Scenario Generation for Interactive Urban Environments

Paritosh Sharma, Hui-Yin Wu

December 2025

Project members

- Scientific team: Paritosh Sharma, Hui-Yin Wu

1 Context and objectives

The document highlights the work plan for the the WP4 of the ANR Creative 3D [1] project. The expected outcome of this project is to create a generative model that is capable of creating personalized training scenarios in urban environments.

2 Introduction

Virtual Reality (VR) and Augmented Reality (AR) technologies have advanced significantly in recent years, enabling the creation of immersive and interactive environments for various applications. These can be further used to provide personalized training and rehabilitation scenarios. In the context of low vision rehabilitation, these models can be particularly useful to study pedestrian behaviours under normal and simulated vision. However, most simulated environments suffer from perceptual gaps between the designer, the user, and the system. The existing GUsT-3D framework [61], developed during the Creative3D project, provides a foundation to address this. However, the current framework is still limited by its reliance on a fixed set of urban environments and interactions innhibiting its ability to scale.

In parallel, recent works on 3D generative models for urban scenario generation has shown promising results in generating diverse and realistic urban environments. Additionally, these models have been able to capture the diverse nature and the complexities of an urban setting, including the interactions between pedestrians, vehicles, and the environment. Driven by these extraordinary capabilities, exploring the potential of generative models will enable us to create more diverse and realistic urban scenarios.

3 Related Work

In this section, we first start by reviewing prior work on pedestrian-in-the-loop simulations. Then, we review recent works on generative models for urban scenario generation as well as diffusion-based

[1] `https://project.inria.fr/creattive3d/`

4D generation. Finally, we discuss common validation methods used to evaluate the performance of these models.

3.1 Pedestrian-in-the-Loop

VR Simulations have been widely used to study pedestrian behavior and interactions in urban scenarios [60, 54, 38, 50]. This human-in-the-loop approach (also referred as pedestrian-in-the-loop[22]) allows researchers to collect data on how pedestrians interact with their environment, including their decision-making processes, movement patterns, and responses to various stimuli. These simulations can be used to study a wide range of scenarios, from simple road crossings to complex urban environments with multiple interacting agents. More recently, JaywalkerVR [38], a VR human-in-the-loop simulator, used CARLA [12] to create four different scenarios: jaywalking, parked cars, four-way stops, and parking lot entrances. Despite the obvious advantages, developing such simulators is still challenging because of the perceptual gaps between the designer, the user and the system as identified by Dourish [13].

The GUsT-3D framework [61] addresses this by first defining the scene using a scenegraph, which captures the relationships between different elements in the scenario (ontology) and then defining the task to be carried out during the course of the scenario using a GUTasks (intersubjectivity). Lastly, it uses a query component for logging and post-scenario analysis of the experience (intentionality). This framework was also applied by creating a dataset of 6 road-crossing scenarios to study pedestrian behavior under normal and low vision [62]. Even though GusT-3D framework addresses the perceptual gaps which were identified by Dourish, it still relies on a fixed set of scenarios and interactions.

3.2 Generative Models for Urban Scenario Generation

3.2.1 Image-based Generation

Diffusion models have shown impressive results in generating high-quality images from textual descriptions. Models like DALL-E 2 [45], Imagen [48], and Stable Diffusion [46] have demonstrated the ability to create diverse and realistic images based on text prompts. Recent works such as GameNGen [55], Genie [6], and DIAMOND [1] have shown the potential of diffusion models to generate game environments in real-time. However, these models suffer from latency issues since they are not truly 3D and treat the rendering process as a conditional markov process i.e. generate the next frame based on the previous frame.

3.2.2 Prompts

Prompts are textual cues provided to the generative model to guide the scenario creation process. Since the popularity of LLMs like GPT-3, prompts have become a common way to interact with generative models. They can be used to specify the desired characteristics of the scenario, such as the type of environment, objects, and their relationships. Table 1 lists some of the recent prompt-based models for scenario generation.

3.2.3 Layout-guided Generation

Layouts are structured representations of scenarios that capture the spatial relationships between different elements, such as buildings, roads, and pedestrians. These may include bird's-eye views,

Model	Technique	Output
ScenicNL [14]	Converts LLM Prompts to Scenic Constraints	Scenic Scenario
ChatScene [73]	Conversational Agent for Scenario Definition using Scenic	Scenic Scenario
LayoutGPT [15]	Prompts converted to CSS-like Layout Formatting by LLMs	Layout Representation
ChatDyn [57]	LLM-based planning and low-level trajectory generation for Pedestrian and Vehicle	3D Scenario
Work by Feng et al. [16]	JSON to describe layout and 3D models	3D Scene
TTSG [47]	LLM-based planning and retrieval	3D Scenario
3D-SceneDreamer [75]	Prompt to point-of-view image followed by image to 3D	3D Scene
GraphCanvas3D [33]	Uses LLM to create a scenegraph and then a 3D scene	3D Scene
Scenethesis [32]	Uses LLM to create a scenegraph and then a 3D scene	3D Scene
SceneX [78]	LLM to plan PCG (Procedural Controllable Generation)	3D Scene
Surreal Drivers [25]	Chain-of-thought prompts	3D Scene
Text2nerf [72]	Text prompts	3D Scene
X-Scene [69]	Text prompts	3D Scene

Table 1: Prompt-based Models for Urban Scenario Generation

top-down maps, or other forms of spatial representations that provide a high-level overview of the scenario. Table 2 lists some of the recent layout-based models for scenario generation.

