# LangDriveCTRL: Natural Language Controllable Driving Scene Editing with Multi-modal Agents

Yun He[1], Francesco Pittaluga[2], Ziyu Jiang[2], Matthias Zwicker[1], Manmohan Chandraker[2,3], and Zaid Tasneem[2]

[1] University of Maryland, College Park
[2] NEC Labs America
[3] UC San Diego
https://yunhe24.github.io/langdrivectrl/


**Fig. 1: Natural-language editing.** Cosmos [3] achieves high visual quality but fails to align with the target behavior and modifies the background, showing poor controllability. While ChatSim [41] preserves background information, it suffers from poor photorealism, inaccurate trajectory generation, and traffic violation (e.g., collision). In contrast, our method achieves photorealism, instruction alignment, structure preservation, and traffic realism simultaneously, significantly outperforming previous methods.

**Abstract.** LangDriveCTRL is a natural-language-controllable framework for editing real-world driving videos to synthesize diverse traffic scenarios. It represents each video as an explicit 3D scene graph, decomposing the scene into a static background and dynamic object nodes. To enable fine-grained editing and realism, it introduces a *feedback-driven* agentic pipeline. An *Orchestrator* converts user instructions into executable graphs that coordinate specialized multi-modal agents and tools. An *Object Grounding Agent* aligns free-form text with target object nodes in the scene graph; a *Behavior Editing Agent* generates multi-object trajectories from language instructions; and a *Behavior Reviewer Agent* iteratively reviews and refines the generated trajectories. The edited scene graph is rendered and harmonized using a video diffusion tool, and then further refined by a *Video Reviewer Agent* to ensure photorealism and appearance alignment. LangDriveCTRL supports both object node editing (removal, insertion, and replacement) and multi-object

behavior editing from natural-language instructions. Quantitatively, it achieves nearly 2× higher instruction alignment than the previous SoTA, with superior photorealism, structural preservation, and traffic realism.

**Keywords:** Video Editing · Multi-modal Agents · Autonomous Driving

## 1 Introduction

Synthetic data generation [32] is increasingly adopted to address the limited diversity and coverage of real-world driving logs [33], especially for training and validating autonomous driving stacks. Because collecting real driving videos, particularly those depicting safety-critical scenarios, is prohibitively expensive and logistically impractical [44]. Traditional driving simulators such as CARLA [10] and AirSim [29] can generate diverse scenarios. However, they rely on manually created 3D assets and require engineers to write scripts for scenario generation and human feedback for refinement.

Recent works attempt to scale these workflows by enabling natural-language-driven scene editing. Agentic pipelines [16, 40, 41] leverage explicit 3D representations and Large Language Models (LLMs) [1] to orchestrate modular tools. However, they suffer from three key issues. 1) They rely solely on unimodal text reasoning without integrating multimodal scene context, which makes it difficult to accurately localize the target objects and generate realistic trajectories. 2) They simply composite the background with the inserted object, resulting in poor rendering quality under large viewpoint changes and failing to achieve lighting-aware insertion. 3) Most importantly, they do not verify intermediate results after each step, leading to error accumulation and poor final results.

In contrast, implicit world models such as Cosmos [3] directly edit videos in pixel space rather than 3D space, achieving strong photorealism and plausible object behavior. However, it sacrifices controllability. Specifically, it does not explicitly support object-level editing and may unintentionally alter scene structure (e.g., inserting unrequested objects). Like agentic pipelines, it is also feed-forward, lacking feedback mechanisms to correct instruction misalignment.

To address these challenges, we propose **LangDriveCTRL**, a feedback-driven, natural-language-controllable framework that unifies explicit scene representation with diffusion-based behavior and video refinement. Our approach is based on two key insights. 1) Fine-grained controllability requires *multi-modal reasoning* that jointly grounds language instructions in visual appearance and traffic context. 2) Photorealism and instruction alignment require *feedback-driven iterative refinement*, where intermediate behaviors and renderings are reviewed and corrected in a closed loop.

LangDriveCTRL operates on a scene-graph representation obtained via explicit 3D decomposition. Each video is modeled as a static background node and dynamic object nodes with trajectories. This design enables object-level editing while preserving scene structure. A central LLM-based *Orchestrator* coordinates reasoning-capable *agents* and functional *tools*. Agents (driven by LLMs or VLMs)

interpret user intent, ground language instructions in scene context, reason about traffic semantics, and iteratively review and refine intermediate outputs. While tools execute atomic operations such as 3D reconstruction [8, 18], text-to-3D generation [51], and multi-object trajectory simulation [7].

Given a user instruction, the *Orchestrator* first decomposes it into object-level sub-tasks and constructs an execution workflow. An *Object Grounding Agent* matches open-vocabulary descriptions to object nodes by jointly reasoning over appearance, behavior, and position information. For behavior editing, a *Behavior Editing Agent* generates counterfactual behavior based on trajectory history and lane information, and invokes a diffusion-based multi-object simulator [7] to generate trajectories. The *Behavior Reviewer Agent* then enforces instruction alignment and traffic realism through a feedback loop. After editing the scene graph, a coarse renderer produces an initial video, which is further harmonized by a custom *Video Diffusion Tool* to address lighting inconsistencies and novel-view artifacts. However, this harmonization may alter the appearance of inserted vehicles. Therefore, a *Video Reviewer Agent* iteratively adjusts diffusion strength and guidance to balance photorealism and appearance preservation. As shown in Figure 1, this feedback-driven, multi-modal design achieves photorealism, instruction alignment, structure preservation, and traffic realism simultaneously, significantly outperforming both world models and prior agentic pipelines.

