Training uses a splatting-based differentiable renderer with seed 42 and 12,579 steps per scene, obtained by scaling the reference budget of 2,000 steps by the ratio of the 522 training frames per scene to the reference count of 83 ($2{,}000 \times 522/83 \approx 12{,}579$); across the 7 scenes retained after the geometric gates of Sec. B this yields the 3,654 training views used in the joint stage, under the composite loss

$$\mathcal{L} = 0.8\,\mathcal{L}_{\text{rgb}} + 0.2\,\mathcal{L}_{\text{ssim}} + 0.04\,\mathcal{L}_{\text{depth}} + 0.1\,\mathcal{L}_{\text{sky}} + 0.02\,\mathcal{L}_{\text{coverage}}, \tag{10}$$

with opacity pruning at threshold 0.003 and no densification during this stage.

Individually reconstructed chunks are then merged and jointly optimized for a further 7,308 steps over all 3,654 training views together with 3,654 samples drawn from 96 frames on either side of each boundary. Densification is disabled during joint optimization so that the boundary samples correct pose and opacity consistency without spawning spurious geometry along the seams. This final stage is what makes the released circuits continuous rather than merely adjacent.

B.3 Real-Time Rendering

At play time the reconstructed circuit is streamed and rendered in real time with a GPU-accelerated splatting pipeline, which serves both the Gaussian radiance field and the mesh representation so that

a viewer can drive the circuit interactively while the agent dynamics and simulation environment respond. Rendering and simulation share the same scene parameterization, so the visual observation a planner receives during closed-loop evaluation is produced by the same model that was reconstructed from the real-world capture. The captures in Fig. 6 are taken from this pipeline: the browser frontend sustains 12–30 FPS at 1.5–2.0M drawn Gaussians, which is fast enough for a human to drive a car by hand while the reactive agents act on the same scene. The three circuits share one control interface, so a user can switch between driving manually, replaying a self-play rollout, and stepping through a staged interaction without leaving the session, and an external planner can attach to the same session through the API in Fig. 6c and receive the rendered frame as its observation.

C User Manual: The Hosted Circuits

Everything reported above is produced by one system that a reader can also operate. This section documents how the released circuits are driven, using frames captured from the hosted frontend; the interface is identical across the three circuits. The environments require no installation or local GPU: a circuit is loaded in a browser, and the session then exposes reinforcement-learning replay, staged multi-agent interaction, and manual driving through one shared control surface.

Training replay. The replay panel is the main evidence that the agents are learned rather than scripted. Selecting a checkpoint steps the same scene forward under the frozen policy, and the per-agent action overlays show the discrete control the network selects at each frame; because both agents share one set of weights (Fig. 6a), any asymmetry in their behavior comes from their initial states and their observations of each other. The checkpoint markers let a user compare the same scene across training, which is what the collision-rate and goal-achievement figures of Sec. 5.1 aggregate.

Staged interaction. The three-car circuit advances through ordered stages, each placing the agents at a designed conflict point; the panel reports the active stage, the elapsed time, and the remaining time before a pending agent enters. A user can pause, replay, or step a stage, or let the run advance automatically, which supports both qualitative inspection and the scripted evaluation of Sec. 5.3.

Programmatic control. The same session can be driven by software instead of a keyboard: a client opens a fixed-step control loop over the API and receives the rendered scene as an RGB frame after each action (Fig. 6c). Nothing about the observation changes between a browser user and an external planner, so an attached policy is evaluated on exactly the frames a person would see, which is the interface that makes Fig. 6 a validation surface rather than only a demo.

Manual driving and splat tuning. In manual mode the ego vehicle is driven from the keyboard while the reactive agents continue to respond, so the same closed loop that the planner exercises in Sec. 5 can be operated by hand. The tuning panel changes display parameters only and leaves the underlying scene files unchanged, which makes it a practical way to check how much of a rendered artifact comes from reconstruction error versus from the splatting configuration. A HUD reports frame rate and the number of drawn Gaussians against the streaming budget, and the camera can be switched between follow, first-person, free, mesh, and Gaussian views without leaving the session.

On the rendering quality of these captures. The frames in Fig. 6 were recorded from the browser frontend and are visibly softer than the rendered results of Sec. 5.2. This is a deliberate budget, not a limitation of the reconstruction: to hold an interactive frame rate the web viewer applies level-of-detail culling against a fixed splat budget of 2.5M, and the drawn count in panel (c) sits well below that cap. The same scene rendered offline at the full Gaussian count is what Table 2 measures, and the two are produced by one model. We report the interactive setting separately here so that the qualitative panels are not read as a quality claim.



