Unraveling the Effects of Synthetic Data on End-to-End Autonomous Driving

Supplementary Material

A. Agent Spawn Setting Setting

As mentioned in Sec. 3, we adopt two agent spawn methods: route-based and trigger-based. Fig. 7 illustrates these two spawn methods.

Route-Based. With the map topology, we first sample lane points at regular intervals as candidate spawn points. From these points, we generate routes and filter out those whose shorter than a predefined distance threshold. We then assess whether these routes interact with ego vehicle's trajectory, forming a set of candidate agent routes. Each time an agent is placed, we remove candidate spawn points within a certain radius to control spawn density and ensure a reasonable scene distribution. Once the agent spawn is complete, all agent vehicles follow normal behavior control mechanisms. This approach can lead to complex and diverse interactions, enriching the generated simulation data.

Trigger-Based. We randomly select points along ego vehicle's trajectory as trigger points, with a higher preference for those located near intersections. Using the map topology, we locate the intersection closest to the route following the trigger point and identify all lane endpoints within the intersection as candidate spawn points for agent vehicles. For route generation, we select the most interactive route from road options such as Follow, TurnLeft, TurnRight, LaneChangeLeft, and LaneChangeRight, based on road topology and its interaction potential with ego vehicle. Unlike the route-based approach, spawned agents in the trigger-based method do not start moving immediately. Instead, they are activated and assigned an initial state only when the ego vehicle reaches the trigger point. Compared to the route-based method, the trigger-based approach ensures interaction and allows further expansion of possible events based on road topology.

B. 3D Gaussian Model Library

Figures 9 and 10 illustrate the foreground models and static models used in our experiments. During open-loop benchmark generation and closed-loop simulation, we select a specific background model from the background model library and register each agent vehicle using one of the foreground models from the foreground model library, which remains consistent throughout the process.

B.1. Background Rendering Detail

Dataset Selection. We select scenarios from Waymo Open Dataset[38] and categorize them based on time and vehicle behavior. Temporally, the scenarios are divided into Day and Night. Behaviorally, they are divided into Straight and Turn depending on whether the vehicle executes a turning maneuver within the scenario. With these classifications, we can easily select suitable scenes to better generate targeted scenarios to address corner cases.

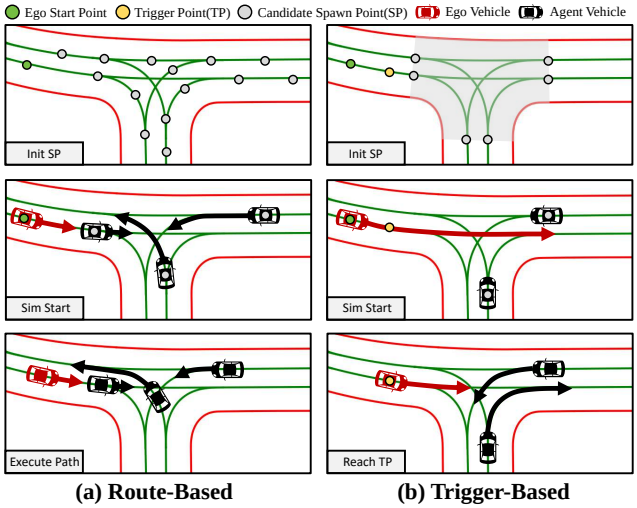

Figure 7. Illustration of agent spawn settings.

Model Training. The training process for our background models follows Street Gaussians. Unlike Street Gaussians[45], which focus on rendering images from the front three camera views, SceneCrafter requires a wider field of view for both open-loop benchmark generation and closed-loop simulation. Therefore, we use images from all five cameras in Waymo Open Dataset to train background models. This ensures that even when adjusting parameters such as camera field of view and relative positioning during simulation, we can still produce high-quality rendered images.