**Contributions.** Our main contributions are:

- We introduce LangDriveCTRL, a feedback-driven, natural-language-controllable framework for fine-grained object-level editing of driving videos. It supports object removal, insertion, replacement, and multi-object behavior editing.
- We design two novel multi-modal reasoning agents: 1) an *Object Grounding Agent* for open-vocabulary object querying, and 2) a *Behavior Editing Agent* for multi-object trajectory generation.
- We propose feedback-driven iterative refinement of behavior and video via a *Behavior Reviewer Agent* and a *Video Reviewer Agent*, improving both instruction alignment and traffic realism.
- Extensive experiments demonstrate that LangDriveCTRL achieves nearly 2× higher instruction alignment than prior state-of-the-art methods and significantly improves structural preservation, photorealism, and traffic realism. Meanwhile, it maintains comparable latency to existing SoTA approaches.

## 2 Related Work

***Neural Rendering for Driving Scene Editing.*** Neural rendering methods [13, 14, 18, 26, 37, 38] such as NeRF and 3D Gaussian Splatting have been widely adopted for autonomous driving due to their ability to reconstruct compositional 3D scenes and support for object-level editing [8, 36, 43]. While these optimization-based approaches enable modular manipulation of foreground objects and background, they struggle under significant view changes, lack multi-object simulation capabilities, and do not natively support lighting-aware insertion of new objects.

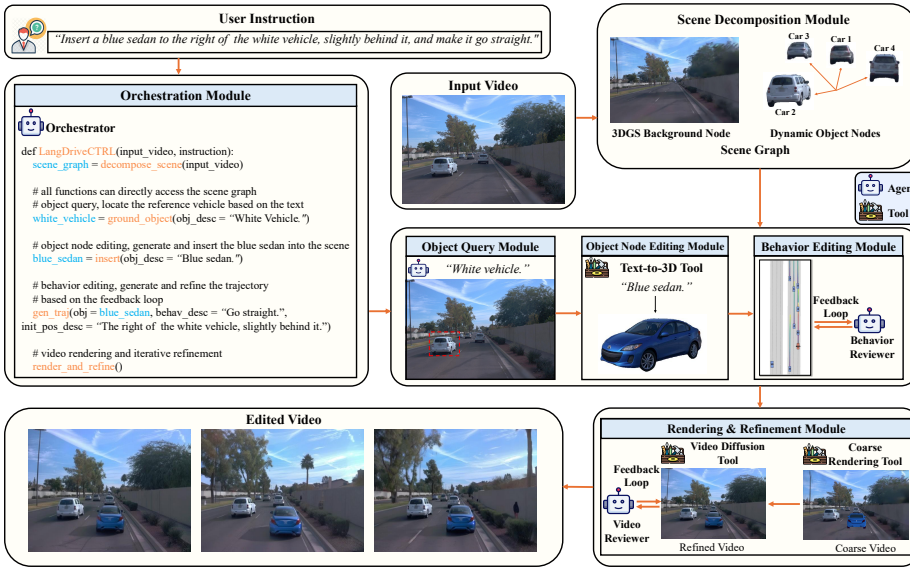


**Fig. 2: Overall Pipeline.** Given an input video and the user instruction, our pipeline first builds a scene graph, which decomposes the scene into a static background node and multiple dynamic object nodes with their trajectories. To execute the instruction, the orchestrator coordinates agents and tools from different modules to work together: the object query module localizes target objects in the scene graph based on textual descriptions; the object node editing module performs object removal, insertion, and replacement; the behavior editing module generates and refines multi-object trajectories based on a feedback loop; finally, the rendering and refinement module renders the edited scene graph and iteratively refines it with a video diffusion tool. While the figure illustrates single-object editing, our pipeline is capable of multi-object editing.

***Diffusion Models for Driving Scene Editing.*** Recent editing methods combine neural rendering with diffusion models [22, 31, 48] for improved robustness to viewpoint changes and lighting-aware object insertion [12, 23, 50, 52]. These pipelines, however, are usually controlled through low-level parameters or 2D/3D bounding box, not natural language. In contrast, the purely generative world model [3] can take natural language instructions and edit videos directly in pixel space, but it lack fine-grained object-level control and often alter the underlying scene structure of the input videos.

***Natural-Language-Controllable Simulation.*** LLM-driven modular simulation pipelines [16, 40, 41] leverage LLMs to provide natural-language control over object-level operations (e.g., removal, insertion and replacement). However, they struggle with accurate target object localization and realistic trajectory generation due to unimodal text reasoning, produce poor rendering quality from naive compositing, and lack iterative refinement.

# 3 Our approach

Our framework follows an agentic pipeline, as shown in Figure 2, that determines which agents (with reasoning ability) and tools (without reasoning ability) to invoke based on user instructions. The pipeline consists of different modules to ensure controllability and interpretability while achieving high realism.

## 3.1 Input

Our pipeline takes a driving video, a user instruction and the scene map as input. We assume that the original object trajectories and map are provided.

## 3.2 Orchestration Module

***Orchestrator Agent.*** The orchestrator is the central agent in our system that controls the overall workflow. It is implemented using an off-the-shelf LLM [1] that we configure using in-context learning [5], and it produces executable Python scripts that call other agents or tools provided in various modules of our system, as shown in Figure 2.

The orchestrator first decomposes the user instruction into sub-instructions for each target object, then designs execution workflows for each object and invokes the corresponding agents and tools from different modules. To enable this, we encapsulate the operations provided by various modules in our system into modular functions that can be easily assembled into executable scripts. We use in-context learning [5] to teach the LLM how to call these functions and generate executable scripts to fulfill user instructions.

