SIMSplat: Predictive Driving Scene Editing with Language-aligned 4D Gaussian Splatting

Sung-Yeon Park[1], Adam Lee[2], Juanwu Lu[1], Can Cui[1], Luyang Jiang[1]
Rohit Gupta[3], Kyungtae Han[3], Ahmadreza Moradipari[3], and Ziran Wang[1]

[1]Purdue University [2]UC Berkeley [3]Toyota InfoTech Labs
{sungyeon, ziran}@purdue.edu

Figure 1. Our framework enables language-guided editing in driving scenarios. It begins by directly querying the target object from the Gaussian scene, followed by editing through the LLM agent, and verification of all agents' motions via multi-agent path refinement. This process ensures that the final edited scene remains realistic, even including pedestrians. The red/green areas indicate possible collision and refined region, respectively.

Abstract

Driving scene manipulation with sensor data is emerging as a promising alternative to traditional virtual driving simulators. However, existing frameworks struggle to generate realistic scenarios efficiently due to limited editing capabilities. To address these challenges, we present ***SIMSplat****, a predictive driving scene editor with language-aligned Gaussian splatting. As a language-controlled editor, SIMSplat enables intuitive manipulation using natural language prompts. By aligning language with Gaussian-reconstructed scenes, it further supports direct querying of road objects, allowing precise and flexible editing. Our method provides detailed object-level editing, including adding new objects and modifying the trajectories of both vehicles and pedestrians, while also incorporating predictive path refinement through multi-agent motion prediction to generate realistic interactions among all agents in the scene. Experiments on the Waymo dataset demonstrate SIMSplat's extensive editing capabilities and adaptability across a wide range of scenarios. Project page: simsplat.*

1. Introduction

Driving simulators have long played a vital role in the development of autonomous driving algorithms, serving as robust virtual testbeds. Traditional platforms such as CARLA [2] and AirSim [16] have been widely adopted due to their scalability enabled by game engines. However, with the rapid progress of generative models and neural scene reconstruction, approaches based on diffusion models [3, 19, 27, 30], Neural Radiance Fields (NeRFs) [5, 21, 23, 25], and Gaussian Splatting [1, 28, 29] have emerged as powerful alternatives. These methods not only reconstruct photo-realistic and geometrically precise driving scenes directly from sensor data but also enable editing scenarios from real-world data, reducing the need for creating long-tail scenarios in virtual environment. More recently, language models have been integrated into this paradigm, allowing users to edit scenes through natural language prompts.

Despite these advances, several limitations remain. First, fine-grained object-level editing is still constrained. To edit a specific object, current methods often rely on an additional 3D object detector or perception model to provide bounding boxes, which are then passed to the generative model as local conditions for removal and inpainting [21, 30]. In addition, because objects are not separately constructed within the scene, per-object editing is rarely supported. As a result, edits to existing objects are typically limited to removal, while adding new objects generally depends on pre-stored virtual assets. This restricts per-object modifications such as changing locations or motions of agents. Second, scenario feasibility is usually validated only for the ego vehicle and a designated target object [21, 27]. In reality, however, any behavioral change affects all surrounding agents, including vehicles and pedestrians. Limiting validation to a single agent often produces unnatural or inconsistent outcomes. Finally, most prior works focus on rigid objects like vehicles, overlooking pedestrians despite the fact that pedestrians are important in safety-critical and corner-case scenarios. Enabling pedestrian-level editing is therefore essential for generating realistic and comprehensive driving simulations.

To address these challenges, we propose a unified language-controlled driving simulator, **SIMSplat**, which supports *object-level editing* and *multi-agent path refinement*. Our framework starts from integrating *motion-aware language embeddings* with 3D Gaussians to directly query and manipulate 3D scenes with prompt. In this stage, the motion-aware embeddings align language features with the dynamic movements and spatial locations of agents, extending beyond prior works such as LangSplat and 4DLangSplat [10, 13], which are effective in relatively static or indoor environments but less suited for highly dynamic driving scenes. Based on this object-grounding, an LLM agent supports detailed editing by identifying target objects for modification. Furthermore, the LLM agent generates initial path plans (e.g., turn left, stop at a location, add a following vehicle) through function calling, which are then refined with a multi-agent motion prediction model to ensure global consistency and realism across the scene. Through iterative conversation, users can further adjust the details by changing various asset/motion parameters before obtaining the final rendered simulation.

In our experiments, SIMSplat demonstrates extensive capabilities for editing and simulating driving scenes, ranging from inserting static objects to sophisticated modifications of dynamic agents. Our motion-aware language-alignment module achieves state-of-the-art performance in road-object querying, surpassing baselines by 61.2% in accuracy. SIMSplat also attains the highest task completion rate among simulators, highlighting its flexible usage and broad range of editing functionalities. Finally, our multi-agent path refinement enables predictive simulation, achieving the lowest collision and failure rates in our experiments.

In summary, our contributions are threefold.