B.2. Foreground Rendering Detail

Our foreground vehicles consist of two parts: one is the virtual assets generated using BlenderNeRF, and the other comes from 3DRealCar. During training, we observe that the real dataset had limitations due to the absence of images from certain perspectives, such as the top and bottom views, as well as reflection challenges. These limitations lead to trained models that do not generalize well in the simulation environment. To address this issue, we explore two approaches.

Virtual Asset Library Editing and Rendering. The pipeline of foreground gaussian model training for each virtual asset is illustrated in Figure 12. To ensure that the rendered results of the 3D Gaussian Model align with real-world scene requirements and remove all reflection of asset,

we apply the following steps in Blender:

1. **Creating a Colored Initial Mesh.** We first create a new variable in *Data - Color Attributes - Color*, setting its domain to Vertex and its data type to Color. Then, using *Render - Bake*, we transfer the original material's color to the newly added color variable.
2. **Enabling Transparent Backgrounds.** To ensure that the rendered images contain only the vehicle and require no additional segmentation before training, we enable the *Render - Film - Transparent* option.
3. **Adjusting Lighting for Balanced Exposure.** To avoid overly dark or overexposed images, we configure the environment lighting under *World - Surface* by setting the background's HSV values to H=0,S=0,V=0.75.
4. **Remove Reflection.** We set the roughness of reflective and metallic material to be 1.0. This ensures that the rendering appearance of objects remains consistent across different perspectives, producing identical visual results at the same position regardless of the viewpoint.
5. **Ensuring Quality Across Varying Distances.** To maintain consistent rendering quality of foreground models at different distances, we use BlenderNeRF[33] to render images of the same 3D asset at 4m, 10m, and 15m distances. For each distance, 150 images are rendered.

Following the above setup, for each 3D asset, we generate a total of 450 images captured from different distances and angles, along with the corresponding camera transforms for each image. Besides, we obtain an initial mesh with color attributes. The number of vertices in this mesh depends on the original vehicle model's resolution, ranging from 100,000 to 600,000 vertices.

Virtual Asset Library Model Training. We use all the rendered images, camera parameters, and the initial mesh from the above process as training data. The initial mesh is downsampled to a maximum of 100,000 vertex to ensure the resulting model remained compact. For each model, the training process follows that of 3D Gaussian Splatting[23]. Specifically, we trained the model for 30,000 iterations.

3DRealCar Library Model Training. We followed the preprocessing steps provided by 3DRealCar to process the raw images, obtaining the vehicle's initial mesh and mask. The model training pipeline is consistent with 3D Gaussian Splatting. However, as noted in 3DRealCar [13], reflection in the original dataset impact the training quality. To address this issue easily, we set the original image backgrounds to green and purple, alternately using different backgrounds during training. Throughout the process, we progressively removed background color points to obtain the final 3DGS model at every 1,500 iteration. In the future, we plan to explore more efficient methods for rapidly generating clean foreground vehicle models.

	NeuroNCAP[30]	DriveArena[47]	Street Gaussians[45]	SceneCrafter
FPS	1.282	0.198	18.922	11.574

Table 6. Quantitative runtime results on different closed-loop simulation platform renderer.

C. Experiment Settings on Runtime Efficiency

In this section, we validate that SceneCrafter achieves real-time rendering capabilities and demonstrates significantly faster performance compared to other simulation platforms that generate high-fidelity images.

All 3D Gaussian Models in SceneCrafter are trained on a single NVIDIA GeForce RTX 4090 GPU. For Street Gaussians[45] and SceneCrafter , we train a scene without any dynamic foreground objects from the Waymo Open Dataset[38]. During rendering, Street Gaussians generate images using the original settings, while SceneCrafter includes additional foreground models. Rendering for NeuroNCAP[30] and DriveArena[47] is performed directly using their provided checkpoints. All rendering experiments are conducted on a single NVIDIA GeForce RTX 4090 GPU. Frame rates are calculated based on the time required to render all views for a single frame. In SceneCrafter and NeuroNCAP, we render six 1600×900 images per frame. In DriveArena, a 2400×224 image is rendered, but unlike the original DriveArena setup, we do not further divide it into six 400×224 images or upsample them to a 1600×900 resolution.