The execution workflow proceeds as follows. First, the orchestrator employs the scene reconstruction tool to decompose the 3D scene into a static background node and dynamic object nodes with associated trajectories, generating a scene graph. This scene graph is then shared across all target objects for subsequent editing operations. For each target object, the orchestrator invokes the object grounding agent to locate the target node in the scene graph based on the textual description. Next, depending on the editing type (removal, insertion, or replacement), it calls the appropriate tool or agent to modify the node and update the scene graph. If the instruction involves trajectory editing, the orchestrator invokes the behavior editing agent to generate trajectories, which are then checked and iteratively refined by a behavior reviewer agent. Finally, after all objects have been processed, the orchestrator calls the coarse rendering tool to generate a coarse video, which is then harmonized by the video diffusion tool. A video reviewer agent iteratively refines the result to achieve both photorealism and appearance alignment.

## 3.3 Scene Decomposition Module

The goal of this module is to decompose the input driving video into a scene graph $SG$ that enables object-level reasoning and controllable editing. The scene graph contains a static background node and multiple dynamic object nodes representing vehicles and pedestrians, providing a modular and interpretable representation for fine-grained editing.

***Scene Reconstruction Tool.*** 3D Gaussian Splatting (3DGS) [18] is good at representing and rendering static scenes with high photorealism. Following [8], the tool decomposes the scene into static background Gaussians and canonical object nodes with trajectory-based transformations. These components form a scene graph:

$$SG(t) = \{N_{\text{bg}}, \{N^{\text{i}}_{\text{asset}}(t)\}_{i=1}^{K}\},$$

where $N_{\text{bg}}$ represents the static background Gaussian primitives, and $N^{\text{i}}_{\text{asset}}(t)$ are time-dependent object nodes with node ID i. Each canonical object node is transformed by its pose that captures its motion trajectory. This formulation preserves the spatiotemporal consistency of real-world trajectories while enabling fine-grained, per-object editing.

### 3.4 Object Query Module

The object query module establishes correspondence between textual object descriptions and scene graph nodes through attribute-based reasoning. To achieve this goal, previous methods can be roughly categorized into two types: open-vocabulary detection/tracking algorithms [9, 24, 28, 45] and 3DGS-based approaches [21, 27, 30]. Although these methods perform well at category-level recognition, they struggle with attribute-based distinctions (e.g., color, type, spatial relationship, and motion).

***Object Grounding Agent.*** To address this limitation, we design an object grounding agent powered by the vision-language model (VLM) [17]. It receives three types of information from the input videos, scene graphs and maps to locate target object nodes: 1) appearance information: each node is projected into pixel space and segmented using SAM [19] to extract its visual appearance; 2) behavior information: motion descriptions are generated from trajectory analysis (speed/heading/lane changes) using heuristic rules; 3) position information: trajectory coordinates and lane information are used for spatial relationship analysis. Given all this context information, the agent identifies the target node through a two-stage process. First, it decomposes the query into a triplet: reference node, target node, and their spatial relation (e.g., "the black SUV on the left of ego": <ego, black SUV, left>). Then, it locates the reference node by matching appearance and behavior, filters candidates by spatial relation, and selects the target node through the same matching process.

### 3.5 Object Node Editing Module

After identifying the target node, editing operations (e.g., removal/insertion/replacement) can be easily performed by corresponding tools and agents.

***Removal Tool.*** It removes all Gaussian primitives belonging to the target node, and updates the scene graph.

***Insertion Agent.*** It first invokes a text-to-3D tool (i.e., Hunyuan3D [51]) to generate a mesh, then adjusts its size and local coordinate system to align it with the scene. Finally, the mesh is added to the scene graph as a new node. For size adjustment, the agent first calculates the mesh's bounding box and rescales it to the scene's actual size. For orientation alignment, the agent aligns the mesh's local coordinate system with the scene's world coordinate system by: 1) rendering the mesh from a fixed axis, 2) analyzing its facing direction in the rendered image, 3) determining the local axes.

***Replacement Agent.*** It essentially combines the removal and insertion operations to replace an existing object node with a new one, while the new node inherits the original trajectory.

### 3.6 Behavior Editing Module

The behavior editing process involves two specialized agents. The Behavior Editing Agent generates a counterfactual behavior combination list for each object node based on its original trajectory and map information, then selects the behavior combination that best matches the instruction. It uses the selected result to generate trajectories through a multi-object simulation tool [7]. The Behavior Reviewer Agent then checks the generated trajectories and performs iterative refinement to ensure instruction alignment and traffic realism.

***Behavior Editing Agent.*** The agent first uses heuristic tools to generate behavior description of the object's original trajectory. This is essentially a behavior combination that describes all matched behaviors (e.g., "slow down, change from the middle lane to the left lane, turn left"). The next step is *counterfactual behavior generation*. Replace/remove/keep/add operations are applied to each behavior in the combination to produce new behavior combinations, which form a combination list. These combinations are then filtered using the map to remove unreasonable behaviors, such as ask a vehicle to make a turn when there is no intersection. Additionally, mutually contradictory behaviors are also filtered out, such as combinations containing both "going straight" and "static". Finally, the agent selects the best match from the combination list according to the user instruction and original behavior. The selected result is then used as the text condition for LangTraj [7], a diffusion-based, language-conditioned trajectory simulator for multi-object simulation. Importantly, the selection is a behavior combination rather than a single behavior (e.g., if the user instruction is "speed up", and the original behavior is "slow down, change from the middle lane to the left lane, turn left", the selected behavior combination will be "speed up, change from the middle lane to the left lane, turn left"). The purpose of doing this is twofold: 1) to filter out unreasonable behaviors; 2) to preserve the object's original behaviors as much as possible. For example, if the original behavior is "go straight" and the user asks the vehicle to slow down, the vehicle should maintain going straight while slowing down.

