Collaborative LLM-Agents for Editable Driving Scene Simulation

Anonymous authors
Paper under double-blind review

Abstract

Scene simulation in autonomous driving has gained significant attention because of its huge potential for generating customized data. However, existing editable scene simulation approaches face limitations in terms of user interaction efficiency, multi-camera photo-realistic rendering and external digital assets integration. To address these challenges, this paper introduces *ChatSim*, the first system that enables editable photo-realistic 3D driving scene simulations via natural language commands with external digital assets. To enable editing with high command flexibility, *ChatSim* leverages a large language model (LLM) agent collaboration framework. To generate photo-realistic outcomes, *ChatSim* employs a novel multi-camera neural radiance field method. Furthermore, to unleash the potential of extensive high-quality digital assets, *ChatSim* employs a novel multi-camera lighting estimation method to achieve scene-consistent assets' rendering. Our experiments on Waymo Open Dataset demonstrate that *ChatSim* can handle complex language commands and generate corresponding photo-realistic scene videos.

1 Introduction

Perception Chen et al. (2015; 2017); Wu et al. (2017); Chen et al. (2016); Caesar et al. (2020); Li et al. (2020) is the window of an autonomous vehicle into the external environment. To ensure the robustness of the vehicle's perceptual capabilities during both training and testing phases, it necessitates the collection of high-quality perception data in substantial volumes Chang et al. (2019); Sun et al. (2020b). However, the operation of a fleet for the acquisition of real-world data often incurs prohibitive expenses, particularly for specialized or customized requirements. For instance, in the aftermath of an accident or intervention involving an autonomous vehicle, it is imperative to test the vehicle's perception system across a spectrum of similar scenarios. While replicating such scenario data from real-world instances is nearly impossible due to the uncontrollability of actual scenes Rong et al. (2020); Bergamini et al. (2021), customized scene simulation emerges as a vital and feasible alternative. It enables the precise modeling of specific conditions without high costs and logistical complexities of real-world data collection Amini et al. (2020); Yang et al. (2020).

To effectively simulate customized driving scenes, we identify three key properties as fundamental. First, the simulation should be capable of following sophisticated or abstract demands, thereby facilitating the production. Second, the simulation should generate photo-realistic, view-consistent outcomes, which allow for the closest approximation to vehicle observations in real-world scenarios. Third, it should allow for the integration of external digital assets Miric et al. (2019); Banta (2016) with their photo-realistic textures and materials while fitting the lighting conditions. This capability would unlock the potential for data expansion by incorporating a wide array of external digital assets, satisfying customized needs.

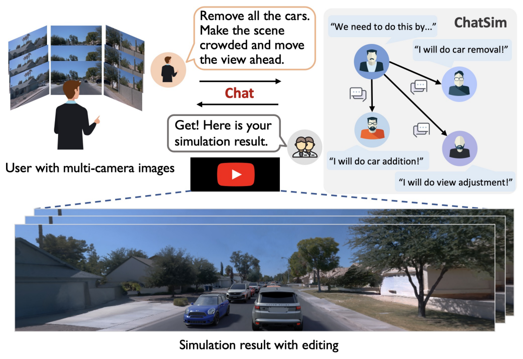

Remove all the cars. Make the scene crowded and move the view ahead.
Chat
Get! Here is your simulation result.
User with multi-camera images
ChatSim
"We need to do this by..."
"I will do car removal!"
"I will do car addition!"
"I will do view adjustment!"
Simulation result with editing

Figure 1: *ChatSim* enables the editing of photo-realistic 3D driving scene simulations via language commands.

A vast array of significant works have been proposed for scene simulation, yet they fail to meet all three of these requirements. Traditional graphics engines, such as CARLA Dosovitskiy et al. (2017) and UE Engine, offer editable virtual environments with external digital assets, but the data realism is restricted by asset modeling and rendering qualities. Image generation based methods, such as BEVControl Yang et al. (2023a), DriveDreamer Wang et al. (2023b), MagicDrive Gao et al. (2023), can generate realistic scene images based on various control signals, including BEV maps, bounding boxes and camera poses. However, they struggle to maintain view consistency and face challenges in importing external digital assets due to the absence of 3D spatial modeling. Rendering-based methods have been proposed to obtain photo-realistic and view-consistent scene simulation. Notable examples like UniSim Yang et al. (2023b) and MARS Wu et al. (2023b) come equipped with a suite of scene-editing tools. However, these systems require extensive user involvement in every trivial editing step via code implementation, which is ineffective when performing the editing. Furthermore, while they handle vehicles in observed scenarios effectively, their inability to support external digital assets restricts opportunities for data expansion and customization.

To fulfill the identified requirements, we introduce *ChatSim*, the first system that enables editable photo-realistic 3D driving scene simulations via natural language commands with external digital assets. To use *ChatSim*, users simply engage in a conversation with the system, issuing commands through natural language without any involvement in intermediate simulation steps; see Figure 1 for illustration. To address complex or abstract user commands effectively, *ChatSim* adopts a large language model (LLM)-based multi-agent collaboration framework. The key idea is to **exploit multiple LLM agents**, each with a specialized role, to decouple an overall simulation demand into specific editing tasks, thereby mirroring the task division and execution typically founded in the workflow of a human-operated company. This workflow offers two key advantages for scene simulation. First, LLM agents' ability to process human language commands allows for intuitive and dynamic editing of complex driving scenes, enabling precise adjustments and feedback. Second, the collaboration framework enhances simulation efficiency and accuracy by distributing specific editing tasks among specialized agents, ensuring detailed and realistic simulations with improved task completion rates.

To generate photo-realistic outcomes, we propose McNeRF in *ChatSim*, a novel neural radiance field method that incorporates multi-camera inputs, offering a broader scene rendering. This integration fully exploits camera setups on vehicles but raises two significant challenges: camera pose misalignment due to asynchronized trigger times and brightness inconsistency due to different camera exposure times. To address camera pose misalignment, McNeRF uses a multi-camera alignment to reduce extrinsic parameter noises, ensuring rendering quality. To address brightness inconsistency, McNeRF integrates the critical exposure times to recover scene radiance in HDR, markedly mitigating the issue of color discrepancies at the intersections of two camera images with different exposure times. To import external digital assets with their realistic textures and materials, we propose McLight, a novel multi-camera lighting estimation that blends skydome and surrounding lighting. Our skydome estimation restores accurate sun behavior with peak intensity residual connection, enabling the rendering of prominent shadows. For surrounding lighting, McLight queries McNeRF to achieve complex location-specific illumination effects, like those in the tree shade with sunlight being blocked. This significantly improves the rendering realism of the integrated 3D assets.

We conduct extensive experiments on the Waymo autonomous driving dataset and show that *ChatSim* generates photo-realistic customized perception data including dangerous corner cases according to various human language commands. Our method is compatible with mixed, highly-abstract and multi-round commands. Our method achieves SoTA performance with an improvement of 4.5% in photo-realism with a wide-angle rendering. Moreover, we demonstrate our lighting estimation outperforms the SoTA methods both qualitatively and quantitatively, reducing the intensity error and angular error by 57.0% and 9.9%.

2 Related Work

Scene simulation for autonomous driving. Current scene simulation methods can be generally divided into three categories: graphics engines, image generation, and scene rendering. Graphics engines, such as CARLA Dosovitskiy et al. (2017), AirSim Shah et al. (2018), OpenScenario Editor Editor, 51Sim-One 51Sim-One and RoadRunner Crescenzi et al. (2001), create a virtual world for simulating a wide range of driving scenarios. However, there exists a significant domain gap between

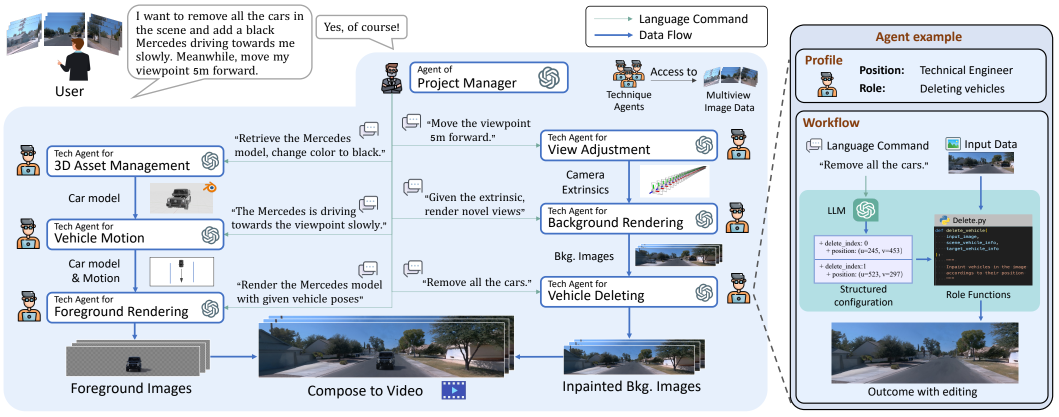
Method	Photo-realistic	Dim.	Multi-camera	Editable	External assets	Language	Open-source
CARLA Dosovitskiy et al. (2017)	×	3D	✓	✓	✓	×	✓
AirSim Shah et al. (2018)	×	3D	✓	✓	✓	×	✓
OpenScenario Editor	×	3D	✓	✓	✓	×	✓
51Sim-One 51Sim-One	×	3D	✓	✓	✓	×	×
RoadRunner Crescenzi et al. (2001)	×	3D	✓	✓	✓	×	✓
BEVGen Swerdlow et al. (2023)	✓	2D	✓	✓	×	×	✓
BEVControl Yang et al. (2023a)	✓	2D	✓	✓	×	×	×
DriveDreamer Wang et al. (2023b)	✓	2D	✓	✓	×	✓	×
DrivingDiffusion Li et al. (2023a)	✓	2D	✓	✓	×	✓	×
GAIA-1 Hu et al. (2023)	✓	2D	×	✓	×	✓	×
MagicDrive Gao et al. (2023)	✓	2D	✓	✓	×	×	×
READ Li et al. (2023b)	✓	3D	×	×	×	×	✓
Neural SG Ost et al. (2021)	✓	3D	×	✓	×	×	✓
Neural PLF Ost et al. (2022)	✓	3D	×	×	×	×	✓
S-NeRF Xie et al. (2023)	✓	3D	✓	×	×	×	✓
UniSim Yang et al. (2023b)	✓	3D	×	✓	×	×	×
MARS Wu et al. (2023b)	✓	3D	×	✓	×	×	✓
ChatSim (Ours)	✓	3D	✓	✓	✓	✓	✓

Table 1: Comparison of existing and proposed methods for autonomous driving simulation.

the virtual world and reality. Image generation methods can generate realistic scene images based on different control signals, such as HD maps Swerdlow et al. (2023); Gao et al. (2023); Li et al. (2023a), sketch layout Yang et al. (2023a), bounding boxes Li et al. (2023a); Wang et al. (2023b); Gao et al. (2023), text Li et al. (2023a); Wang et al. (2023b); Gao et al. (2023); Hu et al. (2023) and driving actions Wang et al. (2023b); Hu et al. (2023). However, these approaches can hardly maintain scene consistency. To obtain a coherent driving scene, methods based on scene rendering target to reconstruct the 3D scene. READ Li et al. (2023b) employs point clouds and uses a U-Net to render images. With the rapid development of Neural Radiance Field (NeRF) Mildenhall et al. (2021); Barron et al. (2021; 2022); Müller et al. (2022); Sun et al. (2022); Wang et al. (2023a), several works Xie et al. (2023); Guo et al. (2023); Yang et al. (2023b); Wu et al. (2023b); Ost et al. (2021); Turki et al. (2023); Kundu et al. (2022); Ost et al. (2022) also exploit NeRFs to model cars and static street backgrounds in outdoor environments. Moreover, notable examples like UniSim Yang et al. (2023b) and MARS Wu et al. (2023b) come equipped with a suite of scene-editing tools. However, these methods require extensive user involvement in intermediate editing steps and they fail to support external digital assets for data expansion. In this work, we propose *ChatSim* that achieves automatic simulation editing via language commands and integrates external digital assets to enhance realism and flexibility. In *ChatSim*, we integrate McNeRF, a novel neural radiance field designed to leverage multi-camera inputs for high-fidelity rendering.