- A motion-aware language-alignment that enables precise 3D object localization in dynamic driving environments.
- An LLM-based object-level editor that supports fine-grained modifications of both vehicles and pedestrians.
- A multi-agent path refinement that ensures globally consistent and realistic scenario generation beyond ego-target validation.

2. Related Work

Scene editing for driving simulation. To generate realistic driving scenarios, diffusion models, NeRFs, and Gaussian Splatting have been widely adopted. Diffusion-based models typically focus on image or video generation conditioned on reference frames, control parameters, or 3D bounding boxes [6, 8, 11, 12, 19, 24, 27]. While some works, such as DriveDreamer4D [27], leverage text to generate novel trajectories, most diffusion approaches remain heavily dependent on reference images or LiDAR points, making it difficult to achieve flexibility and diversity in generation. Other works employ diffusion models for 3D-consistent view synthesis with modified movements such as lane changes, yet these approaches are still limited to editing ego vehicle trajectories [6, 12, 24].

For instance-aware editing with geometric consistency, researchers have explored compositional scene reconstruction using NeRFs and Gaussian Splatting [9, 23, 29]. MARS [23] separates foreground objects (cars, pedestrians, buses, etc.) and background into independent neural fields, enabling modular editing. Similarly, OmniRe [1] proposes a scene-graph representation based on 3D Gaussians with rigid and non-rigid nodes. However, both approaches require explicit geometric manipulation of nodes, which makes editing inefficient. To simplify this process,

language-controlled editing has been recently introduced. ChatSim [21] and SceneCrafter [30] incorporate natural language prompts into NeRF- and multi-view diffusion-based frameworks, respectively. SceneCrafter supports editing of environmental conditions (e.g., weather, time of day) and object-level operations (adding/removing agents), but requires users to provide 3D bounding boxes for object-specific edits. Moreover, because edits are performed per frame using multi-view diffusion, explicit motion editing of dynamic agents is not supported. ChatSim allows free-form language prompts but cannot directly modify existing NeRF instances; instead, it deletes an object and re-inserts a new asset with a different motion. Despite these advances, most existing editors still lack support for pedestrian-level editing, and few include a global refinement step that accounts for interactions among surrounding agents after scene modifications.

Language field in 4D Gaussians. Recent work on distilling language features into 3D Gaussian scenes has largely followed the LangSplat-style approach [13], which extracts object masks with SAM and then encodes static language features with CLIP [14] to embed semantics at the Gaussian level. To extend beyond static representations, DGD [7] incorporates CLIP features into deformable Gaussians, capturing dynamics through deformable parameters. 4-LEGS [4] applies ViCLIP [20] to extract pixel-aligned spatio-temporal features rather than relying solely on static CLIP embeddings. Similarly, 4DLangSplat [10] leverages multimodal LLMs to generate object-level captions that incorporate contextual scene information, thereby embedding temporal semantics. However, these models are computationally heavy for scene-specific training, as they require preprocessing with MLLMs or ViCLIP and render features at the per-pixel level. Moreover, dynamics in indoor scenes with relatively few objects differ substantially from those in driving environments. In particular, behavioral prototypes that commonly appear in driving, such as "turning left at an intersection", are rarely captured effectively by existing methods.

3. Method

Our framework consists of four main stages. First, we train a scene-graph-based 4D Gaussian Splatting (4DGS) model to reconstruct the scene. Next, we perform Language-Gaussian Alignment by embedding appearance, motion, and location features into the Gaussians, enabling direct open-vocabulary querying of road objects. With this language-augmented scene, the LLM agent interprets user prompts to edit the environment, such as adding, removing, or modifying objects. The edits are then refined through a multi-agent path refinement module, which verifies the edited object and adjusts surrounding agents so that they respond naturally to the changes. Scene editing can be further controlled through multi-turn conversations after visualizing the rendered results. Finally, a diffusion-based inpainting model polishes the modified regions to ensure that the rendered outputs appear seamless and realistic.

3.1. Scene-Graph based 4D Gaussian Splatting

4DGS represents a scene as a set of anisotropic Gaussian blobs carrying spatial and appearance attributes. A Gaussian at time t is written as $g_i(t) = (\mu_i(t), s_i(t), q_i(t), c_i(t), o_i)$, where $\mu_i(t)$ is the 3D center, $s_i(t)$ the scale, $q_i(t)$ the orientation, $c_i(t)$ the color features, and o_i the opacity. When projected onto the image plane, each Gaussian contributes a 2D footprint determined by its position, scale, orientation, and opacity, and the final pixel color is obtained by alpha blending of overlapping Gaussians:

$$C_t(u) = \sum_i c_i(t), \alpha_i(t) \prod_{j<i} \big(1 - \alpha_j(t)\big), \tag{1}$$

where $\alpha_i(t)$ is the transparency weight of Gaussian i at pixel u, computed from its spatial and opacity parameters. Extending beyond static 3DGS, the 4D formulation $\mathcal{G}(t) = g_i(t)_{i=1}^{N}$ models temporal evolution by allowing Gaussians to transform smoothly across time.