Results. SceneCrafter demonstrates a clear advantage in balancing real-time performance with high-fidelity rendering. While the introduction of additional foreground models and the model fusion process slightly increases computational demand compared to Street Gaussians, SceneCrafter effortlessly maintains real-time rendering capabilities. Compared to existing methods based on NeRF and diffusion models, SceneCrafter delivers significantly faster rendering speeds along with spatial-temporal consistency. This unique combination of efficiency and quality positions SceneCrafter as a potential method for real-time, high-resolution simulation. It not only meets the demands of real-time simulation platforms, reaffirming its superior capability in rendering dynamic, interactive scenarios.

D. Ego Speed Alteration and Interaction Rate

D.1. Ego Speed Alteration

For each planning frame t, we calculate the future trajectory offset between two consecutive future frames, $\{(w_x^t, w_y^t), (w_x^{t+1}, w_y^{t+1}), \ldots, (w_x^{t+5}, w_y^{t+5})\}$. For each timestamp i from 0 to 6, we compute the difference between the magnitudes of consecutive waypoints, $\left\|w^{t+i+1}\right\|_2 - \left\|w^{t+i}\right\|_2$, to determine whether the vehicle is accelerating or decelerating. When the trend switches from acceleration

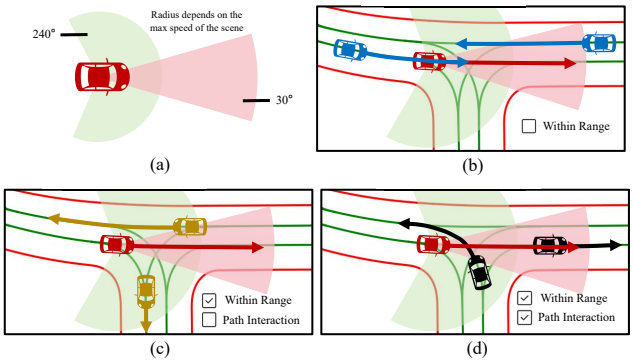

Figure 8. Example of interaction. (a) Illustrates the interaction range, which consists of two parts: a 240° surrounding zone and a 30° forward zone. The radius is determined by the maximum speed of ego vehicle in the original scene data. The vehicles in (b) condition are not considered to be interacting with the ego vehicle as they fall outside the interaction range. Although the vehicles in (c) condition are within the interaction range, their routes do not intersect with ego vehicle's route, which will not be classified as interacting. The vehicles in (d) condition are both within the interaction range and their routes interact with ego vehicle, thus they are considered to be interacting.

to deceleration, or vice versa, we increment the count for Ego Speed Alteration for the frame t by 1.

D.2. Interaction Judgment

The determination of interaction consists of two key criteria. First, whether the vehicle is within the defined interaction range. Second, whether its route intersects with that of ego vehicle. A vehicle is only considered to be interacting if both conditions are met, as illustrated in Eq. (6). Also example are given in Fig. 8.

$$\mathbb{I}_{\text{detect}}(v) = \begin{cases} 1, & \text{if } v \text{ is within range} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$\mathbb{I}_{\text{interact}}(v) = \begin{cases} 1, & \text{if path of } v \text{ intersects} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\mathbb{I}_{\text{interaction}}(v) = \mathbb{I}_{\text{detect}}(v) \cdot \mathbb{I}_{\text{interact}}(v) \tag{6}$$

E. Justification for comparison with other mainstream simulator

We do not compare our experimental results with other driving simulators such as CARLA [12], DriveArena [47], and HugSim [53] due to fundamental differences in their design and capabilities. CARLA lacks sufficient realism in sensor simulation and vehicle dynamics, making it unsuitable as input for end-to-end model trained on real world data. DriveArena is designed around the nuScenes dataset, and adapting it to the Waymo dataset would require significant effort and computational resources, making a fair comparison impractical. HugSim, on the other hand, does not incorporate an expert planner and is unable to generate comprehensive driving logs that include both sensor inputs and ego dynamics. Given these limitations, a direct experimental comparison with these simulators would not provide meaningful insights.