Lighting estimation. Lighting estimation focuses on assessing the illumination conditions of a real-world environment to seamlessly integrate digital objects. Early methods Lalonde & Matthews (2014); Lalonde et al. (2010) for outdoor environments use explicit cues like detected shadows on the ground. Recent works usually adopt learning-based approaches Garon et al. (2019); Hold-Geoffroy et al. (2019); LeGendre et al. (2019); Hold-Geoffroy et al. (2017); Li et al. (2018); Zhang et al. (2019) by predicting different lighting representations like spherical lobes Boss et al. (2020); Li et al. (2018), light probes LeGendre et al. (2019), environment map Sengupta et al. (2019); Somanath & Kurz (2021), HDR sky model Hold-Geoffroy et al. (2019); Zhang et al. (2019); Wang et al. (2022) and lighting volume Wang et al. (2022). However, few of them consider multi-camera input, which is common for driving scenarios. In this paper, we propose a novel multi-camera lighting estimation method, McLight, combining with our McNeRF, to estimate a wider range of lighting and obtain the spatially-varying lighting effects of assets.

Large language model and collaborative framework. Large Language Models (LLMs) are AI systems trained on extensive data to understand, generate, and respond to human language. GPT Brown et al. (2020) is a pioneering work to generate human-like content. The following updated versions GPT-3.5 ChatGPT and GPT-4 OpenAI (2023), provide more intelligent capabilities like chatting, browsing and coding. Notable other large language models include InstructGPT Ouyang et al. (2022), LLaMA Touvron et al. (2023) and PaLM Chowdhery et al. (2022); Anil et al. (2023). Based on LLM, many works Wang et al. (2023c); Du et al. (2023); Zhuge et al. (2023); Hao et al. (2023); Akata et al. (2023) improve the problem-solving abilities by integrating communication among multiple agents. Hong et al. (2023) and Wu et al. (2023a) define a group of well-organized agents to form operating procedures with conversation and code programming. In this paper, we exploit the power of collaborative LLM agents in simulation for autonomous driving, enabling the various editing of 3D scenes via language commands.


User
I want to remove all the cars in the scene and add a black Mercedes driving towards me slowly. Meanwhile, move my viewpoint 5m forward.
Yes, of course!
Language Command
Data Flow
Agent of Project Manager
Technique Agents
Access to
Multiview Image Data
"Retrieve the Mercedes model, change color to black."
"Move the viewpoint 5m forward."
Tech Agent for 3D Asset Management
Tech Agent for View Adjustment
Car model
Camera Extrinsics
"The Mercedes is driving towards the viewpoint slowly."
"Given the extrinsic, render novel views"
Tech Agent for Vehicle Motion
Tech Agent for Background Rendering
Car model & Motion
Bkg. Images
"Render the Mercedes model with given vehicle poses"
"Remove all the cars."
Tech Agent for Foreground Rendering
Tech Agent for Vehicle Deleting
Foreground Images
Compose to Video
Inpainted Bkg. Images
Agent example
Profile
Position: Technical Engineer
Role: Deleting vehicles
Workflow
Language Command
Input Data
"Remove all the cars."
LLM
+ delete_index: 0
+ position: (u=245, v=453)
+ delete_index:1
+ position: (u=523, v=297)
Structured configuration
Delete.py
Role Functions
Outcome with editing

```
def delete_vehicle(
    input_image,
    scene_vehicle_info,
    target_vehicle_info
):
    """
    Inpaint vehicles in the image
    accordings to their position
    """
```

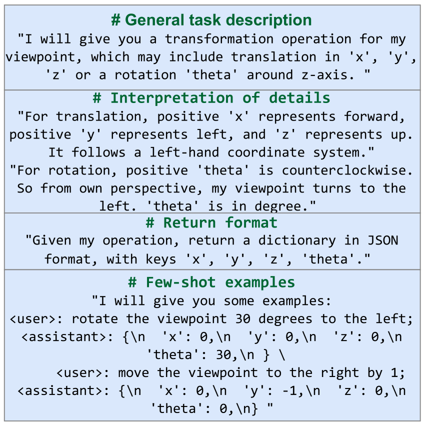
Figure 2: ***ChatSim* system overview.** The system exploit multiple collaborative LLM agents with specialized roles to decouple an overall demand into specific editing tasks. Each agent equips an LLM and corresponding role functions to interpret and execute its specific tasks.

3 Collaborative LLM-Agents for Editing

The *ChatSim* system analyzes specific user commands and returns a video that meets customized needs; see Figure 2. Since user commands could be abstract and sophisticated, it requires the system to have flexible task-handling ability. Directly applying a single LLM agent struggles with multi-step reasoning and cross-referencing. To address this, we design a series of collaborative LLM agents, where each agent is responsible for a unique aspect of the editing task.

3.1 Specific Agent's Functionality

Agents in *ChatSim* comprise two key components: a Large Language Model (LLM) and the corresponding role functions. The LLM is responsible for understanding the received commands while the role functions process the received data. Each agent is equipped with unique LLM prompts and role functions tailored to their specific duties within the system. To accomplish their tasks, agents first convert the received commands to a structured configuration using LLM with the assistance of prompts. Then the role functions utilize the structured configuration as parameters to process the received data and produce the desired outcomes; see an agent example on the right side of Figure 2. This workflow endows agents with both language interpretation capabilities and precise execution capabilities.



```
# General task description
"I will give you a transformation operation for my
viewpoint, which may include translation in 'x', 'y',
'z' or a rotation 'theta' around z-axis. "
# Interpretation of details
"For translation, positive 'x' represents forward,
positive 'y' represents left, and 'z' represents up.
It follows a left-hand coordinate system."
"For rotation, positive 'theta' is counterclockwise.
So from own perspective, my viewpoint turns to the
left. 'theta' is in degree."
# Return format
"Given my operation, return a dictionary in JSON
format, with keys 'x', 'y', 'z', 'theta'."
# Few-shot examples
"I will give you some examples:
<user>: rotate the viewpoint 30 degrees to the left;
<assistant>: {\n  'x': 0,\n  'y': 0,\n  'z': 0,\n
'theta': 30,\n } \
<user>: move the viewpoint to the right by 1;
<assistant>: {\n  'x': 0,\n  'y': -1,\n  'z': 0,\n
'theta': 0,\n} "
```

Figure 3: Prompt of view adjustment agent.

Project Manager Agent. The project manager agent decomposes direct commands into clear natural language instructions dispatched to other editing agents. To equip the project manager agent with the capability of command decomposition, we design a series of prompts for its LLM. The core idea of the prompts is to describe the action set, give the overall goal, and define the output form with examples; The role functions send the decomposed instructions to other agents for editing. The presence of the project manager agent enhances the system's robustness in interpreting various inputs and streamlines operations for clarity and fine granularity.

Tech agent for view adjustment. The view adjustment agent generates suitable extrinsic camera parameters. The LLM in the agent translates the natural language instructions for viewpoint adjustment into movement parameters to the target viewpoint's position and angle. In role functions, these movement parameters are turned into transformation matrices required by the extrinsic, which are then multiplied by the original parameters to yield a new viewpoint.

Tech agent for background rendering. The background rendering agent renders the scene background based on multi-camera images. The LLM receives the rendering command and then operates the role functions for rendering. Notably, in role functions, we specifically integrate a novel neural radiance field method (McNeRF) taking multi-camera inputs and considering exposure time, solving the problem of blurring and brightness inconsistency in multi-camera rendering, see more details in Section 4.1.

Tech agent for vehicle deleting. The vehicle deleting agent removes specified vehicles from the background. It first identifies current vehicle attributes like 3D bounding boxes and colors from given scene information or results from a scene perception model like Liu et al. (2023). The LLM gathers attributes of the vehicles and performs matching with user requests. Upon confirming the targeted vehicles, it employs a per-frame inpainting model as the role functions, such as latent diffusion methods Rombach et al. (2021), to effectively delete them from the scene.

Tech agent for 3D asset management. The 3D asset management agent selects and modifies 3D digital assets according to user specifications. It constructs and maintains a 3D digital asset bank; see our bank details in the Appendix.E. To facilitate the addition of various objects, the agent first uses LLM to select the most suitable asset by key attributes matching with the requirements, such as color and type. If the matching is not perfect, the agent could modify the asset through its role functions like changing the color.

Tech agent for vehicle motion. The vehicle motion agent creates the initial places and subsequent motions of vehicles following the requests. Existing vehicle motion generation methods cannot directly generate motion purely from text and the scene map. To solve the problem, here we propose a novel text-to-motion method. The key idea is linking a placement and planning module as role functions with LLMs to extract and turn motion attributes into coordinates. Motion attributes include position attributes (e.g., distance, direction) and movement attributes (e.g., speed, action). For the placement module, we endow each lane node in the lane map with its attributes to match with the position attributes. The planning module plans the vehicle's approximate destination lane node and then plans the intermediate trajectory by fitting the Bezier curves. We also add trajectory tracking to fit vehicle dynamics; see more details in the Appendix.G.

Tech agent for foreground rendering. The foreground rendering agent integrates camera extrinsic infomation, 3D assets, and motion information to render foreground objects in the scene. Notably, to seamlessly integrate the external assets with the current scene, we design a multi-camera lighting estimation method (McLight) into the role functions, coupling with McNeRF. The estimated illumination is then utilized by Blender API to generate foreground images. The detailed technical aspects will be elaborated in Section 4.2.

3.2 Agent Collaboration Workflow

Agents with tailored functions collaboratively work together to edit based on user commands. The project manager orchestrates and dispatches instructions to editing agents. The editing agents form two teams: background generation and foreground generation. For background generation, the background rendering agent generates rendered images using the extrinsic parameters from the view adjustment agent, followed by inpainting by the vehicle deleting agent. For foreground generation, the foreground rendering agent renders the images using the extrinsic parameters from the view adjustment agent, selected 3D assets from 3D asset management agent, and generated motions from vehicle motion agent. Finally, the foreground and background images are composed to create and deliver a video to the user. The editing information in each agent's configuration is recorded by the project manager agent for possible multi-round editings.