To faithfully reconstruct dynamic scenes while preserving the motion and geometry of diverse road objects, we adopt a scene-graph formulation of 4DGS. In contrast to neural field–based approaches [26], the scene-graph decomposes a scene into separate nodes, enabling modular reconstruction and controllable editing. Following the OmniRe framework [1], we represent the scene with three node types: rigid objects (e.g., vehicles), non-rigid objects (e.g., pedestrians), and the static background. Each node is defined in a canonical space and transformed into the world coordinate system at time t. For a rigid node v, the time-dependent Gaussians are given by

$$G_v^{\text{rigid}}(t) = T_v(t) \otimes \bar{G}_v^{\text{rigid}}, \tag{2}$$

where $\bar{G}_v^{\text{rigid}}$ are canonical Gaussians and $T_v(t) \in SE(3)$ is the rigid body transformation. For a non-rigid node h, the Gaussians undergo both global motion and local deformation, expressed as

$$G_h^{\text{nonrigid}}(t) = T_h(t) \otimes F\big(\bar{G}_h^{\text{nonrigid}}, t\big), \tag{3}$$

where $F(\cdot, t)$ denotes a non-rigid deformation applied to the canonical Gaussians, modeling human pose and gestures through joint-level prediction.

Finally, the complete dynamic scene at time t is obtained by aggregating all node types into a unified scene graph

$$\mathcal{S}(t) = \{G_{\text{bg}}, \{G_v^{\text{rigid}}(t)\}, \{G_h^{\text{nonrigid}}(t)\}\},$$

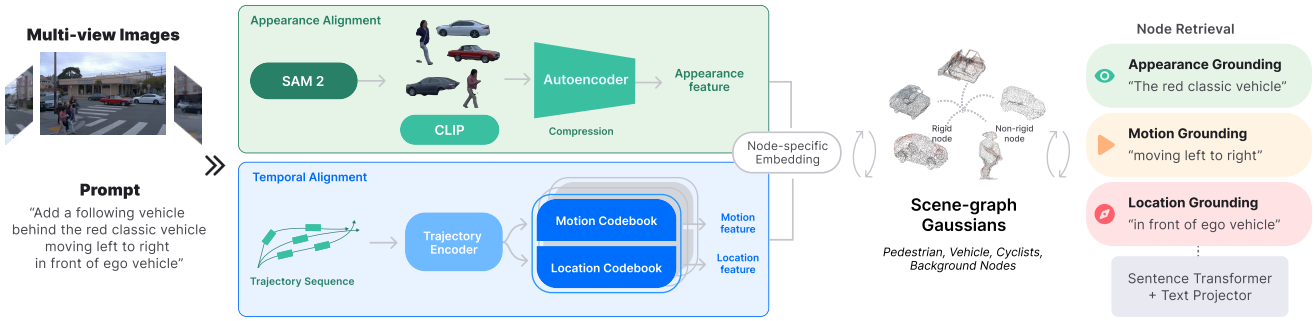

Figure 2. **Pipeline of language alignment.** The appearance and temporal alignment modules extract appearance, motion, and location features, which are then embedded into scene-graph Gaussians. Given a natural language prompt, these features enable grounding of the corresponding objects in the road scene.

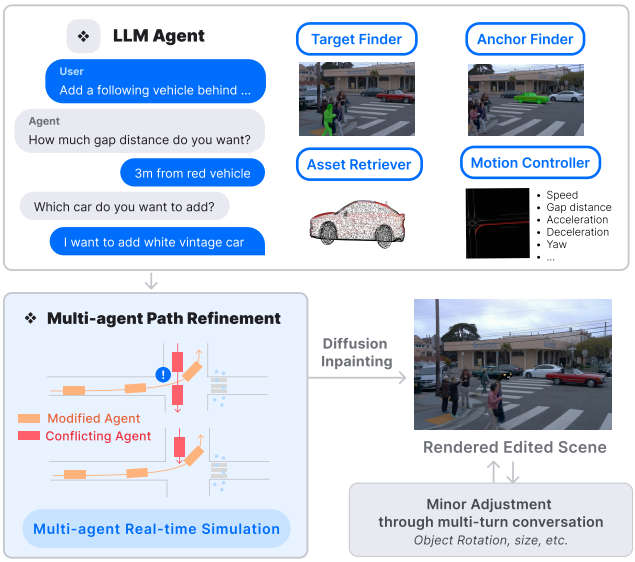

Figure 3. **Editing process.** Given a user prompt, the LLM agent coordinates multiple modules. After identifying the target object, retrieving assets, and planning an initial trajectory, the results are refined by the multi-agent path refinement module. Finally, diffusion-based inpainting is applied and the edited scene is rendered.

where G_{bg} denotes the static background Gaussians, and the other terms correspond to rigid and non-rigid nodes. Through blending and rasterization of these Gaussians, the final rendered scene is produced.