F. Spatial-temporal Consistency

As mentioned in Section 3.2, the parameters of the composite Gaussian model are fully defined for each frame, with only the vehicle positions changing over time. Consequently, SceneCrafter is inherently capable of maintaining spatial-temporal consistency during the rendering process. As shown in Figure 13, both the background and the incorporated foreground objects consistently retain their coherence across consecutive renderings.

G. Closed-loop Evaluation Metrics

Our closed-loop evaluation follows CARLA, calculating key metrics such as route completion (RC), vehicle collision rate (VCR), and layout collision rate (LCR). Vehicle collision rate (VCR) is calculated as:

$$VCR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{j=1}^{T_i} Col_{vehicle}(j) \tag{7}$$

where N is the number of scenarios, T_i is the number of the i-th scenario frames, and $Col_{vehicle}(j)$ indicates whether ego vehicle collides with other vehicles in j-th frame.

Similarly, layout Collision Rate (LCR) is calculated as:

$$LCR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{j=1}^{T_i} Col_{layout}(j) \tag{8}$$

The Route Completion (RC) is determined based on the absence of vehicle collisions and timeouts, as follows:

$$RC = \frac{1}{N}\sum_{i=1}^{N}\delta(VCR_i = 0 \wedge \neg \text{timeout}_i) \tag{9}$$

where timeout_i indicates whether ego vehicle's behavior timed out in the i-th scenario, and the function $\delta(\cdot)$ returns 1 when the condition is met, and 0 otherwise. The calculation of RC excludes LCR because we observe that the original model frequently collides with the layout in our created scenarios, resulting in zero RC score across the board, which diminishes its comparative value.

H. Adaptive Kinematic Model Estimation

Given real world IMU data from two consecutive frames $(x_t, y_t, \varphi_t, v_t, x_{t+1}, y_{t+1}, \varphi_{t+1}, v_{t+1})$ in waymo with $\Delta t = 0.1s$, we can obtain based on the second line of Eq. (1) that:

$$\beta_t = \arcsin\left(\frac{l_f}{v_t \Delta t}(\varphi_{t+1} - \varphi_t)\right)$$

where β_t is the slip angle, v_t is the vehicle velocity, φ_t is the yaw angle and x_t, y_t represent the vehicle position. Then we aim to obtain the best value of u_1 and u_2 by solving the following optimization problem:

$$\min_{u_1,u_2} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left((x_{t+1} - \hat{x}_{t+1})^2 + (y_{t+1} - \hat{y}_{t+1})^2\right)$$

subject to:

$$\begin{aligned} v_u &= (1 - \boldsymbol{u_1}) v_t + \boldsymbol{u_1} v_{t+1}, \\ \hat{x}_{t+1} &= x_t + v_u \cos(\varphi_t + \boldsymbol{u_2} \cdot \beta_t)\Delta t, \\ \hat{y}_{t+1} &= y_t + v_u \sin(\varphi_t + \boldsymbol{u_2} \cdot \beta_t)\Delta t. \end{aligned}$$

Where N represent the number of scenes and T_n represent the number of frame pairs in the n th scene.