4 Novel Rendering Methods

Based on the collaborative LLM agents framework introduced in Section 3, this section presents two novel rendering techniques to enhance photo-realism in simulations. To tackle the rendering challenges caused by multiple cameras, we propose multi-camera neural radiance field (McNeRF), a novel NeRF model considering the varied camera exposure times for visual consistency. To render realistic external digital assets with location-specific lighting and accurate shadows, we propose McLight, a novel hybrid lighting estimation method combined with our McNeRF. Note that McNeRF and McLight are leveraged by the background rendering agent and the foreground rendering agent, respectively.

4.1 McNeRF for Background Rendering

An autonomous vehicle typically equips multiple cameras to achieve a comprehensive perception view. However, this poses challenges for NeRF training due to the misaligned multi-camera poses from asynchronized camera trigger times and the brightness inconsistency originating from different exposure times. To address these challenges, the proposed McNeRF uses two techniques: multi-camera alignment and brightness-consistent rendering.

Multi-camera alignment. Autonomous vehicles, despite having a localization module for accurate camera poses, face challenges with asynchronous trigger times across multiple cameras. To align camera extrinsics for NeRF training, our core idea is to leverage a consistent spatial coordinate system provided by Agisoft Metashape Agisoft (2019) to align the images captured by multiple cameras at different timestamps.

Specifically, let $\mathcal{I}^{(i,k)}$ and $\xi^{(i,k)}$ be the image captured by the ith camera at the kth trigger and the corresponding camera pose in the vehicle's global coordinate space, respectively. We first input all images into Metashape for recalibration. The aligned camera pose is then obtained as:

$$\widehat{\xi}^{(i,k)} = T_{M\to G} \cdot \xi_M^{(i,k)},$$

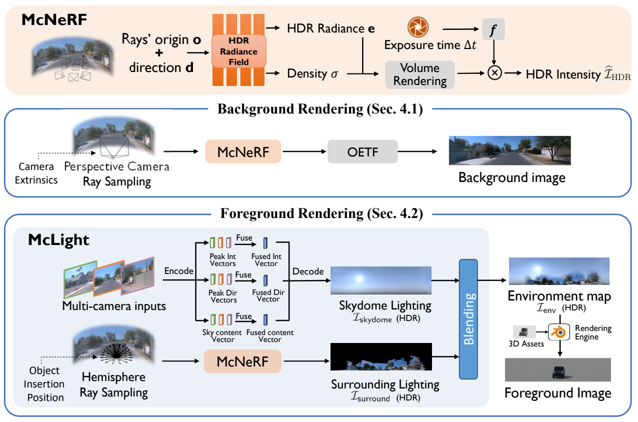
where $\xi_M^{(i,k)}$ denotes the recalibrated camera pose in the Metashape's unified spatial coordinate space, and $T_{M\to G}$ is the transformation from the Metashape's coordinate space to the vehicle's global coordinate space. After alignment, the pose noise can be significantly reduced. Then, the aligned camera pose $\widehat{\xi}^{(i,t)}$ can be used to generate the origins and directions of rays for McNeRF, enabling high-fidelity rendering. The aligned pose can also facilitate the foreground rendering agent's operations.

Brightness-consistent rendering. The exposure times of cameras can differ substantially, causing significant brightness differences across images, hindering the NeRF training. As shown in Figure 4, McNeRF, addresses this by incorporating exposure times into HDR radiance fields, prompting brightness consistency.

We adopt F2-NeRF Wang et al. (2023a) as our backbone model to handle the unbounded scene, sampling K points along the ray $\mathbf{r}$ and estimating each point's HDR radiance $\mathbf{e}_k$ and density σ_k. The HDR light intensity is then calculated as:

$$\widehat{\mathcal{I}}_{\mathrm{HDR}}(\mathbf{r}) = f(\Delta t) \cdot \sum_{k=1}^{K} T_k \alpha_k \mathbf{e}_k, \tag{1}$$

where $\alpha_k = 1 - \exp(-\sigma_k \delta_i)$ is the opacity, δ_i is the point sampling interval, $T_k = \prod_{i=0}^{k-1}(1-\alpha_i)$ is the accumulated transmittance and Δt is the exposure time. The normalization function $f(\Delta t) = 1 + \epsilon(\Delta t - \mu)/\sigma$ is designed to stabilize training, where ϵ is a hyperparameter for scaling, μ and σ are the mean and standard deviation of the exposure times of all images, respectively.



McNeRF
Rays' origin $\mathbf{o}$ + direction $\mathbf{d}$
HDR Radiance Field
HDR Radiance $\mathbf{e}$
Density σ
Exposure time Δt
f
Volume Rendering
HDR Intensity $\widehat{\mathcal{I}}_{\mathrm{HDR}}$
Background Rendering (Sec. 4.1)
Camera Extrinsics
Perspective Camera Ray Sampling
McNeRF
OETF
Background image
Foreground Rendering (Sec. 4.2)
McLight
Multi-camera inputs
Encode
Peak Int Vectors
Fuse
Fused Int Vector
Peak Dir Vectors
Fused Dir Vector
Sky content Vector
Fused content Vector
Decode
Skydome Lighting $\mathcal{I}_{\mathrm{skydome}}$ (HDR)
Object Insertion Position
Hemisphere Ray Sampling
McNeRF
Surrounding Lighting $\mathcal{I}_{\mathrm{surround}}$ (HDR)
Blending
Environment map $\mathcal{I}_{\mathrm{env}}$ (HDR)
3D Assets
Rendering Engine
Foreground Image

Figure 4: Rendering framework. The main components include McNeRF and McLight. Background rendering uses McNeRF to predict HDR pixel value and convert it to LDR with sRGB OETF. McLight includes a skydome lighting estimation network and adopts McNeRF to generate surrounding lighting.

By predicting scene radiance in HDR and multiplying it by the exposure time, we recover the light intensity received by the sensor and tackle the inconsistent color supervision at the intersections of two camera images with distinct exposure times. Moreover, the HDR light intensity outputted by McNeRF can provide scene-level illumination for foreground object rendering, a topic further discussed in Section 4.2.

To train the rendering network, we enforce the consistency of radiance between the rendered image and the captured image (ground-truth). Given the ground-truth image $\mathcal{I}$, the loss function is then:

$$\mathcal{L} = \frac{1}{|R|} \sum_{\mathbf{r} \in R} \left(\mathrm{OETF}\left(\widehat{\mathcal{I}}_{\mathrm{HDR}}(\mathbf{r}) \right) - \mathcal{I}(\mathbf{r}) \right)^2 ,$$

where R represents the ray set and $\mathrm{OETF}(\cdot)$ is the sRGB opto-electronic transfer function (gamma correction) Commission et al. (1999) that converts HDR light intensity to LDR colors.

4.2 McLight for Foreground Rendering

To enrich the scene's content with substantial digital 3D assets, we employ Blender Community (2018) foreground virtual objects' rendering. A seamless insertion critically depends on accurately estimating the scene's illumination conditions. Thus, as shown in Figure 4, we propose McLight, a novel hybrid lighting estimation consisting of skydome lighting and surrounding lighting.

Skydome lighting estimation. Estimating skydome lighting from images is challenging for restoring accurate sun behavior. To achieve this, we propose a novel residual connection from the estimated peak intensity to the HDR reconstruction to address over-smoothing output. Further, we adopt a self-attention mechanism to fuse multi-camera inputs, capturing complementary visual cues.

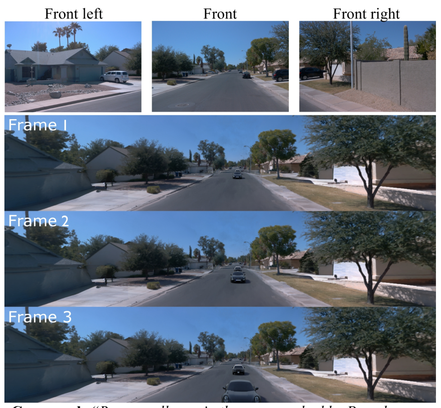
Here we employ a two-stage process. In the first stage, we train an autoencoder to reconstruct the corresponding HDR panorama from an LDR panorama. Following Wang et al. (2022), the encoder transforms the LDR skydome panorama into three intermediate vectors, including the peak direction vector $\mathbf{f}_{\mathrm{dir}} \in \mathbb{R}^3$, the intensity vector $\mathbf{f}_{\mathrm{int}} \in \mathbb{R}^3_+$, and the sky content vector $\mathbf{f}_{\mathrm{content}} \in \mathbb{R}^{64}$. However, as HDR intensity behaves like an impulse response at its peak position, with pixel values thousands of times higher than its neighbors, it is difficult for the decoder to recover such patterns. To tackle this, we design a residual connection that injects $\mathbf{f}_{\mathrm{int}}$ into the decoded HDR panorama with a spherical Gaussian lobe attenuation. This explicitly restores the peak intensity of the sun in the reconstructed HDR panorama, allowing us to render strong shadows for virtual objects.

In the second stage, we train an image encoder and a multi-camera fusion module built upon the pretrained decoder from the first stage. Specifically, for images from each camera, a shared image encoder predicts the peak direction vector $\mathbf{f}^{(i)}_{\mathrm{dir}}$, the intensity vector $\mathbf{f}^{(i)}_{\mathrm{int}}$, and the sky content vector $\mathbf{f}^{(i)}_{\mathrm{content}}$ for each image $\mathcal{I}^{(i)}$, where i is the camera index. We design the latent vector fusion across the multiple camera views as follows: all $\mathbf{f}^{(i)}_{\mathrm{dir}}$ are aligned to the front-facing view using their extrinsic parameters and averaged to form $\bar{\mathbf{f}}_{\mathrm{dir}}$; all $\mathbf{f}^{(i)}_{\mathrm{int}}$ are averaged to yield $\bar{\mathbf{f}}_{\mathrm{int}}$; all $\mathbf{f}^{(i)}_{\mathrm{content}}$ are integrated into $\bar{\mathbf{f}}_{\mathrm{content}}$ through a self-attention module. Finally, the pretrained decoder reconstructs the HDR skydome image $\mathcal{I}_{\mathrm{skydome}}$ from $\bar{\mathbf{f}}_{\mathrm{dir}}$, $\bar{\mathbf{f}}_{\mathrm{int}}$ and $\bar{\mathbf{f}}_{\mathrm{content}}$.

Compared to alternative approaches Wang et al. (2022); Hold-Geoffroy et al. (2019), our multi-camera sky dome estimation technique accurately reproduces the sun's intensity response behavior at its peak with our residual designs, significantly improving the accuracy and fidelity of the skydome reconstruction.