3.2. Language-Gaussian Alignment

To edit the scene with natural language prompt, the most important thing is precisely detect the target or related objects in Gaussian scenes. Unlike the existing editable simulators that utilizes separate detection modules or bounding boxes inputs to localize specific objects, we instead embed language features on the Gaussians so that we can directly query on the Gaussian scenes. Through this direct language embedding on Gaussians, open-vocabulary querying is possible both in pixel and Gaussian level.

Appearance Alignment. As shown in Figure 2, first, we embed static language features on Gaussians to capture object appearance independently of time. Similar to LangSplat [13], we use SAM-2 [15] to extract segmentation masks $M_t(o)$ for each object o, but unlike their hierarchical segmentation, we only use a single mask per object, considering the relatively small size of road agents. Bounding boxes are provided as prompts to SAM-2 for accurate mask generation. From each masked region, we extract CLIP features $f_t^{\text{CLIP}}(o) \in \mathbb{R}^D$. Since CLIP features are high-dimensional, we train a lightweight autoencoder (E, Ψ) to compress them into a latent space:

$$h_t(o) = E(f_t^{\text{CLIP}}(o)) \in \mathbb{R}^d, \quad d \ll D,$$

with reconstruction objective $\Psi(h_t(o)) \approx f_t^{\text{CLIP}}(o)$. These latent codes $h_t(o)$ act as ground-truth supervision for Gaussian appearance features, denoted as $c_i^{\text{app}} \in \mathbb{R}^d$, optimized in a self-supervised manner. Accordingly, each Gaussian at time t is extended to

$$g_i(t) = \left(\theta_i(t), c_i^{\text{app}}\right),$$

where $\theta_i(t) = (\mu_i(t), s_i(t), q_i(t), c_i(t), o_i)$ denotes the original Gaussian parameters, and the additional term c_i^{app} represents the appearance-aligned feature in the latent space. During inference, c_i^{app} is decoded back into the CLIP space $\Psi(c_i^{\text{app}}) \in \mathbb{R}^D$ and compared against the encoded text query, enabling open-vocabulary selection at both pixel and Gaussian levels.

Temporal Alignment. Through experiments, we found that existing methods to embed static or dynamic language features such as LangSplat and 4DLangSplat [10, 13] are not capable of understanding the behavior of road objects. In particular, queries often involve traffic-scene-specific descriptions such as "a pedestrian crossing the street on the left side of the ego vehicle" or "a vehicle crossing the intersection from right to left," which include motion cues (speed, direction, turning) and relative location.

To address this limitation, we propose a temporal alignment module that encodes trajectories $X = \{(x_t, y_t)\}_{t=1}^{T}$

of each object into a latent representation $z = E_{\text{traj}}(X) \in \mathbb{R}^d$ using a trajectory encoder, which is then associated with two codebooks: a *motion codebook* and a *location codebook*, defined as

$$\mathcal{C}^{\text{motion}} = \{p_k^{\text{motion}}\}_{k=1}^{K_m}, \quad \mathcal{C}^{\text{location}} = \{p_k^{\text{location}}\}_{k=1}^{K_l},$$

where each prototype corresponds to a canonical motion or location description (e.g., turning left, moving right to left, in front of ego, on the left side of ego). To account for the different motion patterns of rigid and non-rigid agents, we maintain separate motion codebooks for vehicles and pedestrians, while sharing the same location codebook across object types. Given a trajectory embedding z, similarity scores with prototypes are computed, and the aggregated features $f^{\text{motion}}, f^{\text{location}} \in \mathbb{R}^d$ are obtained as convex combinations of the corresponding codebook vectors. These serve as temporal language features capturing both how the object moves and where it is relative to the ego vehicle.

During training, motion and location alignment losses are combined with commitment terms as

$$\mathcal{L}_{\text{temp}} = \lambda_{\text{align}}\big(\mathcal{L}_{\text{motion}} + \mathcal{L}_{\text{location}}\big) + \lambda_{\text{commit}}\mathcal{L}_{\text{commit}},$$

where $\mathcal{L}_{\text{motion}}$ and $\mathcal{L}_{\text{location}}$ measure cosine distance to text prototypes, and the commitment terms enforce closeness between z and the selected codebook features. Finally, the temporal feature is associated with each object node, represented as

$$G_o(t) = \big(\{g_i(t)\}_{i=1}^{N_o}, c_o^{\text{temp}}\big),$$

where $\{g_i(t)\}_{i=1}^{N_o}$ are the Gaussians belonging to object o, and $c_o^{\text{temp}} = (f^{\text{motion}}, f^{\text{location}})$ encodes the behavioral context at the object level. During inference, object trajectories are first encoded into temporal features c_o^{temp} and compared with encoded text prompt. This yields a unified embedding that compactly represents the behavioral dynamics of each object.