I. End-to-End Model Training Details

The training of the E2E model follows the official repositories of VAD and GenAD. To align with the Waymo Open Dataset settings, we use five cameras and set the output dimension of the detection classification head to four, corresponding to the four categories in the dataset. Training is stopped at epoch 30, as the validation loss begins to increase beyond this point. For fine-tuning on real data, we use the same configuration as the initial training while adjusting the initial learning rate to $lr = 5e^{-5}$

J. Additional Closed-Loop Results

Tab. 7 presents the performance of original and fine-tuned models on scenes from $\mathcal{S}_{recon}$. Surprisingly, the original model performs poorly in closed-loop evaluation, despite being trained on real-world logs from the same scenes. We attribute this to a distributional bias in the training set, where approximately one-third of samples involve only acceleration or deceleration without phase changes. As a result, imitation-learning-based E2E models tend to adopt a flawed pattern—either continuously accelerating until a collision with other agents or decelerating to a stop. Our results show that synthetic data helps mitigate this bias, leading model to learn more reasonable driving patterns.

Method	All Test Scenes			Turn Scenes		
	RC↑	VC↓	LCR↓	RC↑	VC↓	LCR↓
VAD	29.88	8.35	8.62	19.44	7.92	11.67
VAD$_{ft}$	**39.02**	**6.93**	**5.67**	**31.15**	**5.71**	**4.53**

Table 7. Closed-Loop Evaluation of different models across S_{recon}. Performance metrics include: **RC** (Route Completion), **VC** (Vehicle Collision Rate, and **LCR** (Layout Collision Rate).

Index	Segment Name	Time	Type
019	segment-10275144660749673822_5755_561_5775_561_with_camera_labels	Day	Turn
036	segment-10676267326664322837_311_180_331_180_with_camera_labels	Day	Straight
053	segment-11017034898130016754_697_830_717_830_with_camera_labels	Day	Straight
078	segment-11623618970700582562_2840_367_2860_367_with_camera_labels	Day	Turn
155	segment-13390791323468600062_6718_570_6738_570_with_camera_labels	Day	Turn
250	segment-15445436653637630344_3957_561_3977_561_with_camera_labels	Day	Turn
276	segment-15943938987133888575_2767_300_2787_300_with_camera_labels	Day	Turn
367	segment-17818548625922145895_1372_430_1392_430_with_camera_labels	Day	Turn
382	segment-18111897798871103675_320_000_340_000_with_camera_labels	Day	Turn
402	segment-1918764220984209654_5680_000_5700_000_with_camera_labels	Day	Straight
427	segment-2259324582958830057_3767_030_3787_030_with_camera_labels	Day	Turn
451	segment-268278198029493143_1400_000_1420_000_with_camera_labels	Night	Straight
619	segment-580580436928611523_792_500_812_500_with_camera_labels	Night	Straight
674	segment-7007702792982559244_4400_000_4420_000_with_camera_labels	Night	Straight
739	segment-8513241054672631743_115_960_135_960_with_camera_labels	Night	Straight

Table 8. Information for some selected scenarios in the Waymo Open Dataset.

Figure 9. Overview of foreground gaussian model library.

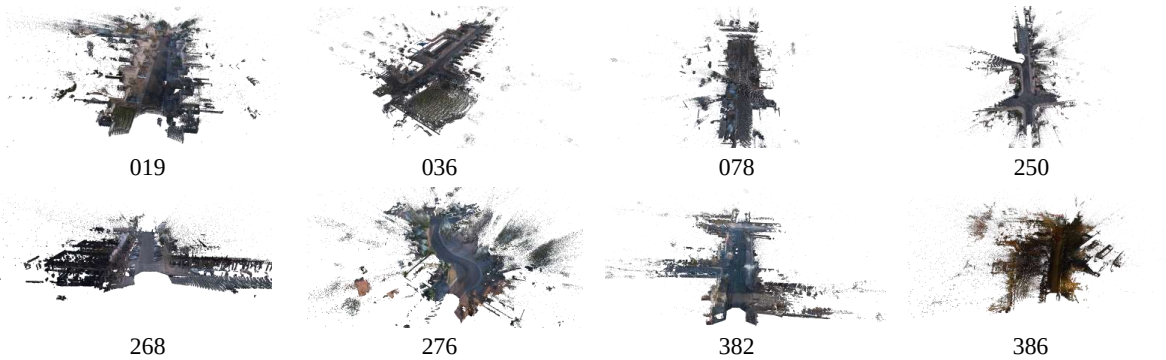

Figure 10. Overview of background gaussian model library.