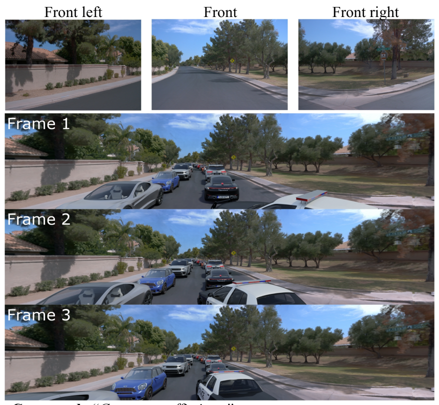
Surrounding lighting estimation. Merely modeling the skydome cannot replicate the complex location-specific lighting effects, like those in the shade with sunlight blocked by trees or buildings. Our McNeRF is capable of storing precise 3D scene information, enabling us to capture the surrounding scene's impact on lighting. This approach facilitates the achievement of spatially-varying lighting estimation. Specifically, we sample the hemisphere rays at the virtual object's position $\mathbf{o}$. The rays' directions, $\mathbf{d}_i, i = 0, 1, \cdots, h \times w$, are aligned with pixel coordinates on a unit sphere using equirectangular projection from an environment map, where h and w are map's height and width. With the ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}_i$, we query our McNeRF as Equation 1 to obtain HDR surrounding lighting $\mathcal{I}_{\mathrm{surround}}(\mathbf{o}, \mathbf{d}_i)$. The surrounding lighting estimation reconstructs complex environmental lighting, achieving a spatially varying effect and high consistency with the background.

Blending. We blend the HDR intensity value from the skydome and surrounding lighting by transmittance of the final sampling point from McNeRF. The idea is that the rays emitted outside the radiance fields will definitely hit the skydome. Given the direction $\mathbf{d}_i$, we retrieve the skydome's



Command*: "Remove all cars in the scene and add a Porsche driving the wrong way toward me fast. Additionally, add a police car also driving the wrong way and chasing behind the Porsche. The view should be moved 5 meters ahead and 0.5 meters above."*

Figure 5: Editing result under complex command.



Command*: "Create a traffic jam."*

Figure 6: Editing result under abstract command.

intensity $\mathcal{I}_{\text{skydome}}(\mathbf{d}_i)$ with equirectangular projection. The final HDR light intensity $\mathcal{I}_{\text{env}}(\mathbf{o}, \mathbf{d}_i)$ is a combination of scene and skydome:

$$\mathcal{I}_{\text{env}}(\mathbf{o}, \mathbf{d}_i) = \mathcal{I}_{\text{surround}}(\mathbf{o}, \mathbf{d}_i) + T_K \mathcal{I}_{\text{skydome}}(\mathbf{d}_i),$$

where T_K is the last sampling point's transmittance.

McLight offers two main advantages: i) it explicitly recovers the illuminance behavior at the peak and use complementary information from multiple cameras to restore accurate skydome; and ii) it enables location-specific lighting with consideration of complex scene structures.

5 Experimental Results

5.1 Datasets and implementation details

We demonstrate a variety of results mainly on the Waymo Open Dataset Sun et al. (2020a), which contains high-quality multi-camera images and the corresponding calibrations. For McLight skydome estimation, we collect 449 HDRIs from online HDRI databases for the autoencoder training and use HoliCity Zhou et al. (2020), a street view panorama dataset for the second stage.

In our experiment, we use front, left front, and right front cameras in each frame. During the rendering process, we choose 40 frames per scene at a 10Hz sampling rate, totaling 120 images. We evenly select $1/8$ of these as the test set, with the remainder used for training. The input images are used at the dataset's initial resolution of 1920×1280; we employ GPT-4 as the LLMs in all of our experiments; see more experiment details in the Appendix.A.

5.2 System results

Editing via language commands. We select three representative commands to demonstrate the editing results. All of the results demonstrate we achieve photo-realistic wide angle results, thanks to McNeRF and McLight.

Mixed and complex command. We send the system with a mixed and complex command, implying that a police car is chasing a wrong-way racer. The target scene, command and the result are shown in Figure 5. We see that i) every requirement in the complex command is accurately executed thanks to our multi-agent collaboration design; ii) this command successfully simulates one rare but dangerous driving condition, which is significant in accident testing.

Highly abstract command. The second type is a highly abstract command. The inputs and results are presented in Figure 6. We see that i) this highly abstract command is hard to decompose by sentence division but still can be correctly executed by our method, and ii) our 3D asset bank offers a large variety of objects for addition.

Multi-round command. We also perform a multi-round chat with our system, and the commands in different rounds exist context dependencies. The final results are shown in Figure 7. We see that i) our system is well-equipped to handle multi-round commands and execute the commands in each

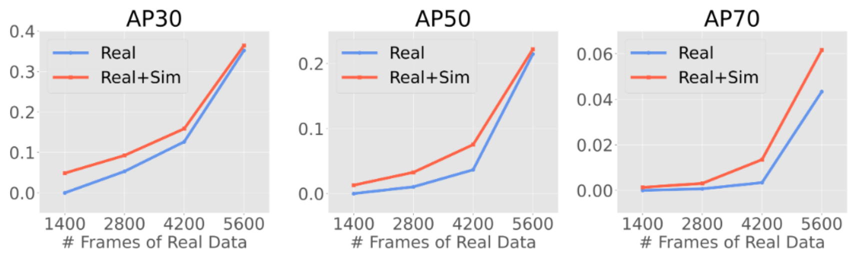
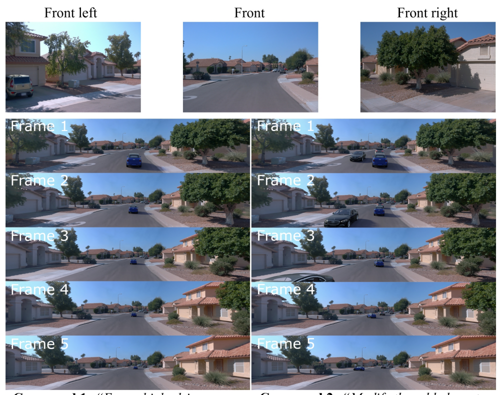


Figure 8: Comparison of detection performance w/o and with our simulated data under different amounts of real data during training.

round precisely; ii) our system can handle the context dependencies across different rounds thanks to the recording ability of the project manager agent.

3D detection with simulation data. We validate the benefits of our simulation as data augmentation for a downstream 3D object detection task on Waymo Open Dataset Sun et al. (2020b). We simulate 1960 frames, derived from scenes in the training dataset. In the simulation, cars with various types, locations, and orientations are incorporated. The detection model adopts Lift-Splat Philion & Fidler (2020). Figure 8 shows detection performances with and without fixed augmentation under various amounts of real data. We see that i) a significant and consistent improvement across different data sizes is achieved; ii) when real data is limited, our simulation notably aids in rough detection (AP30); iii) when the amount of real data increases, our simulation further significantly improves fine-grained detection (AP70), reflecting the high-quality of our simulation.



Command 1: *"Ego vehicle drives ahead slowly. Add a car to the close front that is moving ahead."*

Command 2: *"Modify the added car to turn left. Add a Chevrolet to the front of the added car. Add another vehicle to the left of the added Mini driving toward me."*

Figure 7: Editing result under multi-round commands.

5.3 Multi-agent collaboration.

We evaluated the effectiveness of the multi-agent collaboration by checking whether the command is successfully executed in Table 2.

In scenarios without multi-agent collaboration, all operations are executed by a single LLM agent. We see that a single LLM agent leads to notably lower execution accuracy across all categories due to process limitations. In contrast, the collaborative multi-agent approach can manage most commands, attributed to its task division and agent role specificity.

Multi-agent	Language command category				
collaboration	Deletion	Addition	View change	Revision	Abstract
	0.617	0.383	0.717	0.367	0.216
✓	**0.983**	**0.867**	**0.967**	**0.917**	**0.883**

Table 2: The accuracy (%) of task completion by LLM without and with multi-agent collaboration.

6 Conclusions and Limitations

This paper introduces *ChatSim*, the first system for editing 3D driving scene simulations via language commands and realistic rendering with import of external digital assets. To effectively execute user commands, *ChatSim* adopts an LLM-agent collaboration workflow. To promote photo-realistic simulation, we propose McNeRF and McLight for background and foreground rendering, respectively, accommodating multi-camera inputs. Experiments show that *ChatSim* successfully simulates customized data via multiple kinds of language commands, achieving high-quality, photo-realistic outcomes. **In future**, we plan to integrate more background editing functionalities to *ChatSim*, such as weather changes.

REFERENCES

51Sim-One. https://wdp.51aes.com/news/27.

LLC Agisoft. Metashape–photogrammetric processing of digital images and 3d spatial data generation, 2019.

Elif Akata, Lion Schulz, Julian Coda-Forno, Seong Joon Oh, Matthias Bethge, and Eric Schulz. Playing repeated games with large language models. *arXiv preprint arXiv:2305.16867*, 2023.

Alexander Amini, Igor Gilitschenski, Jacob Phillips, Julia Moseyko, Rohan Banerjee, Sertac Karaman, and Daniela Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, 5(2):1143–1150, 2020.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Natalie M Banta. Property interests in digital assets: The rise of digital feudalism. *Cardozo L. Rev.*, 38:1099, 2016.

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5855–5864, 2021.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5470–5479, 2022.

Luca Bergamini, Yawei Ye, Oliver Scheel, Long Chen, Chih Hu, Luca Del Pero, Błażej Osiński, Hugo Grimmett, and Peter Ondruska. Simnet: Learning reactive self-driving simulations from real-world observations. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5119–5125. IEEE, 2021.

Mark Boss, Varun Jampani, Kihwan Kim, Hendrik Lensch, and Jan Kautz. Two-shot spatially-varying brdf and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3982–3991, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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.

Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8748–8757, 2019.

ChatGPT. https://openai.com/blog/chatgpt.

Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pp. 2722–2730, 2015.

Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2147–2156, 2016.

Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1907–1915, 2017.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

International Electrotechnical Commission et al. Iec 61966-2-1: 1999 multimedia systems and equipment-colour measurement and management- part 2-1: Colour management- default rgb colour space- srgb, 1999.

Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL `http://www.blender.org`.

Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pp. 109–118, 2001.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Openscanerio Editor. https://github.com/ebadi/openscenarioeditor.

Unreal Engine. https://www.unrealengine.com/.

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. *arXiv preprint arXiv:2310.02601*, 2023.

Mathieu Garon, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, and Jean-François Lalonde. Fast spatially-varying indoor lighting estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6908–6917, 2019.

Jianfei Guo, Nianchen Deng, Xinyang Li, Yeqi Bai, Botian Shi, Chiyu Wang, Chenjing Ding, Dongliang Wang, and Yikang Li. Streetsurf: Extending multi-view implicit surface reconstruction to street views. *arXiv preprint arXiv:2306.04988*, 2023.

Rui Hao, Linmei Hu, Weijian Qi, Qingliu Wu, Yirui Zhang, and Liqiang Nie. Chatllm network: More brains, more intelligence. *arXiv preprint arXiv:2304.12998*, 2023.

Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-François Lalonde. Deep outdoor illumination estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7312–7321, 2017.

Yannick Hold-Geoffroy, Akshaya Athawale, and Jean-François Lalonde. Deep sky modeling for single image outdoor lighting estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6927–6935, 2019.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12871–12881, 2022.