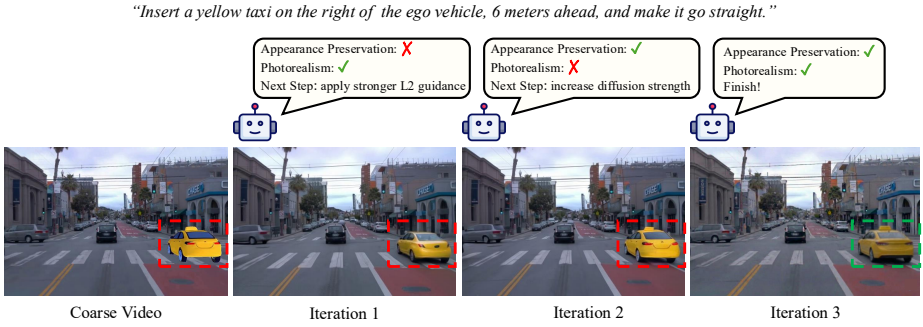


**Fig. 3: Effect of the video reviewer agent.** The video reviewer dynamically adjusts the diffusion strength and the L2 guidance weight based on feedback. *Iteration 1:* The taxi looks realistic, but appearance preservation is compromised (the roof light disappears), so the agent increases the L2 guidance weight. *Iteration 2:* The appearance is preserved, but it looks unrealistic (too bright), so the agent increases the diffusion strength. *Iteration 3:* Both photorealism and appearance preservation are achieved, and the agent stops the refinement.

***Behavior Reviewer Agent.*** However, generated trajectories from LangTraj [7] may not align with instructions and may involve collisions or off-road scenarios. The reviewer agent addresses this through an automatic feedback loop that iteratively validates and refines trajectories. Specifically, it employs trajectory validation functions to evaluate the instruction alignment and traffic rule compliance. For multi-object simulation, the agent handles successful and unsuccessful objects differently. For objects that already satisfy all requirements, it stores their successful trajectories and uses them as guidance for subsequent generations. This makes it easier to achieve trajectory-instruction alignment and also enables interaction with other objects. For objects that do not meet the requirements, the agent adjusts the guidance configuration for LangTraj [7] accordingly. Specifically, if behavior misaligns with the instruction, it increases the classifier-free guidance weight. For off-road or collision violations, it adds the corresponding off-road or collision avoidance guidance and adjusts its weight to improve traffic compliance. Based on this feedback loop, the module can consistently generate realistic and accurate trajectories.

### 3.7 Rendering and Refinement Module

***Coarse Rendering Tool.*** This tool renders the edited scene from the updated scene graph. Specifically, it renders the 3DGS based scene graph using the rasterization algorithm [18], and renders the inserted object meshes using PyVista [34]. The rendered components are then composited with depth information. However, videos generated by this tool typically lack photorealism. The newly inserted objects often appear unnatural, and when new viewpoints differ significantly from the original ones (e.g., when modifying the ego vehicle's trajectory), the rendering quality of 3DGS drops quickly.

***Video Diffusion Tool.*** To address the quality issue, this tool employs a video diffusion model that takes the coarse video as condition to generate the enhanced output. Specifically, it adopts CogVideoX [46] as the backbone and finetunes the model using two strategies: 1) replacing Gaussian primitives in the 3DGS representation with object meshes to learn the photorealistic vehicle appearances, 2) training on noisy Gaussian rendering pairs curated via cycle reconstruction strategy [42] for effective denoising.

***Video Reviewer Agent.*** However, while the video diffusion tool can generate photorealistic results, it may alter the appearance (shape, key parts, type, color, etc) of inserted vehicles. In diffusion processes, higher denoising strength (i.e., greater noise levels) generally produces more realistic outputs but risks losing information from the conditioning input [4, 25]. For example, in Iteration 1 of Figure 3, the yellow taxi's roof light disappears. To improve video quality while preserving vehicle appearance, this VLM powered agent employs feedback-driven iterative refinement. It dynamically adjusts the denoising strength to control photorealism and tunes the L2 guidance loss weight to preserve object appearance. The L2 guidance loss is computed at each denoising step by measuring the L2 distance (in latent space) between the inserted-vehicle regions of the predicted and the condition video.

The iterative refinement process works as follows. First, the agent applies diffusion with relatively high denoising strength, which empirically produces photorealistic results. The agent then reviews the output. If appearance is compromised (e.g. key parts are missing, shape/color changes), it increases the L2 guidance weight. Conversely, if the inserted vehicle appears unrealistic (e.g., lighting mismatches the environment), it increases the denoising strength. This process continues until both photorealism and appearance preservation are satisfied, or the maximum iteration number is reached, as shown in Figure 3.

## 4 Experiments

### 4.1 Evaluation Metrics

We evaluate our method on the following five aspects. 1) **Photorealism.** We use *FID* [15] to assess image realism, and *FVD* [39] to evaluate temporal consistency and overall video quality. 2) **Instruction Alignment.** We measure instruction alignment using the following two metrics. For *Appearance Alignment* metric, we sample frames from both the original and edited videos. We then use the VLM [17] to compare them and determine whether the target object has been accurately deleted, inserted, or replaced. For *Behavior alignment* metric, we first use the Grounded-SAM-2 [28] model to track the edited object, and then back-project its trajectory from pixel coordinates to world coordinates. Finally, based on the map information, we evaluate whether the trajectory matches the instructions. 3) **Structure Preservation.** Following [20], we use the self-similarity matrix from DINO [6] to capture the structural information of images. We then compare the difference between the matrices of the original and edited images.


**Fig. 4: Qualitative comparison with baselines**. The results generated by Cosmos [3] fail to align with the instruction and do not preserve the background well. ChatSim [41] produces editing results with poor visual quality, inaccurate trajectories, and collision issues. Our method clearly outperforms them in photorealism, instruction alignment, structure preservation, and traffic realism.