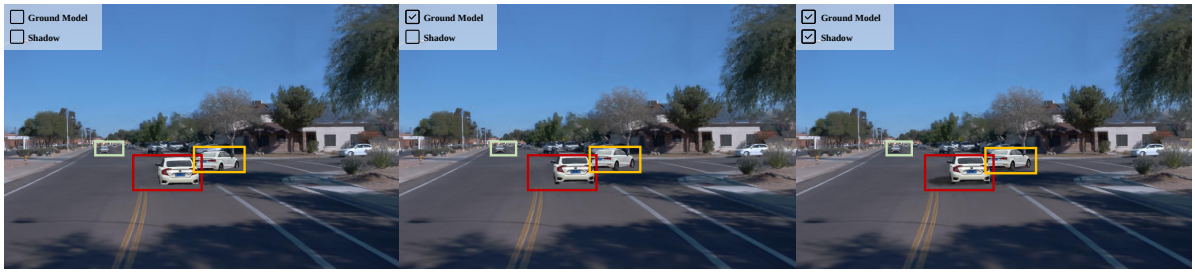

Figure 11. Example of the effect of ground model and shadow addition.

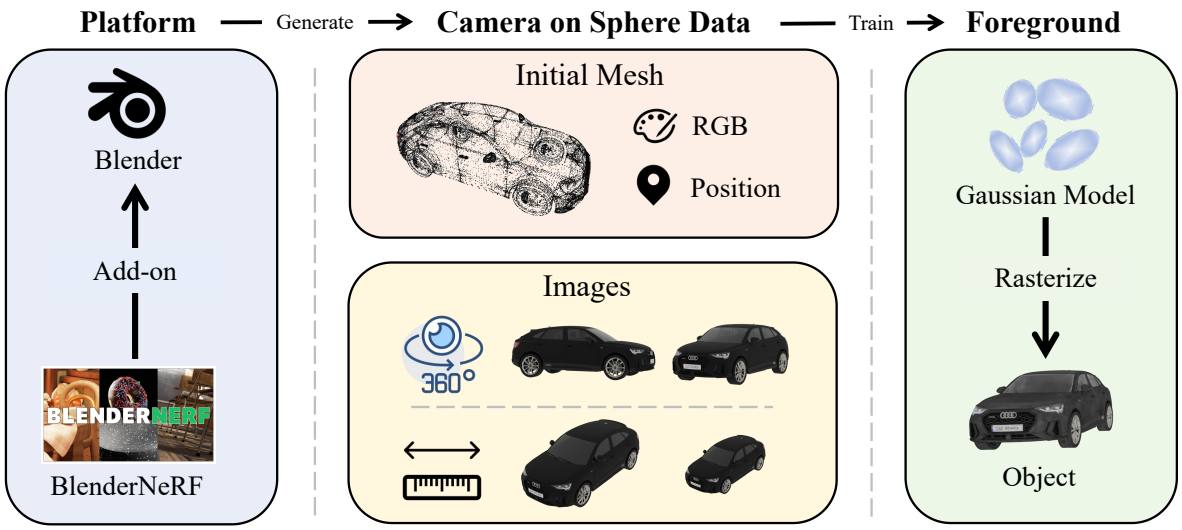

Figure 12. Overview of foreground gaussian model training pipeline for virtual assets. We used Blender with add-on BlenderNeRF to generate camera-on-sphere data, which is collected along a spherical trajectory with a specified radius. Generated data including images and a intial colored mesh is then used to train foreground Gaussian model.

Figure 13. Reconstructed perspectives from multi-view cameras across sequential time steps. Each row represents a specific timestamp, with time advancing incrementally downward. The visualization demonstrates the rendering outcomes of surround-view cameras over time, highlighting spatial-temporal consistency of SceneCrafter . The original video is also in the supplementary material.