Jean-François Lalonde and Iain Matthews. Lighting estimation in outdoor image collections. In *2014 2nd international conference on 3D vision*, volume 1, pp. 131–138. IEEE, 2014.

Jean-François Lalonde, Srinivasa G Narasimhan, and Alexei A Efros. What do the sun and the sky tell us about the camera? *International Journal of Computer Vision*, 88:24–51, 2010.

Chloe LeGendre, Wan-Chun Ma, Graham Fyffe, John Flynn, Laurent Charbonnel, Jay Busch, and Paul Debevec. Deeplight: Learning illumination for unconstrained mobile mixed reality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5918–5928, 2019.

Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In *European Conference on Computer Vision*, pp. 644–660. Springer, 2020.

Xiaofan Li, Yifu Zhang, and Xiaoqing Ye. Drivingdiffusion: Layout-guided multi-view driving scene video generation with latent diffusion model. *arXiv preprint arXiv:2310.07771*, 2023a.

Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

Zhuopeng Li, Lu Li, and Jianke Zhu. Read: Large-scale neural scene rendering for autonomous driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 1522–1529, 2023b.

Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. Maptr: Structured modeling and learning for online vectorized hd map construction. *arXiv preprint arXiv:2208.14437*, 2022.

Bencheng Liao, Shaoyu Chen, Yunchi Zhang, Bo Jiang, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Maptrv2: An end-to-end framework for online vectorized hd map construction. *arXiv preprint arXiv:2308.05736*, 2023.

Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela L Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2774–2781. IEEE, 2023.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106, 2021.

Milan Miric, Kevin J Boudreau, and Lars Bo Jeppesen. Protecting their digital assets: The use of formal & informal appropriability strategies by app developers. *Research Policy*, 48(8):103738, 2019.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

OpenAI. Gpt-4 technical report. 2023.

Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2856–2865, 2021.

Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18419–18429, 2022.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pp. 194–210. Springer, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*, pp. 1–6. IEEE, 2020.

Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W Jacobs, and Jan Kautz. Neural inverse rendering of an indoor scene from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8598–8607, 2019.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics: Results of the 11th International Conference*, pp. 621–635. Springer, 2018.

Gowri Somanath and Daniel Kurz. Hdr environment map estimation for real-time augmented reality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11298–11306, 2021.

Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5459–5469, 2022.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020a.

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, 2020b.

Alexander Swerdlow, Runsheng Xu, and Bolei Zhou. Street-view image generation from a bird's-eye view layout. *arXiv preprint arXiv:2301.04634*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12375–12385, 2023.

Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4150–4159, 2023a.

Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, and Jiwen Lu. Drivedreamer: Towards real-world-driven world models for autonomous driving. *arXiv preprint arXiv:2309.09777*, 2023b.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. *arXiv preprint arXiv:2307.05300*, 2023c.

Zian Wang, Wenzheng Chen, David Acuna, Jan Kautz, and Sanja Fidler. Neural light field estimation for street scenes with differentiable virtual object insertion. In *European Conference on Computer Vision*, pp. 380–397. Springer, 2022.

Bichen Wu, Forrest Iandola, Peter H Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 129–137, 2017.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023a.

Zirui Wu, Tianyu Liu, Liyi Luo, Zhide Zhong, Jianteng Chen, Hongmin Xiao, Chao Hou, Haozhe Lou, Yuantao Chen, Runyi Yang, Yuxin Huang, Xiaoyu Ye, Zike Yan, Yongliang Shi, Yiyi Liao, and Hao Zhao. Mars: An instance-aware, modular and realistic simulator for autonomous driving. *CICAI*, 2023b.

Ziyang Xie, Junge Zhang, Wenye Li, Feihu Zhang, and Li Zhang. S-nerf: Neural radiance fields for street views. *arXiv preprint arXiv:2303.00749*, 2023.

Yinda Xu and Lidong Yu. Drl-based trajectory tracking for motion-related modules in autonomous driving. *arXiv preprint arXiv:2308.15991*, 2023.

Kairui Yang, Enhui Ma, Jibin Peng, Qing Guo, Di Lin, and Kaicheng Yu. Bevcontrol: Accurately controlling street-view elements with multi-perspective consistency via bev sketch layout. *arXiv preprint arXiv:2308.01661*, 2023a.

Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1389–1399, 2023b.

Zhenpei Yang, Yuning Chai, Dragomir Anguelov, Yin Zhou, Pei Sun, Dumitru Erhan, Sean Rafferty, and Henrik Kretzschmar. Surfelgan: Synthesizing realistic sensor data for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11118–11127, 2020.

Jinsong Zhang, Kalyan Sunkavalli, Yannick Hold-Geoffroy, Sunil Hadap, Jonathan Eisenman, and Jean-François Lalonde. All-weather deep outdoor lighting estimation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 10158–10166, 2019.

Yichao Zhou, Jingwei Huang, Xili Dai, Linjie Luo, Zhili Chen, and Yi Ma. HoliCity: A city-scale data platform for learning holistic 3D structures. 2020. arXiv:2008.03286 [cs.CV].

Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, et al. Mindstorms in natural language-based societies of mind. *arXiv preprint arXiv:2305.17066*, 2023.

A More Experiments Results

A.1 Rendering results

Background rendering. We compare our McNeRF with several other state-of-the-art methods on the background novel view synthesis task. We perform reconstruction and rendering on 32 selected scenes. Table 3 shows the quantitative results comparison on three metrics: PSNR, SSIM, and LPIPS. We see that i) McNeRF achieves SoTA performance on all three metrics, significantly outperforming other baselines; ii) McNeRF has a fast inference speed, enabling quick responses to image rendering.

Figure 9 in Appendix demonstrates qualitative comparisons between other methods and ours. We see that existing NeRF methods do not consider the exposure time, leading to noticeable changes in brightness at the junctions of different cameras in the image, as well as an overall inconsistency in exposure across the wide-angle view. Our method can make the brightness of the entire image more consistent.

Methods	PSNR↑	SSIM↑	LPIPS↓	Inf. time (s)↓
DVGO Sun et al. (2022)	23.57	0.770	0.508	7.7
Mip-NeRF360 Barron et al. (2022)	24.40	0.754	0.528	101.8
S-NeRF Xie et al. (2023)	24.71	0.759	0.519	114.5
F2NeRF Wang et al. (2023a)	23.26	0.773	0.439	2.4
Ours w/o alignment	23.32	0.776	0.437	2.5
Ours w/o exposure	25.18	0.819	0.381	2.4
McNeRF (Ours)	**25.82**	**0.822**	**0.378**	2.5

Table 3: Background rendering performance evaluation.

Foreground rendering. We compare our McLight with the other two SoTA methods Wang et al. (2022); Hold-Geoffroy et al. (2019). Table 4 shows the comparison of relative intensity(log 10) error on our HDRI dataset, angular error on HoliCity Zhou et al. (2020), and user study. We see that McLight achieves more accurate peak behavior and receives noticeably higher user preferences.

Method	Peak Intensity(log10) Error		Peak Angular Error (deg)		User study(%) ↑
	Mean ↓	Median ↓	Mean ↓	Median ↓	
Hold-Geoffroy et al. Hold-Geoffroy et al. (2019)	0.899	0.975	48.4	51.6	19.5
Wang et al. Wang et al. (2022)	0.590	0.628	33.5	29.4	37.3
McLight (Ours)	**0.449**	**0.270**	**32.3**	**26.5**	**43.1**

Table 4: Lighting estimation comparison.

Figure 10 in Appendix shows the visualizations of vehicle insertion. The vehicles added through McLight feature significantly more realistic reflections and strong shadows consistent with the scene.

A.2 Vehicle motion.

As shown in Table 5, we compare the motion generation method from user commands with two of our designed baselines: 1. GPT2Motion, which directly uses LLM to return the motion coordinates; 2. GPT2Code, which first generates code using LLM and executes it to obtain the vehicle motion. We validate multiple actions in multiple scenarios and report the user study result. The user study is to determine if the generated motions matched the command intents and fitted with the lane map. We see that our method demonstrated a significant advantage in generating motions from language commands. Additionally, it maintained a high rate of keeping the trajectories within the lane boundaries.

Methods	Straight	Left Turn	Right Turn	Speed	Within-road
GPT2Code	0.738	0.559	0.536	0.893	0.214
GPT2Motion	0.595	0.119	0.167	0.345	0.277
Ours	**0.988**	**0.940**	**0.976**	**0.952**	**1.000**

Table 5: Comparison with motion generation from text methods.

Our motion generation design returns accurate and compliant results which can not be easily completed with direct methods.

B Explanation of Table 2 in Main Text

Table 2 evaluates the execution success rate of commands of 5 instruction categories across 4 different driving sequences. For each category, the accuracy is measured as the average success rate across 3 trials of 15 commands that are specifically designed for this category. Each trial is deemed successfully executed if the LLM-agent(s) accurately perform the required operations, including setting correct configurations and parameter values.

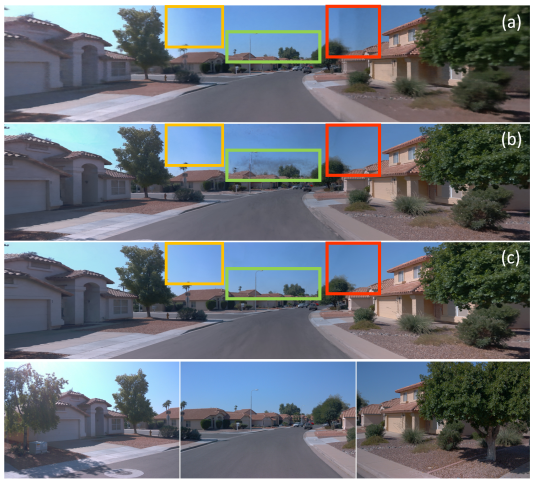


Figure 9: Comparisons of wide-angle images generation. (a) S-NeRF.(b) F2NeRF. (c) McNeRF (Ours). Last row: target images.

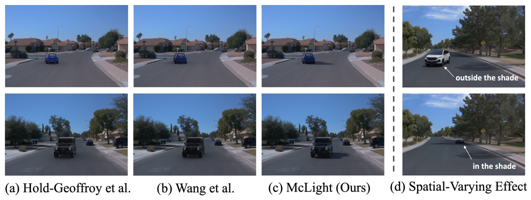


Figure 10: Comparison with different lighting estimation methods.

C LLM-Agents Details

C.1 Agent Implement Details

LLM-Agents consist of their LLM (Large Language Model) component and corresponding functionalities. All experiments utilize the GPT-4 APIOpenAI (2023) to implement the LLM part. In each agent's prompt, there are elements involving the agent's function, the definition of actions that the agent needs to perform, the definition of information inputted to the agent, and the definition of outputs required from the agent. To facilitate the integration of Python code and ensure stable calls, the LLM part is required to return information in the format of a JSON dictionary. Additionally, each LLM part's prompt includes some examples, which contain inputs for certain scenarios and the corresponding expected outputs. If the input command does not contain the information of the keys of the output JSON dictionary, a default one will be filled in the dictionary. The parameters related to the GPT-4 API are all set to the official default values.