4) **Traffic Realism.** The generated trajectories should not violate traffic rules. Therefore, we also report the *Collision Rate* and *Off-road Rate* by sampling frames from edited videos and using the VLM [17] to detect such incidents. 5) **User Study.** For all four aspects mentioned above, we also conduct human evaluation with 26 participants. For each aspect, participants are asked to select which method performs best among the three methods and "none".

### 4.2 Experiment Settings

***Dataset and Instructions.*** We curate 30 diverse scenes from a real-world driving dataset (Waymo Open Dataset [35]), covering different times of day, road types, and weather conditions. Our work primarily focuses on vehicle editing in driving scenes, so we select scenes with fewer pedestrians. For test instructions, we generate them using GPT-4 [1], followed by human filtering. For each scene, we generate 4 types of instructions (with 2-3 instructions per type): 1) *removal* (e.g., "remove the blue sedan in front"); 2) *replacement* (e.g., "replace the van on the right with a yellow taxi"); 3) *behavior editing* (e.g., "make the ego vehicle turn left"); 4) *insertion* (e.g., "insert a black SUV 10 meters in front of the ego vehicle and make it change to the right lane").

***Baselines.*** We use both LLM agent-based (ChatSim [41]) and diffusion model-based (Cosmos [3]) driving scene editors as baselines. Both are state-of-the-art open-source methods in their respective categories. To ensure fair comparison, we strictly follow the original settings of each method. We evaluate all methods on the same scene and instruction pairs.

**Table 1: Quantitative comparison with baselines.** Our method consistently outperforms all baselines across all metrics while maintaining efficiency. Abbreviations: App. = Appearance, Beh. = Behavior, Str. = Structure Distance, Col. = Collision, Off. = Off-road. User (%) shows the percentage of user study participants who choose each method as the best for that aspect, with options including the three methods and "none". Editing time is measured in minutes per scene on a single A6000 GPU.

| | Photorealism | | | Instr. Align. | | | Struct. Pres. | | Traffic Realism | | | Efficiency |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | FID ↓ | FVD ↓ | User (%) ↑ | App. (%) ↑ | Beh. (%) ↑ | User (%) ↑ | Str. ↓ | User (%) ↑ | Col. (%) ↓ | Off. (%) ↓ | User (%) ↑ | Time (min) ↓ |
| Cosmos [3] | 33.42 | 797.51 | 34.60 | 46.25 | 32.86 | 10.50 | 74.07 | 19.10 | 3.16 | 3.95 | 35.5 | **16.9** |
| ChatSim [41] | 47.70 | 605.69 | 4.80 | 42.33 | 26.64 | 3.90 | 46.52 | 7.40 | 27.62 | 24.71 | 5.60 | 19.3 |
| **Ours** | **32.85** | **467.20** | **54.20** | **82.19** | **71.67** | **60.40** | **34.62** | **65.00** | **0.58** | **1.73** | **48.30** | 17.1 |

***Implementation Details.*** We use the front camera of scenes for experiments. The input videos are 8 seconds long with an FPS of 10. For the LLM model, we use GPT-4 [1], and for the VLM model, we use GPT-4o [17]. For the behavior and video reviewer agents, we set the maximum iteration number for the feedback loop as 5. For the video diffusion tool, we adopt CogVideoX [46] as the backbone and initialize the model with pretrained weights fromTrajectoryCrafter [47]. We further fine-tune it in two stages: 1) 40k iterations on 33-frame short videos, followed by 2) 20k iterations on 81-frame long videos. Both stages are trained with a batch size of 4 on a 4×H100 GPU workstation, for 48 hours each.

### 4.3 Quantitative Comparison with Baselines

***Editing Performance.*** We report editing performance metrics across four key aspects in Table 1. As can be seen, our method outperforms the baselines across all metrics, particularly in appearance and behavior alignment. In terms of photorealism and structure preservation, our editing results not only achieve the highest visual quality and temporal consistency, but also preserve the original structure well. While Cosmos [3] demonstrates good photorealism on individual frames, it tends to modify the background simultaneously. ChatSim [41], on the other hand, shows poor performance in both visual quality and structure preservation. For instruction alignment, both Cosmos [3] and ChatSim [41] fail to accurately remove, insert, or replace objects, and cannot generate precise trajectories for target behaviors. In contrast, our method substantially outperforms them in both appearance and behavior alignment. Regarding traffic realism, our method effectively models multi-object interactions, thus significantly reducing traffic violations such as collisions and off-road incidents. For user study, our method also greatly outperforms baselines, particularly in instruction alignment and structure preservation.

***Computational Efficiency.*** We also report editing time for each method in Table 1. All methods generate 8-second videos at 10 fps on a single NVIDIA A6000 GPU. Our method requires 17.1 minutes per edit, comparable to baselines (Cosmos: 16.9 minutes, ChatSim: 19.3 minutes). Note that both ChatSim and our method require a one-time 3D reconstruction preprocessing step per scene, which takes around 2 hours. We further analyze per-module timing for our method in Table 2. The object node editing module (dominated by Text-to-3D Tool) and video iterative refinement are the most time-consuming components.

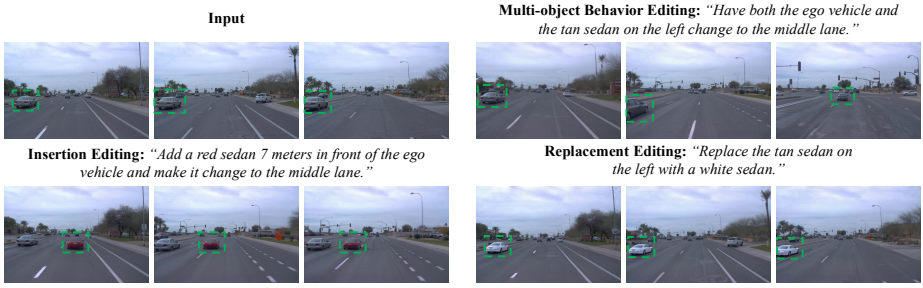


**Fig. 5: Qualitative editing results.** We demonstrate our method's editing capabilities for diverse scenario generation. Note that for better visualization, the timestamps are not strictly aligned.