(a) Self-play training replay

(b) Three-agent interaction, Stage 3

(c) Programmatic control through the session API

(d) Manual driving with live splat tuning

Figure 6: **The released circuits as a training and evaluation playground.** All panels are unedited captures of the browser frontend. **(a)** A reinforcement-learning training replay: two agents start from opposite ends of one stretch of road under a *single* policy at the same 200,000-step checkpoint, so the trajectories that emerge are a product of the interaction rather than of two separately scripted controllers. The elevated view shows their shared bird's-eye state and markers for the intermediate 0 and 100,000 checkpoints, and each inset streams the first-person observation the agent receives together with the action inferred at that frame, making the policy's decision visible as it is made. Hovering a goal marker reports the live distance to the parking zone and whether the agent has stopped within the arrival tolerance. **(b)** A staged three-agent interaction at Stage 3: agents A, B, and C carry distinct styles (emerald green, night purple, titanium) and are shown from their own first-person views as they negotiate an intersection, with the pending agent queued for entry at a stated time. Per-agent speed readouts and the action log let a user attribute an outcome to a specific policy rather than to the scenario. **(c)** The session API exposes the same circuit to an external planner: a client holds a fixed-step control session and receives the rendered Gaussian scene as an RGB frame after every action, so a planner is evaluated on the observations a browser user would see. Here the ego vehicle is at rest before any client has connected. **(d)** Manual driving: a user takes the ego vehicle over the reconstructed circuit while the surrounding agents keep responding, with live splat controls (Gaussian size, opacity, splat budget, level-of-detail, blur, and extent) and a frame-rate and drawn-Gaussian counter for inspecting the reconstruction under load.

D Additional Implementation Details

D.1 Policy Network

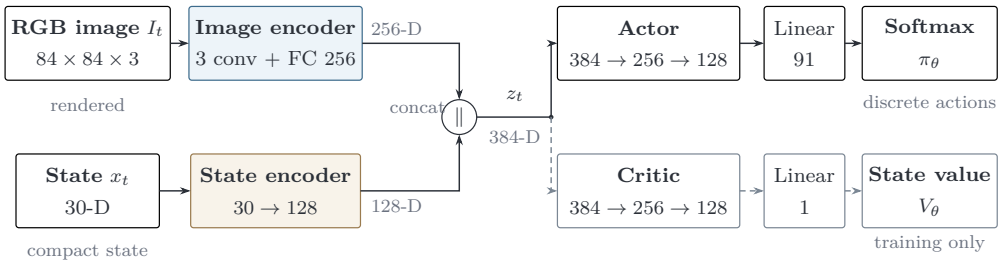
The actor and critic heads operate on the 256-dimensional joint representation $\mathbf{h}_j$, mapping it to the 91-way action distribution and the scalar state value respectively. All hidden layers use ReLU activations, and the encoder, shared trunk, and both heads are trained jointly under a single PPO objective, following the schedule and hyperparameters given in Sec. 3.2.

D.2 Reward and Fidelity Evaluation

During training, the activation of $\mathbb{I}_{\text{coll}}$ is modulated by the style-specific distance threshold $d_{\min}(\phi_j)$, so that ϕ selects a risk tolerance rather than a reward scale. The fidelity metric $D_{\text{JS}}(P_{\text{recorded}} \| P_{\text{react}})$ compares the recorded behavior distribution against the reactive one, in which P_{react} incorporates the ϕ_j-modified behaviors of the simulator's reactive agents (Sec. 3.3); the constrained optimization step preserves infrastructure compliance through the static component representations in (1).

E EGO PLANNER ARCHITECTURE

The reactive agents of Sec. 3.2 act on vectorized scene state, whereas the ego planner operates on rendered visual observations augmented with a compact geometric and navigation state. Those are different problems, and R^{2}Drive therefore keeps them in separate networks: the planner described here shares the reactive agents' PPO objective but not their input modality, so that closed-loop planning remains exposed to rendering fidelity exactly as a real perception stack would be.



(a) Multimodal encoders **(b)** Actor–critic heads

Figure 7: **Ego planner architecture.** A convolutional encoder embeds the rendered RGB frame and a state encoder embeds the state vector; the concatenated representation feeds an actor head over discrete driving actions and a critic head used during training.

Multimodal encoders. The image branch is a convolutional encoder that maps $I_t \in \mathbb{R}^{84\times84\times3}$ to a 256-dimensional embedding, and the state branch is a multi-layer perceptron that maps the 30-dimensional state vector x_t to a 128-dimensional embedding. The state vector provides complementary ego-motion, local geometric, and navigation information to the visual observation, including the ego vehicle's motion state, scan-line features encoding local ground-profile variations, and the relative distance and direction to the navigation goal. The two embeddings are concatenated into the 384-dimensional representation z_t of Fig. 7(a).

Actor–critic heads. From z_t, the actor head predicts a distribution over the 91 discrete actions and the critic head predicts a scalar state value used only during training, following the same objective as (6); at inference only the actor is retained. This reference network drives the interactive circuit in Fig. 1; Sec. 5.3 instead uses VAD and UniAD in the same closed-loop stack, with the same PPO schedule and world count (Appendix D).