Note that, for supporting modification operations during multi-round commands, the 3D asset management agent, the vehicle motion agent, and the vehicle deleting agent have the ability to modify the information of already added or deleted cars.

C.2 Reasoning Processes

This section describes the natural language reasoning processes for the three cases presented in Section 5.2 of the main text.

Mixed and complex command. The initial input command is: *"Remove all cars in the scene and add a Porsche driving the wrong way toward me fast. Additionally, add a police car also driving the wrong way and chasing behind the Porsche. The view should be moved 5 meters ahead and 0.5 meters above."* The command is decoupled by the project manager agent as following commands: 1.*"Remove all cars."*; 2. *"Add a Porsche driving the wrong way toward me fast."*; 3. *"Add a police car also driving the wrong way and chasing behind the Porsche."*; 4. *"The view should be moved 5 meters ahead and 0.5 meters above."*. The *"Remove all cars."* command is distributed to the vehicle deleting agent, and then the agent finds the 3D boxes of all cars and applies the inpainting function

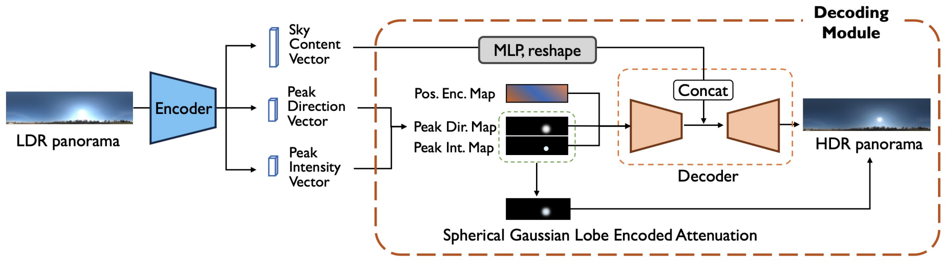


Figure 11: LDR to HDR reconstruction network. We add an explicit spherical Gaussian lobe encoded attenuation to overcome the over-smoothness in the decoded HDR panorama. It effectively ensures that the sun's intensity significantly exceeds that of surrounding pixels, rendering strong shadow effects for inserted objects.

for the removal operation. *"Add a Porsche driving the wrong way toward me fast."* command is distributed to the 3D asset management agent for selecting the proper 3D asset. This command will also be distributed to the vehicle motion agent, which utilizes the key information in the command including "wrong way", "toward me" and "fast" to choose the appropriate start and end points and generate the motion with the motion generation function. *"Add a police car also driving the wrong way and chasing behind the Porsche."* command will also be executed in the same way as the former operation. This command mentions the information of the added car, and the added car's information has been memorized by the project manager. This information is offered to the vehicle motion agent for determining the added police car's location. *"The view should be moved 5 meters ahead and 0.5 meters above."* command is distributed to the view adjustment agent. The view adjustment agent returns the adjustment information of extrinsic as configuration, and calls the function to change the extrinsics to achieve view adjustment. Finally, background rendering and foreground rendering agents are required to generate the background and foreground results according to the information returned by the other agents, and the results are composed as the final outputs.

Highly abstract command. The initial input command is: *"Create a traffic jam."* The project manager agent analyzes the command and decouples it as multiple repeats of car addition. These addition commands are processed by the 3D asset management agent and vehicle motion agent successively and are rendered by the foreground rendering agent. Combined with the rendered results from the background rendering agent, we can get the final outputs.

Multi-round command. The first initial command is: *"Ego vehicle drives ahead slowly. Add a car to the close front that is moving ahead."* The command is decoupled by the project manager agent as 1: *"Ego vehicle drives ahead slowly."*; 2: *"Add a car to the close front that is moving ahead."*. The first sub-command is distributed to view the adjustment agent, and the agent generates the extrinsics that represent moving ahead slowly. The second sub-command is executed as the process introduced above.

The second initial command is: *"Modify the added car to turn left. Add a Chevrolet to the front of the added car. Add another vehicle to the left of the added Mini driving toward me."* The command is decoupled by the project manager agent as 1: *"Modify the added car to turn left."*; 2: *"Add a Chevrolet to the front of the added car."*; 3: *"Add another vehicle to the left of the added Mini driving toward me."* The first sub-command is distributed to the vehicle motion agent, which generates new motion based on the command for the determined added car. The following two sub-commands are executed in the same way as mentioned in the paragraphs above. Compositing the outputs of background rendering and foreground rendering agents can get the final outputs.

D Skydome Lighting Estimation Details

D.1 HDRI dataset

We collect 449 high-quality outdoor panorama HDRIs from Poly Heaven Website. These HDRIs are all licensed as CC0. We randomly selected 357 HDRIs for the training set and the remaining for the test set. A script for downloading these HDRIs will be available.

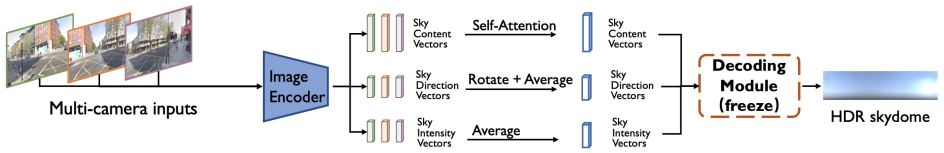


Figure 12: Reconstructing HDR skydome from multi-camera images. Training on HoliCity Zhou et al. (2020) dataset.

D.2 LDR to HDR Skydome Reconstruction

In this step, we utilize our HDRI dataset to train an LDR to HDR autoencoder with the aim of converting the skydome into a compact feature representation. We use the sRGB opto-electronic transfer function (also known as gamma correction) to get the LDR sky panorama, and follow Wang et al. (2022) to transform the LDR sky panorama to 3 intermediate vectors, including the sky content vector $\mathbf{f}_{\text{content}} \in \mathbb{R}^{64}$, the peak direction vector $\mathbf{f}_{\text{dir}} \in \mathbb{R}^3$ and the intensity vector $\mathbf{f}_{\text{int}} \in \mathbb{R}^3_+$. In the process of converting intermediate vectors into a reconstructed HDR sky panorama, we construct the peak direction map $\mathbf{M}_{\text{dir}}$, the peak intensity map $\mathbf{M}_{\text{int}}$ and the positional encoding map $\mathbf{M}_{\text{pe}}$.

Peak direction map ($\mathbf{M}_{\text{dir}}$): For each pixel in $\mathbf{M}_{\text{dir}}$, we calculate the peak direction embedding. This calculation utilizes a spherical Gaussian lobe, formulated as $\mathbf{M}_{\text{dir}}(\mathbf{u}) = e^{100*(\mathbf{u}\cdot\mathbf{f}_{dir}-1)}$, where $\mathbf{f}_{\text{dir}}$ denotes the peak direction vector. This map is represented in $\mathbb{R}^{H\times W\times 1}$.

Peak intensity map ($\mathbf{M}_{\text{int}}$): Each pixel in this map is determined based on its corresponding value in the peak direction map. Specifically, for a given direction $\mathbf{u}$, if $\mathbf{M}_{\text{dir}}(\mathbf{u}) > 0.9$, then $\mathbf{M}_{\text{int}}(\mathbf{u})$ is assigned the value of $\mathbf{f}_{\text{int}}$. If not, $\mathbf{M}_{\text{int}}(\mathbf{u})$ is set to zero. This map is represented in $\mathbb{R}^{H\times W\times 3}_+$.

Positional encoding map ($\mathbf{M}_{\text{pe}}$): This map encodes the direction vector of each pixel, determined through equirectangular projection, thus contributing to the accurate reconstruction of the HDR sky panorama. It is defined in $\mathbb{R}^{H\times W\times 3}$.

The input of the decoder $\mathbf{M}_{\text{input}}$ is a concatenation of $\mathbf{M}_{\text{pe}}, \mathbf{M}_{\text{dir}}$ and $\mathbf{M}_{\text{int}}$. We use a 2D UNet to decode the concatenated input map to the HDR sky panorama. For sky content vector $\mathbf{f}_{\text{content}}$, we use an MLP to increase its feature dimension, reshape it to a 2D feature map, and concatenate it with the intermediate features at the bottleneck of the UNet. This concatenated feature will be further decoded to the HDR sky panorama.

In the context of HDR imaging, the intensity of the peak often exhibits characteristics akin to an impulse response, displaying pixel values that are significantly elevated by orders of magnitude in comparison to adjacent pixels. This presents a substantial challenge for the decoder in accurately recovering these patterns. Thus, we design a residual connection to explicitly inject the peak intensity information into the final HDR sky panorama. Let $\mathbf{M}_{\text{peak}}$ be the product of $\mathbf{M}_{\text{dir}}$ and $\mathbf{M}_{\text{int}}$, representing an attenuation encoded by a spherical Gaussian lobe. In our design, we specifically substitute the decoded HDR sky panorama at the peak position with $\mathbf{M}_{\text{peak}}$. This substitution is applied where the value of $\mathbf{M}_{\text{int}}(\mathbf{u})$ is non-zero, ensuring that the peak position in the HDR sky panorama is accurately represented by $\mathbf{M}_{\text{peak}}$. This makes a significant difference between us and Wang et al. (2022). Accurate and strong peak intensity can generate very strong shadow effects, resulting in better rendering realism. See Figure 11.

To train the LDR to HDR skydome reconstruction, we computer the ground truth peak direction $\mathbf{f}^{\text{gt}}_{\text{dir}}$ and peak intensity $\mathbf{f}^{\text{gt}}_{\text{int}}$ from the HDR ground-truth. During the network training process, we employ four losses for supervision. These losses are as follows: peak direction loss L_{dir}, which measures the L1 angular error of the peak direction vectors; peak intensity loss L_{int}, which quantifies the log-encoded L2 error of the peak intensity vectors; HDR reconstruction loss $L_{\text{hdr-recon}}$, which evaluates the log-encoded L2 error between the reconstructed HDR output and the ground truth HDR data; LDR reconstruction loss $L_{\text{ldr-recon}}$, which is calculated as the L1 error between the input LDR sky panorama and the gamma-corrected HDR reconstruction.

The total loss is $L_{\text{total}} = \lambda_1 L_{\text{dir}} + \lambda_2 L_{\text{int}} + \lambda_3 L_{\text{hdr-recon}} + \lambda_4 L_{\text{ldr-recon}}$, where $\lambda_1 = 1, \lambda_2 = 0.1, \lambda_3 = 2$ and $\lambda_4 = 0.2$.

Data augmentation methods, including rotation, flipping, exposure adjustment and white balance adjustment, are implemented to enrich the training data. Noticing a strong white balance inaccuracy (the color temperature is too high) in the image data from Waymo Open Dataset Sun et al. (2020b), we augment the HDRI with corresponding white balance adjustment. The blue channel is randomly enlarged by 1.2-1.3 times, and the red channel is randomly reduced by 1.2-1.3 times.

D.3 Predict HDR Skydome from Multi-Camera Images

This step involves estimating skydome lighting from multi-camera images collected by the vehicle. The core idea is to estimate intermediate features from multiple views and restore the skydome lighting using the well-trained HDR reconstruction decoding module. We emphasize the fusion of intermediate features from multiple cameras to get a complementary and comprehensive prediction for the skydome lighting.