**Table 2: Per-module timing for our method.** Object node editing and video iterative refinement dominate the runtime. Note that behavior editing time already includes the feedback loop. All timings measured on a single A6000 GPU.

| Ours | Object Query | Object Node Editing | Behavior Editing | Coarse Rendering | Video Iterative Refinement |
|---|---|---|---|---|---|
| Time | 1.4 mins | 6.1 mins | 1.5 mins | 0.5 mins | 7.6 mins |

### 4.4 Qualitative Comparison with Baselines

We provide visual comparison results in Figure 4. The first example involves ego vehicle trajectory editing ("Make the ego vehicle change to the rightmost lane."). We not only achieve accurate ego view changes, but also capture the surrounding environmental lighting information (e.g., realistic highlights and reflections on the vehicle body). In contrast, both Cosmos [3] and ChatSim [41] fail to perform accurate ego vehicle trajectory editing. ChatSim [41] also suffers from poor visual quality (e.g., artifacts in moving objects).

In the second example ("Insert a black sedan 4 meters to the left of ego vehicle, 9 meters ahead, and make it change to the right lane."), we use a more challenging scenario at nighttime. Our editing results show that the inserted vehicle not only executes the lane change accurately, but also seamlessly adapts to the nighttime lighting. Cosmos [3] not only fails to insert the requested vehicle but also adds unrequested pedestrians, completely disregarding the instruction. Although ChatSim [41] correctly inserts a black sedan, the result looks highly unnatural. Moreover, the generated trajectory does not follow the target behavior, and the inserted vehicle collides with existing vehicles, failing to achieve proper multi-vehicle interaction.

### 4.5 Diverse Editing Capabilities

In Figure 5, we show our method's diverse editing capabilities, including multi-object behavior editing, object-level insertion and replacement. The examples illustrate that the object grounding agent reliably localizes the target object nodes based on textual descriptions ("tan sedan on the left"), while the behavior editing module correctly modifies the trajectories of both the ego vehicle and the referenced sedan according to the instructions.

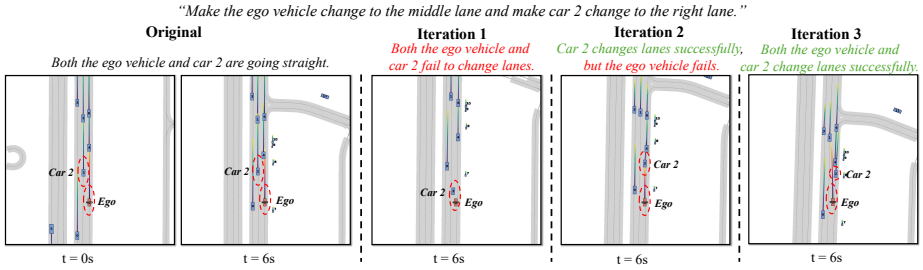


**Fig. 6: Effect of the behavior iterative refinement.** The feedback loop effectively improves the alignment between generated trajectories and instructions while avoiding off-road behavior and collisions.

### 4.6 Ablation Studies

To validate the effectiveness of our behavior and video refinement modules (the behavior reviewer agent and the video reviewer agent), we conduct ablation studies on these two components. For behavior refinement experiments, we use the same 30 test scenes as in Table 1, but generate new test instructions (70 in total). While the behavior editing instructions from Table 1 mostly target single objects, here we create more challenging instructions that specify target behaviors for multiple objects simultaneously. Additionally, during evaluation, we directly evaluate the generated trajectories instead of the edited videos.

**Table 3: Ablation study for iterative behavior refinement.** Overall success indicates that behavior alignment is achieved while avoiding both collisions and off-road behavior. With iterative behavior refinement, the overall success rate of trajectory generation increases significantly.

| Behavior Refinement | Behavior Alignment (%) ↑ | Collision (%) ↓ | Off-road (%) ↓ | Overall Success (%) ↑ |
|---|---|---|---|---|
| ✗ | 54.29 | 35.71 | 30.00 | 34.29 |
| ✓ | **70.00** | **22.86** | **14.29** | **51.43** |

As shown in Table 3, our feedback loop significantly improves trajectory–inst ruction alignment and decreases both off-road and collision cases. We also present a visual comparison in Figure 6 ("Make the ego vehicle change to the middle lane and make car 2 change to the right lane."). In iteration 1, the reviewer checks the generated trajectories and finds that neither the ego vehicle nor car 2 produces target trajectories, so it increases the Classifier-Free Guidance (CFG) weight for both. In iteration 2, the reviewer observes that car 2 now generates the correct trajectory, while the ego vehicle still does not. Therefore, it saves car 2's trajectory as guidance for the next iteration and continues to increase the CFG weight for the ego vehicle. After iteration 3, both ego vehicle and car 2 generate correct trajectories, so the process stops.

Table 4 provides ablation results for the video iterative refinement. To illustrate how video diffusion models may alter the original appearance of inserted vehicles, we use only insertion and replacement instructions in this experiment. The coarse video (row 1) aligns well with instructions but exhibits poor photo-realism. Applying the video diffusion tool once (row 2) for harmonization sig-

**Table 4: Ablation study for iterative video refinement.** With iterative video refinement, both photorealism and appearance alignment can be achieved.

| Video Diffusion Tool | Video Reviewer Agent | FID ↓ | FVD ↓ | Appearance Alignment (%) ↑ |
|---|---|---|---|---|
| ✗ | ✗ | 43.54 | 613.72 | **87.69** |
| ✓ | ✗ | 36.83 | 501.84 | 65.47 |
| ✓ | ✓ | **36.78** | **493.25** | 85.28 |

nificantly improves photorealism but compromises appearance alignment. Our iterative refinement (row 3) achieves both objectives simultaneously.