Gaussian Object Querying. For querying the scene with both appearance and temporal features, we first encode the text prompt with E5 Sentence Transformer [18]. Appearance features rendered from Gaussians are decoded through a pretrained autoencoder and compared against prompt embeddings to generate pixel-level similarity maps, which are post-processed into binary masks. These masks are then used to select candidate object instances per prompt. For vehicles and pedestrians, separate trajectory encoders are loaded (trained with motion and location codebooks) together with their text projectors, and the temporal features of candidate trajectories are aligned with E5 text embeddings using cosine similarity. At inference, appearance similarity first narrows down candidate regions, after which motion and location features are combined to select the most relevant object based on the highest cosine similarity.

3.3. LLM Agent

In our framework, the LLM agent serves as the central coordinator that bridges users and multiple functional modules, as described in Figure 3. Given a natural language prompt, it reformats the input into a predefined structure consisting of task type, action parameters, asset parameters, and target/anchor queries. Based on this structured representation, if querying Gaussians is required, the *Target/Anchor Finder* is first triggered. For example, with a prompt such as "Make the black car turning at the intersection go straight," the Target Finder identifies the target object by extracted caption "black car turning at the intersection," encoding it with a Sentence Transformer, and retrieving the most relevant object from the reconstructed Gaussian scene as described in the previous section. Similarly, for prompts such as "Add a bulldozer 5m behind the black car crossing the street," the Anchor Finder is triggered to localize the anchor object ("black car crossing the street") in the scene and determine the referenced position ("5m behind") in map coordinates.

Once the target or anchor object is identified, the *Asset Retriever* is invoked when new assets are required. It retrieves appropriate assets (vehicles, pedestrians, traffic signs, or road objects such as bulldozers, barriers, and cones) from the asset bank. Unlike existing simulators that rely on purely virtual or static pedestrian models, our asset bank includes dynamic real pedestrian assets extracted directly from Waymo scenes, captioned with details such as "a pedestrian walking from left to right with a red backpack." This enables realistic insertion of new pedestrians and enhances the fidelity of simulations. The Asset Retriever also allows control over asset size, rotation, offset, and other parameters, enabling seamless insertion and iterative adjustment through multi-turn interactions.

Finally, the *Motion Controller* is responsible for modifying or generating trajectories. It can adjust an existing object's motion (e.g., accelerate, decelerate, or change direction) or generate new position/trajectories for inserted objects (e.g., adding a following vehicle, placing a static obstacle, or adding a turning vehicle). It also supports object replacement and removal. The controller accepts various action parameters, such as speed, direction, start time, relative distance, and start/end positions, to flexibly guide scene editing. After rendering the edited scenes, users can further adjust these parameters to achieve the desired outcomes.

3.4. Multi-agent Path Refinement

While the trajectory outputs generated by the LLM provide a useful starting point, the LLM is not specifically designed as a path planner and therefore has limited ability to reason about surrounding traffic participants and road context. Existing approaches often generate modified trajectories with built-in functions, but these typically focus only on producing collision-free paths for the target object. In contrast,

our framework aims to create realistic scenarios in which surrounding agents also respond naturally to changes in the scene.

To achieve this, we apply multi-agent path refinement using a motion prediction model that forecasts the future trajectories of all agents. Specifically, we train SMART-1B [22] on the Waymo Open Dataset [17] and use it to simulate the collective behavior of agents. Given the pre-generated trajectory τ^{edit} of the target object from the Motion Controller and the observed history $X_{1:t} = \{x^o_{1:t}\}_{o=1}^N$ of all agents, the predictor P generates future rollouts

$$\hat{X}_{t+1:T} = P(X_{1:t}, \tau^{\text{edit}}),$$

where $\hat{X}_{t+1:T}$ denotes the refined trajectories for both the target and surrounding objects. This allows our framework to roll out realistic scenarios. For example, a following vehicle will make a detour or stop when the edited vehicle ahead brakes abruptly at an intersection, or nearby cars will react accordingly when a jaywalking pedestrian appears. This not only adapts the behavior of vehicles in the scene but also models pedestrian dynamics, for example, a pedestrian stopping when an obstacle is newly added in front. Since the refinement depends on a prediction model and may be sensitive to uncertainty or failure cases, we also provide the option to bypass this module when deterministic editing is preferred. In this way, the module ensures that edited scenarios remain both plausible and consistent with real-world multi-agent dynamics.

4. Experiments

4.1. Datasets and Settings

We conduct experiments on the Waymo Open Dataset [17]. For training, we use images from the front, front-left, and front-right cameras, with 100 frames sampled at 10 Hz for each sequence. Every 10th frame is reserved for the test set and training is performed on a single NVIDIA A100 GPU. For appearance feature extraction, we adopt OpenCLIP ViT-B/16, while prompt encoding is handled by the E5-mistral-7B model [18] . For motion generation, the first 11 timesteps of each trajectory are used as input to SMART-1B [22], which predicts 80 future timesteps.