Multi-camera image data will first go through a shared image encoder to predict the peak direction vector $\mathbf{f}_{\text{dir}}^{(i)}$, the intensity vector $\mathbf{f}_{\text{int}}^{(i)}$, and the sky content vector $\mathbf{f}_{\text{content}}^{(i)}$ for each image $\mathcal{I}^{(i)}$, where i is the camera index. For those vectors from N cameras, we fuse all the features in the following strategy:

We transform $\mathbf{f}_{\text{dir}}^{(i)}, i = 1, 2, ..., N$ to the front-facing view using their extrinsic parameters and averaged the rotated direction vector to $\bar{\mathbf{f}}_{\text{dir}}$; we average $\mathbf{f}_{\text{int}}^{(i)}, i = 1, 2, ..., N$ to $\bar{\mathbf{f}}_{\text{int}}$; we utilize the attention mechanism to fuse sky content vectors as $\bar{\mathbf{f}}_{\text{content}} = \texttt{Attn(q, k, v)}$, where $\texttt{q} = \mathbf{f}_{\text{content}}^{(0)}$, $\texttt{k} = \texttt{v} = \texttt{stack}(\{\mathbf{f}_{\text{content}}^{i}\}_{i=0,1,...,N-1})$. Here index 0 refers to the first (front-facing) view image and $\texttt{Attn}(\cdot,\cdot,\cdot)$ the standard attention operator. Given $\bar{\mathbf{f}}_{\text{dir}}$, $\bar{\mathbf{f}}_{\text{int}}$, $\bar{\mathbf{f}}_{\text{content}}$, we use the pre-trained decoding module from the previous stage to recover the fused intermediate vectors to HDR panorama. See Figure 12.

Since there is no relevant panoramic data in the autonomous driving dataset for supervision, We use HoliCity Zhou et al. (2020) to simulate multi-camera images. Based on the arrangement and FOV of the three forward-facing cameras on the Waymo vehicle Sun et al. (2020b), we cropped the corresponding image from the HoliCity panorama as the model inputs. To supervise the learning of the image encoder, we use the LDR to HDR reconstruction network from the previous stage to generate pseudo peak intensity vector GT, peak direction vector GT, sky content vector GT, and HDR skydome GT.

We apply five losses to supervise the network during training. These losses are as follows: the peak direction loss L_{dir}, which measures the L1 angular error of the fused peak direction vector; the peak intensity loss L_{int}, which calculates the log-encoded L2 error of the fused peak intensity vectors; the sky content loss L_{content}, which evaluates the L1 error of the fused sky content vectors; the HDR reconstruction loss $L_{\text{hdr-recon}}$ with log-encoded L2 error; the LDR reconstruction $L_{\text{ldr-recon}}$ with L1 error.

The total loss is $L_{\text{total}} = \lambda_1 L_{\text{dir}} + \lambda_2 L_{\text{int}} + \lambda_3 L_{\text{content}} + \lambda_4 L_{\text{hdr-recon}} + \lambda_5 L_{\text{ldr-recon}}$, where $\lambda_1 = 0.5, \lambda_2 = 0.25, \lambda_3 = 0.005, \lambda_4 = 0.1$ and $\lambda_5 = 0.2$.

E 3D Asset Bank

To ensure ease of access and modification of 3D assets, we normalize our Blender models within their Blender files using the following procedure:

1. We ensure that the model has accurate physical dimensions in the unit of meter.
2. The origin of the car model is set at the middle of the bottom of the car. We position the model at the center of the world coordinate system, ensuring that the car model's origin aligns with the origin of the world coordinate system. The car is oriented to face the positive direction of the x-axis.
3. We uniformly apply the `Principled BSDF` material to the car body, and name the material "car_paint". Prompt that changes the asset's color will affect the "Base Color" attribute of the `Principled BSDF` node.

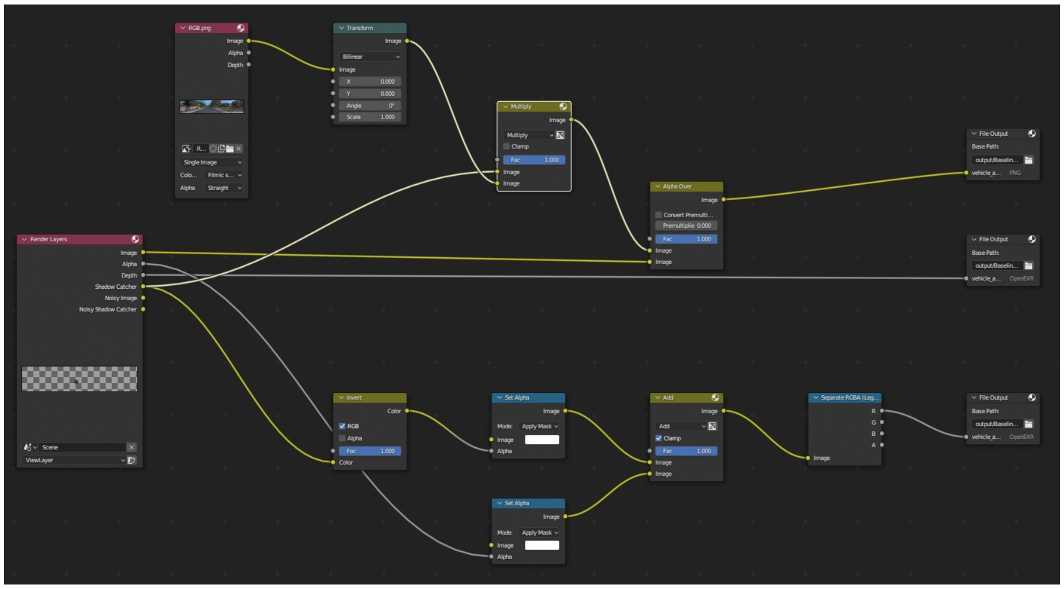


Figure 13: Compositing node graph design in Blender Community (2018)

4. We use the `Join` operator to merge all meshes into one object.

Following the aforementioned approach, we normalize the Blender models collected from the Internet to continuously expand our 3D Asset Bank.

F BLENDER RENDERING DETAILS

We fully implement the Blender rendering workflow using Python scripting, incorporating features such as alpha channel, depth channel, and shadow effect, **all achieved within a single rendering pass**.

1. To get a transparent background, we first enable the `Render Properties - Film - Transparent` option.
2. To get multiple rendering output, we enable the `Combined` pass, `Z` pass and `Shadow Catcher` pass in `View Layer Properties` panel.
3. To render the shadow, we add a very large plane under the car and enable the plane's `Object Properties - Visibility - Mask - Shadow Catcher` option.
4. To obtain scene-related colored shadows, we construct the compositing node graph as Figure 13. This configuration generates the rendered image overlaid on the scene image, along with the accompanying depth information and mask of the vehicle and its corresponding shadow.
5. Using depth information and mask, we can handle the occlusion relationship with the original objects in the scene. We also added a moderate amount of motion blur to the rendered car to match the background.

G MOTION GENERATION DETAILS

The vehicle motion agent creates the initial places and subsequent motions of vehicles following the requests commands. Existing vehicle motion generation methods cannot directly generate motion purely from text and the scene map. Here we elaborate on the details of our text-to-motion methods. Our method consists of two parts: vehicle placement to generate the starting points and vehicle motion planning to generate the subsequent motions.

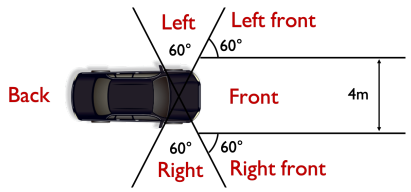


Figure 14: The neighboring area division for vehicle placement.

G.1 Vehicle Placement

We use the language command and the scene map to generate the initial position. The scene map $\mathcal{M}$ follows the lane map form $\mathcal{M} = \{\mathbf{n}_i, i = 1, 2, \cdots, m\}$, where m is the number of lane nodes and the ith lane node $\mathbf{n}_i = (x_{\mathrm{s}}, y_{\mathrm{s}}, x_{\mathrm{e}}, y_{\mathrm{e}}, c_{\mathrm{type}})$ consists of lane starting position $(x_{\mathrm{s}}, y_{\mathrm{s}})$, ending position $(x_{\mathrm{e}}, y_{\mathrm{e}})$ and the lane type c_{type}. The map range is cropped with the range of front 80m, left 20m and right 20m. Generally, we use the lane map from the ground-truth data. If the lane map does not exist, it is applicable to use a lane map estimation method like Liao et al. (2022; 2023) to obtain the lane map.

Given the language command, the LLM first extracts key placement attributes, including vehicle number, distance range, relative direction with the observer and direction of driving, and crazy mode. With these attributes, the role function of placement begins to find suitable lane nodes from the scene map. Here we assume all the placed vehicles are on the centerline of the road. If the distance range $(d_{\min}, d_{\max})$ is identified, the role function selects the lane centerline nodes according to their distance with the ego location. For the relative direction, we divide the ego neighboring area into 6 categories: front, left front, right front, left, right, and back, see Figure 14 for illustration. For the direction of driving, we consider two types: driving close to the ego and driving away from the ego, which determines the left/right side of the vehicle on the road. The crazy mode, which is designed for non-compliant inverse driving behavior, is a bool variable. When it is true, we will inverse the direction of the map (swap the starting and ending point of each lane) for that vehicle to represent inverse driving. We select the matched lane node set and randomly select one lane node from the set. We also consider the conflict of placing vehicles by an iterative approach that incoming vehicles should not overlap with the existing vehicles. After obtaining lane nodes for every vehicle, we set the midpoint of the lane node to be the initial position of a vehicle and the direction of the lane to be the initial heading of the vehicle.

G.2 Vehicle Motion Planning

After obtaining the initial positions, we generate motions in two steps: plan the destination and plan the middle trajectory. We first extract movement attributes including speed, action, interval and time length. Notably, we divide actions into 5 categories: straightforward, turn left, turn right, park, and backward. To obtain the destination, if the action category is straightforward or park, and backward, we directly calculate a raw destination by assuming the car driving following a line with the target speed. Then we find the closest lane node with the raw destination to be the final destination. If the action category is turning left or turning right, we select a set of nodes whose vertical distance with the initial line of heading is in a range (5m-30m) and fit the driving directions (the direction of the line should be away from the starting point). We randomly pick a lane node to be the destination.