### 4.7 Hallucinations of Video Diffusion Tool

While video diffusion models can improve realism, they may also introduce hallucinations [2]. To analyze potential hallucinations from our video diffusion tool, we evaluate two additional metrics. 1) We use the 3D Consistency metric from WorldScore [11], which measures geometric consistency via depth-based reprojection error across consecutive frames. 2) We assess road structure preservation using NTL-IoU metric from DriveDreamer4D [49], which computes the mean IoU between predicted 2D lanes and projected ground-truth 3D lanes. We report both metrics before and after video diffusion refinement (i.e., our coarse video v.s. refined video), as well as comparisons against all baselines. As shown in the Table 5, video diffusion can introduce mild hallucinations, e.g., slightly degrading geometric consistency and lane structure. However, these effects are limited, and our method still outperforms all baselines.

**Table 5: Hallucination analysis before and after video diffusion refinement.**

| Metric | Ours (Coarse) | Ours (Refined) | ChatSim [41] | Cosmos [3] |
|---|---|---|---|---|
| 3D Consistency [11] ↑ | 74.07 | 72.64 | 71.32 | 69.85 |
| NTL-IoU [49] ↑ | 52.11 | 50.96 | 50.13 | 48.76 |

## 5 Conclusion

We introduce LangDriveCTRL, a natural-language-controllable framework for editing real-world driving videos, supporting both object-level operations (removal, insertion, and replacement) and multi-object behavior editing. We demonstrate through extensive quantitative and qualitative evaluations that our method simultaneously achieves photorealism, instruction alignment, structure preservation, and traffic realism, significantly outperforming prior methods.

## References


1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Aithal, S.K., Maini, P., Lipton, Z., Kolter, J.Z.: Understanding hallucinations in diffusion models through mode interpolation. Advances in neural information processing systems **37**, 134614–134644 (2024)
3. Ali, A., Bai, J., Bala, M., Balaji, Y., Blakeman, A., Cai, T., Cao, J., Cao, T., Cha, E., Chao, Y.W., et al.: World simulation with video foundation models for physical ai. arXiv preprint arXiv:2511.00062 (2025)
4. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: Learning to follow image editing instructions. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 18392–18402 (2023)
5. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
6. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9650–9660 (2021)
7. Chang, W.J., Zhan, W., Tomizuka, M., Chandraker, M., Pittaluga, F.: Langtraj: Diffusion model and dataset for language-conditioned trajectory simulation. arXiv preprint arXiv:2504.11521 (2025)
8. Chen, Z., Yang, J., Huang, J., de Lutio, R., Esturo, J.M., Ivanovic, B., Litany, O., Gojcic, Z., Fidler, S., Pavone, M., et al.: Omnire: Omni urban scene reconstruction. arXiv preprint arXiv:2408.16760 (2024)
9. Cheng, Y., Li, L., Xu, Y., Li, X., Yang, Z., Wang, W., Yang, Y.: Segment and track anything. arXiv preprint arXiv:2305.06558 (2023)
10. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning. pp. 1–16. PMLR (2017)
11. Duan, H., Yu, H.X., Chen, S., Fei-Fei, L., Wu, J.: Worldscore: A unified evaluation benchmark for world generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 27713–27724 (2025)
12. Hassan, M., Stapf, S., Rahimi, A., Rezende, P., Haghighi, Y., Brüggemann, D., Katircioglu, I., Zhang, L., Chen, X., Saha, S., et al.: Gem: A generalizable ego-vision multimodal world model for fine-grained ego-motion, object dynamics, and scene composition control. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 22404–22415 (2025)
13. He, Y., Ren, X., Tang, D., Zhang, Y., Xue, X., Fu, Y.: Density-preserving deep point cloud compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2333–2342 (2022)
14. He, Y., Tang, D., Zhang, Y., Xue, X., Fu, Y.: Grad-pu: Arbitrary-scale point cloud upsampling via gradient descent with learned distance functions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5354–5363 (2023)
15. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)
16. Hsu, H.Y., Lin, C.H., Zhai, A.J., Xia, H., Wang, S.: Autovfx: Physically realistic video editing from natural language instructions. In: 2025 International Conference on 3D Vision (3DV). pp. 769–780. IEEE (2025)