4.2. Road Object Querying

We first compare open-vocabulary querying results against state-of-the-art baselines, LangSplat and 4DLangSplat. As shown in Table 1, when given queries containing object descriptions, our method achieves an overall accuracy of 0.64 and a vIoU of 0.19, outperforming other models by a significant margin. In particular, for vehicle objects, our method attains 0.76 in accuracy and 0.26 in vIoU, nearly doubling the performance of the second-best model. Even for pedestrians, which are typically smaller and harder to detect than vehicles, our model achieves 61.2% and 22.2% higher accuracy and vIoU scores, respectively, compared to 4DLangSplat.

Figure 4 illustrates qualitative examples of querying results. Compared with 4DLangSplat, which often fails to locate the correct object following a motion description, our model successfully identifies the target. For instance, in the first case, our model successfully finds the vehicle turning right among several moving vehicles, whereas 4DLangSplat instead selects a vehicle driving straight. In another case, our model identifies the pedestrian standing still on the left side, while 4DLangSplat incorrectly highlights a pedestrian walking across the crosswalk. Although not shown in the figure, even when specific appearance-based prompts are provided together (e.g., black sedan, pedestrian with red coat), multiple agents with similar appearances may exist due to the density of road participants. In such cases, our model resolves ambiguity by incorporating motion and location cues into the language field. These results demonstrate that our temporal alignment strategy effectively embeds the dynamic nature of road agents, thereby enabling more accurate and reliable language-grounded querying.

Method	Vehicle		Pedestrian		Total	
	Acc.	vIoU	Acc.	vIoU	Acc.	vIoU
LangSplat [13]	0.24	0.1	0.25	0.06	0.24	0.08
4DLangSplat [10]	0.35	0.15	0.31	0.09	0.33	0.12
Ours	**0.76**	**0.26**	**0.5**	**0.11**	**0.64**	**0.19**

Table 1. **Object querying results.** Metrics reported are accuracy for Acc. and video-level IoU for vIoU. (Since ground-truth object masks often include multiple objects due to prompt ambiguity in road environments, the resulting vIoU scores are low in general.)

4.3. Scene Editing

We evaluate scene editing results across various types of natural language commands. As baselines, we primarily compare against ChatSim [21], which offers language-based editing most similar to ours, and OmniRe [1], which supports object-level editing in Gaussian splatting scene. For fariness, all models are trained under the same scenario. Considering the different editing capabilities of each baseline, we adapt the prompts to fit the respective framework, such as replacing assets or modifying position prompts with adjusted coordinates.

Qualitative Results. Figure 5 presents qualitative examples of scene editing with our framework. In the first row, we demonstrate adding static road objects, which influence traffic behavior. As shown in the prompts, users can specify object placement either through relative descriptions (e.g., “in front of the construction worker” or “next

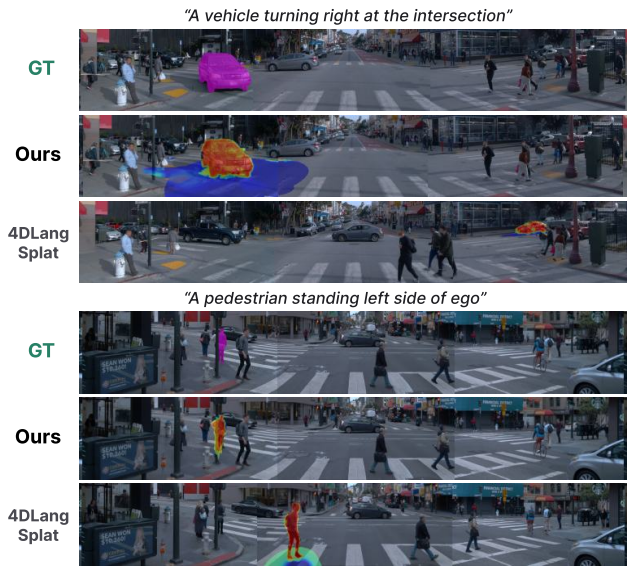

Figure 4. **Qualitative comparison of object querying.** Our method effectively captures the behaviors of road agents within the scene.

Figure 5. **Qualitative editing results.** SIMSplat supports various types of editing, including adding new objects, removal or replacement, and modification of pedestrians or vehicles. Gray areas indicate the edited regions, and images are zoomed in for clearer visualization.

to the bus ahead") or with explicit coordinates (x, y, z). In the second row, object removal and replacement are illustrated. Beyond targeting individual objects, our framework also supports group-level commands such as "remove all moving pedestrians/vehicles". In addition to static edits, our framework uniquely supports pedestrian editing. A key strength of our approach is that it not only enables modification of existing pedestrians (e.g., adjusting walking speed or removal) but also allows the incorporation of real-human pedestrian assets extracted from other scenarios. This allows users to add realistic, dynamically moving pedestrians. Unlike prior works, our pedestrian assets preserve natural joint motions and gestures, avoiding the appearance of artificial animation. For example, the figure shows a newly added pedestrian crossing the street with natural motion. This further enables creating safety-critical cases, such as inserting jaywalking pedestrians or wheelchair users, through fine-grained control over pedestrians.