To plan the middle trajectory, we use an iterative adjustment approach to make the trajectory match with the map and avoid off-road driving. We first use one cubic Bezier curve to fit the overall trajectory with the condition of starting point, starting direction, ending point and ending direction. The cubic Bezier curve is formulated by

$$\begin{aligned} B(t) =& (1-t)^3 P_0 + 3t(1-t)^2 P_1 \\ &+ 3t^2(1-t)P_2 + t^3 P_3, t \in [0, 1], \end{aligned} \tag{2}$$

where $P_0, P_1, P_2, P_3 \in \mathbb{R}^2$ is the control points that can be solved by given starting point, starting direction, ending point and ending direction. Then to avoid off-road driving of the intermediate trajectory, we adjust the middle coordinate by replacing it with the closest lane node. We split the whole trajectory into two parts with the boundary of the middle coordinate and use one cubic Bezier

Sequence	Start Frame
segment-10247954040621004675_2180_000_2200_000	0
segment-13469905891836363794_4429_660_4449_660	40
segment-14333744981238305769_5658_260_5678_260	40
segment-1172406780360799916_1660_000_1680_000	50
segment-4058410353286511411_3980_000_4000_000	90
segment-10061305430875486848_1080_000_1100_000	30
segment-14869732972903148657_2420_000_2440_000	0
segment-16646360389507147817_3320_000_3340_000	0
segment-13238419657658219864_4630_850_4650_850	0
segment-14424804287031718399_1281_030_1301_030	60
segment-15270638100874320175_2720_000_2740_000	60
segment-15349503153813328111_2160_000_2180_000	100
segment-15868625208244306149_4340_000_4360_000	110
segment-16608525782988721413_100_000_120_000	10
segment-17761959194352517553_5448_420_5468_420	0
segment-3425716115468765803_977_756_997_756	0
segment-3988957004231180266_5566_500_5586_500	0
segment-9385013624094020582_2547_650_2567_650	130
segment-8811210064692949185_3066_770_3086_770	30
segment-10275144660749673822_5755_561_5775_561	0
segment-10676267326664322837_311_180_331_180	100
segment-12879640240483815315_5852_605_5872_605	20
segment-13142190313715360621_3888_090_3908_090	0
segment-13196796799137805454_3036_940_3056_940	70
segment-14348136031422182645_3360_000_3380_000	140
segment-15365821471737026848_1160_000_1180_000	0
segment-16470190748368943792_4369_490_4389_490	0
segment-11379226583756500423_6230_810_6250_810	0
segment-13085453465864374565_2040_000_2060_000	110
segment-14004546003548947884_2331_861_2351_861	0
segment-15221704733958986648_1400_000_1420_000	70
segment-16345319168590318167_1420_000_1440_000	0

Table 6: Information on the selected and trimmed Waymo Open Dataset Sun et al. (2020b). For each sequence, we select 40 frames starting from the Start Frame.

curve to fit each split trajectory. We iteratively repeat the process to represent the planned trajectory by multiple cubic Bezier curves. Finally, to make the planned trajectory fit with vehicle dynamics, we use a trajectory tracking method in Xu & Yu (2023) as post-processing to revise the planned trajectory.

H Background rendering details

H.1 Dataset Selection

For all Waymo Open Dataset Sun et al. (2020b) experiments, we use images captured from three frontal cameras. The details of selection are shown in Table 6. There are 120 images in total for each scenerio.

H.2 Multi-Camera Alignment

This section will introduce the details of our multi-camera alignment algorithm. Let $R_{C_i,t}$ and $T_{C_i,t}$ represents the camera C_i's extrinsic matrix that aligned to vehicle's coordinates at timestamp t. C_0 is the front camera. The superscript (V) and (M) represents the original vehicle's coordinates in autonomous driving dataset and the coordinates under Metashape's unified space. Then the rotation $R_{C_i,t}$ and translation $T_{C_i,t}$ can be calculated as:

$$\begin{aligned} R_{C_i,t} &= R^{(V)}_{C_0,0}(R^{(M)}_{C_0,0})^{-1}R^{(M)}_{C_i,t} \\ T_{C_i,t} &= \frac{R^{(V)}_{C_0,0}(R^{(M)}_{C_0,0})^{-1}(T^{(M)}_{C_i,t} - T^{(M)}_{C_0,0})}{S} + T^{(V)}_{C_0,0}, \end{aligned}$$

where $S = \frac{T^{(M)}_{C_0,1} - T^{(M)}_{C_0,0}}{T^{(V)}_{C_0,1} - T^{(V)}_{C_0,0}}$ is a scaling factor that ensures the aligned space has the same unit length as the real world.

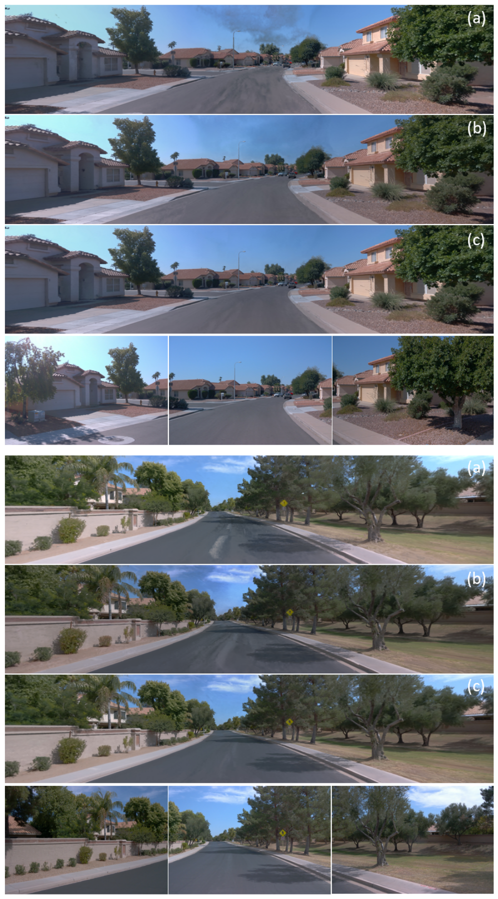


Figure 15: Qualitative ablation of background rendering. (a) McNeRF w/o pose alignment.(b) McNeRF w/o exposure. (c) Full McNeRF. Last row: target images.

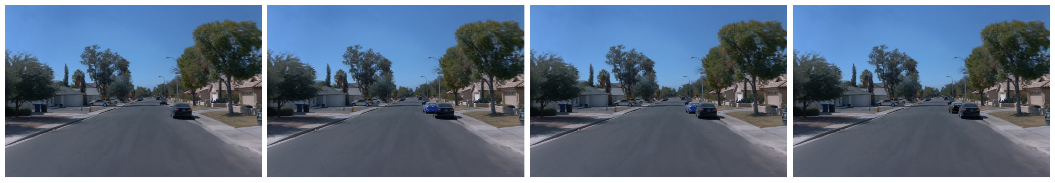

Initial rendered image
Command: *"Add a Mini in front of the right car"*(without occlusion postprocess)
Command: *"Add a Mini in front of the right car"*(with occlusion postprocess)
Command: *"Add a black Mini in front of the right car"*(with occlusion postprocess)

Figure 16: Qualitative result of occlusion postprocess and the color control for added car.

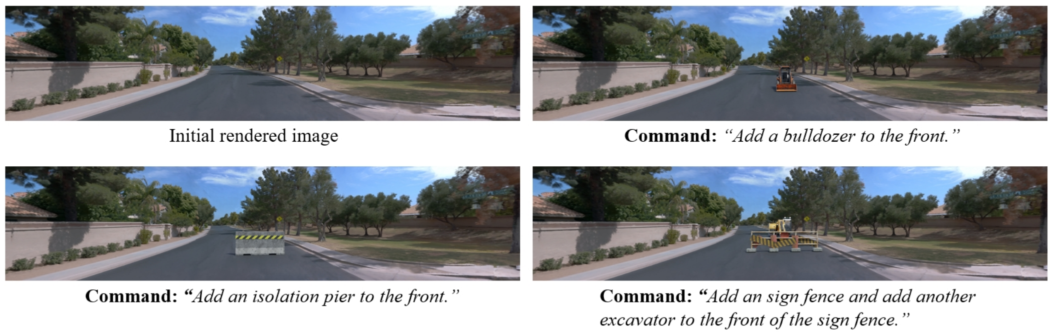

Initial rendered image
Command: *"Add a bulldozer to the front."*
Command: *"Add an isolation pier to the front."*
Command: *"Add an sign fence and add another excavator to the front of the sign fence."*

Figure 17: Qualitative results of rare cases simulation.

I Supplymentary Experiments

I.1 Qualitative Ablation Study of Background Rendering

Figure 15 illustrates the effects of the ablation study on background rendering. It is evident that in the absence of pose adjustment, the rendered results exhibit significant blur and anomalies. Without the intervention of exposure adjustments, there are noticeable changes in brightness at the junctions of different cameras, particularly in the sky. McNeRF, however, successfully avoids these two issues and achieves the optimal rendering outcomes.

I.2 Occlusion with Depth Test

During the process of adding vehicles, there may be instances of occlusion. For occlusions among multiple vehicles to be added, Blender considers this issue during the rendering process. Therefore, we only need to focus on the occlusion between the foreground vehicles and the background objects. The most straightforward method to handle occlusion is determined by the depth map of the foreground and background, respectively. The depth maps of both the foreground and background could be used to choose for each pixel with the lesser depth to be displayed in the front, while the one with greater depth is occluded. However, accurately estimating the background's depth map directly is challenging. The point cloud data in autonomous driving datasets is too sparse, and the depth maps obtained through depth completion are also sparse and excessively noisy, making them unsuitable for pixel-level accuracy in practical use. Here, we combine the sparse depth data from point clouds with the object segmentation method SAMKirillov et al. (2023). SAM can achieve pixel-level accuracy in segmentation results at the image level, without extra finetuning. We first use SAM to obtain different patches in the background image, then identify patches that overlap with the foreground objects. Using the sparse depth map derived from the point clouds, we calculate the average sparse depth within these patches as the depth of each patch. Since the segmentation results of patches often represent a complete instance, and occlusion occurs between instances, it is reasonable to calculate the depth for the entire instance represented by a patch. Subsequently, we create the background's depth map from the depths of these patches and perform occlusion calculations with the depth map rendered for the foreground, presenting each pixel with the lesser depth to finalize the occlusion computation. The results of the occlusion calculation, as shown in Fig. 16, illustrate that the added vehicles are occluded by those with shallower depths. This figure also displays the adjustment of the added vehicles' colors.

Simulation data	AP30	AP50	AP70
0	0.1263	0.0366	0.0034
600	0.1910	0.0878	0.0153
1000	**0.2074**	**0.0930**	**0.0189**
2200	0.2064	0.0900	0.0182

Table 7: Comparison of detection model's performance with different number of data simulated by ChatSim

I.3 Rare Cases Simulation

Leveraging diverse external digital assets, ChatSim can simulate rare and challenging-to-collect real-world scenarios within reconstructed existing scenes. Figure 17 demonstrates ChatSim's ability to emulate rare cases by placing uncommon elements like bulldozers, isolation piers, fences, excavators, and other infrequently encountered situations in reconstructed scenes. This capability enables ChatSim to create rare digital twins for existing collected data, thus fulfilling the need for these specific scenarios.

I.4 Supplementary 3D Detection Augmentation Experiment

We conducted 3D detection augmentation experiments under a new setting: we fixed the real data amount from the original dataset at 4200 frames and augmented it with varying quantities of simulation data generated by ChatSim. We continued to use Lift-Splat Philion & Fidler (2020) as the detection model, with results shown in Table 7. It is observed that the use of simulation data significantly enhances the performance of the 3D detection task. As the amount of simulation data increases, the final performance tends to stabilize after a certain point of improvement.