17. Hurst, A., Lerer, A., Goucher, A.P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al.: Gpt-4o system card. arXiv preprint arXiv:2410.21276 (2024)
18. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4), 139–1 (2023)
19. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al.: Segment anything. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4015–4026 (2023)
20. Li, M., Xie, C., Wu, Y., Zhang, L., Wang, M.: Five: A fine-grained video editing benchmark for evaluating emerging diffusion and rectified flow models. arXiv preprint arXiv:2503.13684 (2025)
21. Li, W., Zhou, R., Zhou, J., Song, Y., Herter, J., Qin, M., Huang, G., Pfister, H.: 4d langsplat: 4d language gaussian splatting via multimodal large language models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 22001–22011 (2025)
22. Liang, R., Gojcic, Z., Ling, H., Munkberg, J., Hasselgren, J., Lin, C.H., Gao, J., Keller, A., Vijaykumar, N., Fidler, S., et al.: Diffusion renderer: Neural inverse and forward rendering with video diffusion models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 26069–26080 (2025)
23. Liang, Y., Yan, Z., Chen, L., Zhou, J., Yan, L., Zhong, S., Zou, X.: Driveeditor: A unified 3d information-guided framework for controllable object editing in driving scenes. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 5164–5172 (2025)
24. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Jiang, Q., Li, C., Yang, J., Su, H., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In: European conference on computer vision. pp. 38–55. Springer (2024)
25. Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.Y., Ermon, S.: Sdedit: Guided image synthesis and editing with stochastic differential equations. arXiv preprint arXiv:2108.01073 (2021)
26. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
27. Qin, M., Li, W., Zhou, J., Wang, H., Pfister, H.: Langsplat: 3d language gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20051–20060 (2024)
28. Ren, T., Liu, S., Zeng, A., Lin, J., Li, K., Cao, H., Chen, J., Huang, X., Chen, Y., Yan, F., et al.: Grounded sam: Assembling open-world models for diverse visual tasks. arXiv preprint arXiv:2401.14159 (2024)
29. Shah, S., Dey, D., Lovett, C., Kapoor, A.: 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 (2017)
30. Shi, J.C., Wang, M., Duan, H.B., Guan, S.H.: Language embedded 3d gaussians for open-vocabulary scene understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5333–5343 (2024)
31. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
32. Song, Z., He, Z., Li, X., Ma, Q., Ming, R., Mao, Z., Pei, H., Peng, L., Hu, J., Yao, D., et al.: Synthetic datasets for autonomous driving: A survey. IEEE Transactions on Intelligent Vehicles **9**(1), 1847–1864 (2023)

33. Strickland, E.: Are Self-Driving Cars Closer Than We Think? Discover How Synthetic Data Is Paving the Way — spectrum.ieee.org. https://spectrum.ieee.org/synthetic-data-self-driving (2025), [Accessed 13-11-2025]
34. Sullivan, B., Kaszynski, A.: PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). Journal of Open Source Software **4**(37), 1450 (May 2019). https://doi.org/10.21105/joss.01450, https://doi.org/10.21105/joss.01450
35. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2446–2454 (2020)
36. Sun, S., Zhuang, B., Jiang, Z., Liu, B., Xie, X., Chandraker, M.: Lidarf: Delving into lidar for neural radiance field on street scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19563–19572 (2024)
37. Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8248–8258 (2022)
38. Tasneem, Z., Dave, A., Singh, A., Tiwary, K., Vepakomma, P., Veeraraghavan, A., Raskar, R.: Decentnerfs: Decentralized neural radiance fields from crowdsourced images. In: European Conference on Computer Vision. pp. 144–161. Springer (2024)
39. Unterthiner, T., Van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., Gelly, S.: Towards accurate generative models of video: A new metric & challenges. arXiv preprint arXiv:1812.01717 (2018)
40. Wei, Y., Wang, J., Du, Y., Wang, D., Pan, L., Xu, C., Feng, Y., Dai, B., Chen, S.: Chatdyn: Language-driven multi-actor dynamics generation in street scenes. arXiv preprint arXiv:2412.08685 (2024)
41. Wei, Y., Wang, Z., Lu, Y., Xu, C., Liu, C., Zhao, H., Chen, S., Wang, Y.: Editable scene simulation for autonomous driving via collaborative llm-agents. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15077–15087 (2024)
42. Wu, J.Z., Zhang, Y., Turki, H., Ren, X., Gao, J., Shou, M.Z., Fidler, S., Gojcic, Z., Ling, H.: Difix3d+: Improving 3d reconstructions with single-step diffusion models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 26024–26035 (2025)
43. Xiong, Y., Zhou, X., Wan, Y., Sun, D., Yang, M.H.: Drivinggaussian++: Towards realistic reconstruction and editable simulation for surrounding dynamic driving scenes. arXiv preprint arXiv:2508.20965 (2025)
44. Xu, R., Lin, H., Jeon, W., Feng, H., Zou, Y., Sun, L., Gorman, J., Tolstaya, K., Tang, S., White, B., et al.: Wod-e2e: Waymo open dataset for end-to-end driving in challenging long-tail scenarios. arXiv preprint arXiv:2510.26125 (2025)
45. Yang, J., Gao, M., Li, Z., Gao, S., Wang, F., Zheng, F.: Track anything: Segment anything meets videos. arXiv preprint arXiv:2304.11968 (2023)
46. Yang, Z., Teng, J., Zheng, W., Ding, M., Huang, S., Xu, J., Yang, Y., Hong, W., Zhang, X., Feng, G., et al.: Cogvideox: Text-to-video diffusion models with an expert transformer. arXiv preprint arXiv:2408.06072 (2024)
47. YU, M., Hu, W., Xing, J., Shan, Y.: Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models. arXiv preprint arXiv:2503.05638 (2025)

48. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 3836–3847 (2023)
49. Zhao, G., Ni, C., Wang, X., Zhu, Z., Zhang, X., Wang, Y., Huang, G., Chen, X., Wang, B., Zhang, Y., et al.: Drivedreamer4d: World models are effective data machines for 4d driving scene representation. In: Proceedings of the computer vision and pattern recognition conference. pp. 12015–12026 (2025)
50. Zhao, G., Wang, X., Zhu, Z., Chen, X., Huang, G., Bao, X., Wang, X.: Drivedreamer-2: Llm-enhanced world models for diverse driving video generation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 10412–10420 (2025)
51. Zhao, Z., Lai, Z., Lin, Q., Zhao, Y., Liu, H., Yang, S., Feng, Y., Yang, M., Zhang, S., Yang, X., et al.: Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. arXiv preprint arXiv:2501.12202 (2025)
52. Zhu, Z., Zou, Y., Jiang, C.M., Sun, B., Casser, V., Huang, X., Wang, J., Yang, Z., Gao, R., Guibas, L., et al.: Scenecrafter: Controllable multi-view driving scene editing. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 6812–6822 (2025)