Figure 6 illustrates how our framework predicts the future trajectories of agents and incorporates them into edited scenes. In the top-left case, a right-turning black vehicle is edited to go straight. Then, the path refinement module predicts a potential collision with pedestrians, so the vehicle is instead adjusted to stop and yield. In the top-right case, we insert a truck proceeding straight through an intersection. To merge with right-turning vehicles, it waits for a safe gap before merging smoothly into traffic, demonstrating our framework's ability to adapt new objects to ongoing traffic conditions. In the bottom-left case, a traffic cone is added in the middle of a crosswalk. Two approaching vehicles detect the new obstacle and gradually stop in front of it. In the last case, a vehicle attempting parallel parking abruptly halts, likely due to becoming stuck. The following gray vehicle then makes a slight detour to avoid it. Other than these cases, pedestrians also adapt to edits: for example, in the previous Figure 5, a construction worker originally crosses the street, but after a barrier blocks the crosswalk, it remains stopped at the crosswalk. These qualitative results demonstrate that our framework not only enables efficient scene editing but also generates coherent multi-agent interactions through predictive path refinement.

Task Completion. We compare task completion rates using prompts categorized into three types: adding new objects, modifying existing objects, and removal. As shown in Table 2, our model achieves the highest task completion rate of 84.2%. While ChatSim can add new and static objects, it does not support editing pedestrians or modifying existing objects, since its method does not model each road agent individually or capture motion changes over time. In contrast, OmniRe supports adding pedestrians and modifying existing objects through its scene-graph-based modeling. However, it lacks the ability to control objects with detailed prompts, such as specifying turning directions or speeds. Furthermore, neither baseline can target specific objects with open-vocabulary queries, which limits their ability to provide fine-grained guidance for scene editing, such as positioning a particular object at a particular location or making it move to desired direction.

Motion Generation. Table 3 reports failure rates in terms of interactions with the surrounding environment. GPT2Motion refers to using GPT-5 directly as a motion generator, and we also evaluate a variant of our model with-

Figure 6. **Qualitative results with predictive path refinement.** The multi-agent path refinement module adapts both target and surrounding objects to interact in edited scenes. Gray areas indicate the edited regions, and images are zoomed in for clearer visualization.

Method	Add new			Modify		Remove	Total
	Veh.	Obj.	Ped.	Veh.	Ped.		
ChatSim [21]	50.0	33.3	0.0	0.0	0.0	16.7	18.4
OmniRe [1]	0.0	33.3	14.3	40.0	80.0	83.3	39.5
Ours	**83.3**	**88.9**	**85.7**	**80.0**	**80.0**	**83.3**	**84.2**

Table 2. **Task completion results.** The metric is defined as the percentage(%) of successfully executed prompts to the total number of prompts. Veh., Obj., and Ped. denote vehicle, static object, and pedestrian, respectively.

out the multi-agent path refinement module. As shown, when analyzing failure cases, including collisions with vehicles or pedestrians, off-road driving, and overall failure, our model consistently achieves the lowest error rates. Other baselines perform well when a static object is placed in isolation, away from surrounding agents. However, when the edited object must coordinate with nearby agents, collisions or off-road driving occur much more frequently. While target vehicle oriented planning and validation may succeed in sparse traffic or isolated settings, they often fail to adapt effectively in complex urban scenarios. These results highlight that scene editing in real-world driving requires not only generating collision-free trajectories for a single object but also dynamically adjusting the motions of surrounding agents in line with their predicted interactions.

Method	Collision Veh.	Collision Ped.	Off-road	Failure
ChatSim [21]	33.3	20.0	20.0	93.3
OmniRe [1]	26.7	35.0	4.8	86.7
GPT2Motion	54.0	67.0	20.0	83.0
Ours w/o refinement	46.7	26.7	6.7	66.7
Ours	**8.5**	**3.3**	**6.7**	**10.4**

Table 3. **Motion generation results.** Among completed tasks, we calculate collision rate with vehicles/pedestrians, off-road driving, and total failure cases.

5. Conclusion

We presented SIMSplat, a novel framework for language-guided editing of driving scenes. As a unified system centered on an LLM agent, SIMSplat supports detailed modifications and realistic simulations. By introducing motion-aware language alignment on Gaussian objects, it enables efficient grounding of road objects and intuitive manipulation through natural language prompts. Our scene-graph-based reconstruction further allows object-level editing of newly added or existing agents, including pedestrians. The editing functions support modifications of positions and trajectories with flexible parameters, enabling users to simulate diverse scenarios and analyze their outcomes. In addition, our multi-agent path refinement module produces reactive simulations in which surrounding agents respond to edits, yielding realistic and predictive behaviors after scene modifications. Experiments on the Waymo dataset demonstrate SIMSplat's extensive editing capabilities, achieving the highest task completion rate and precision. In future work, we plan to enhance the framework's scalability across a broader range of scenarios and extend its functionality to support more diverse editing tasks.

References

[1] Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, Li Song, and Yue Wang. Omnire: Omni urban scene reconstruction, 2025. 2, 3, 6, 8

[2] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 2

[3] Lue Fan, Hao Zhang, Qitai Wang, Hongsheng Li, and Zhaoxiang Zhang. Freesim: Toward free-viewpoint camera simulation in driving scenes, 2024. 2

[4] Gal Fiebelman, Tamir Cohen, Ayellet Morgenstern, Peter

Hedman, and Hadar Averbuch-Elor. 4-legs: 4d language embedded gaussian splatting. *arXiv preprint arXiv:2410.10719*, 2024. 3

[5] 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, 2023. 2

[6] Jiazhe Guo, Yikang Ding, Xiwu Chen, Shuo Chen, Bohan Li, Yingshuang Zou, Xiaoyang Lyu, Feiyang Tan, Xiaojuan Qi, Zhiheng Li, and Hao Zhao. Dist-4d: Disentangled spatiotemporal diffusion with metric depth for 4d driving scene generation, 2025. 2

[7] Isaac Labe, Noam Issachar, Itai Lang, and Sagie Benaim. Dgd: Dynamic 3d gaussians distillation, 2024. 3

[8] Bohan Li, Jiazhe Guo, Hongsi Liu, Yingshuang Zou, Yikang Ding, Xiwu Chen, Hu Zhu, Feiyang Tan, Chi Zhang, Tiancai Wang, Shuchang Zhou, Li Zhang, Xiaojuan Qi, Hao Zhao, Mu Yang, Wenjun Zeng, and Xin Jin. Uniscene: Unified occupancy-centric driving scene generation, 2025. 2

[9] Tianyu Li, Yihang Qiu, Zhenhua Wu, Carl Lindström, Peng Su, Matthias Nießner, and Hongyang Li. Mtgs: Multi-traversal gaussian splatting, 2025. 2

[10] Wanhua Li, Renping Zhou, Jiawei Zhou, Yingwei Song, Johannes Herter, Minghan Qin, Gao Huang, and Hanspeter Pfister. 4d langsplat: 4d language gaussian splatting via multimodal large language models, 2025. 2, 3, 4, 6

[11] Hongbin Lin, Zilu Guo, Yifan Zhang, Shuaicheng Niu, Yafeng Li, Ruimao Zhang, Shuguang Cui, and Zhen Li. Drivegen: Generalized and robust 3d detection in driving via controllable text-to-image diffusion generation, 2025. 2

[12] Chaojun Ni, Guosheng Zhao, Xiaofeng Wang, Zheng Zhu, Wenkang Qin, Guan Huang, Chen Liu, Yuyin Chen, Yida Wang, Xueyang Zhang, Yifei Zhan, Kun Zhan, Peng Jia, Xianpeng Lang, Xingang Wang, and Wenjun Mei. Recondreamer: Crafting world models for driving scene reconstruction via online restoration, 2024. 2

[13] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting, 2024. 2, 3, 4, 6

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 3

[15] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. 4

[16] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, 2017. 2

[17] 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)*, 2020. 6

[18] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models, 2024. 5, 6

[19] Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. Drivedreamer: Towards real-world-driven world models for autonomous driving, 2023. 2

[20] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, Conghui He, Ping Luo, Ziwei Liu, Yali Wang, Limin Wang, and Yu Qiao. Internvid: A large-scale video-text dataset for multimodal understanding and generation, 2024. 3

[21] Yuxi Wei, Zi Wang, Yifan Lu, Chenxin Xu, Changxing Liu, Hao Zhao, Siheng Chen, and Yanfeng Wang. Editable scene simulation for autonomous driving via collaborative llm-agents, 2024. 2, 3, 6, 8

[22] Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time motion generation via next-token prediction, 2024. 6

[23] 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*, page 3–15. Springer Nature Singapore, 2024. 2

[24] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians: Modeling dynamic urban scenes with gaussian splatting. In *ECCV*, 2024. 2

[25] Jiawei Yang, B. Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, and Yue Wang. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. *ArXiv*, abs/2311.02077, 2023. 2

[26] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction, 2023. 3

[27] Guosheng Zhao, Chaojun Ni, Xiaofeng Wang, Zheng Zhu, Xueyang Zhang, Yida Wang, Guan Huang, Xinze Chen, Boyuan Wang, Youyi Zhang, Wenjun Mei, and Xingang Wang. Drivedreamer4d: World models are effective data machines for 4d driving scene representation, 2024. 2

[28] Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugs: Holistic urban 3d scene understanding via gaussian splatting, 2024. 2

[29] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes, 2024. 2

[30] Zehao Zhu, Yuliang Zou, Chiyu Max Jiang, Bo Sun, Vincent Casser, Xiukun Huang, Jiahao Wang, Zhenpei Yang, Ruiqi Gao, Leonidas Guibas, Mingxing Tan, and Dragomir Anguelov. Scenecrafter: Controllable multi-view driving scene editing, 2025. 2, 3