Model	Technique	Output
CC3D [2]	2D Layout-based 3D Scene Generation	3D Scene
CityDreamer4D [65]	Uses a Layout Generator and a traffic scenario generator	3D Scenario
Infinicube [37]	Text prompts, HD Maps and Bounding Boxes	3D Scenario
Work by Zhang et al. [74]	BEV map	3D Scene
UniScene [27]	BEV map	Multi-view video
Savkin et al. [49]	Scenegraph	Scenario Image

Table 2: Layout-based Models for Urban Scenario Generation

Additionally, some works have also combined multimodal inputs with different types of data, prompts, layouts and other structured representations, to provide a richer context for scenario generation. Table 3 lists some of the recent multimodal models for scenario generation.

Model	Input Type	Output
CityX [76]	Prompt, OSM file or Semantic Map	3D Scenario
CityCraft [11]	Layout data and text prompts	3D Scene
Work by Liu et al. [34]	Scenegraph assisted using text prompts	3D Scene
GAUDI [4]	Conditioning using prompts or point-of-view images	3D Scene
MagicDrive3D [19]	Text prompts, Bird Eye View (BEV) map and 3D Bounding Boxes	Reconstructed 3D video
Scene123 [70]	Text prompt, point-of-view image or Text Description	3D Scene
StreetScapes [10]	BEV and height map with support for prompts	Video
Urban Architect [36]	3D Layout and Text Prompts	3D Scene
Urban World [51]	Layout (generation) and prompts (refinement)	3D Scenario
Wonderplay [30]	point-of-view image and action (physics)	3D Video

Table 3: Multimodal Models for Urban Scenario Generation

3.3 Validation

Table 4 lists common qualitative and quantitative validation methods used to evaluate the performance of generative models for urban road-crossing scenarios. These methods assess the quality, realism, and diversity of generated scenarios.

Table 4: Qualitative and Quantitative Evaluation Methods for Road-Crossing Scenarios

Method Type	**Evaluation Approach**	**Description**	**Use Case**
Qualitative	Human Review	Human experts assess realism, scenario diversity, and layout plausibility.	Validates human-perceived quality and applicability of the scene.
	Scenario Visualization	3D visual inspection or rendered videos showing pedestrian, vehicle, and environment interactions.	Helps detect unrealistic or unnatural behavior/configurations.
	Surveys	Collects subjective feedback on realism, perceived difficulty, or stress levels from participants.	Measures human-centric realism or emotional response.
	Comparison	Compare scenario features (traffic density, gap opportunities, layout) to real-world statistics or distributions.	Validates realism by matching key scenario characteristics to empirical data.
	Failure Cases	Identification and analysis of implausible or unsafe scenarios generated by the model.	Guides iterative model improvement.
Quantitative	Scenario Feature Statistics	Statistical analysis of scenario properties (traffic speed, number/duration of gaps, crosswalk presence, etc.).	Ensures generated scenarios span realistic distributions.
	Coverage and Diversity Metrics	Measures distributional entropy or spread of key attributes across all generated scenarios.	Assesses generalizability, scenario variety, and model's exploration of edge cases.
	Criticality and Opportunity Metrics	Quantifies the frequency and severity of challenging situations (number of safe gaps, minimum feasible gap size, "no-go" cases).	Evaluates risk/challenge spectrum in the scenario catalog.
	Sim2Real Gap (Domain Distance)	Computes metrics like KL-divergence, Earth Mover's Distance, t-SNE or FID/KID/CKL between generated and real scenario feature distributions.	Evaluates how closely synthetic scenarios match reality.
	Controllability Metrics	Measures how well the model can generate scenarios based on prompts (CLIP, BLIP, VQA, etc.)	Assesses controllability and flexibility of the generative model.

3.4 Code

This section contains the existing tested SOTA code repositories.

3.4.1 Scenic

Easy to setup and run on colab since the repository is well maintained.

3.4.2 MagicDrive

Had issues running due to model size and GPU memory limitations. Also realised only 2D video generation works for now since the code is not uploaded yet for 3D generation.

3.4.3 MetaUrban/ScenarioNET

Working colab provided.

3.4.4 Threestudio

Threestudio [35] is a collection of 3D generative techniques. Easy to setup and run since docker container is provided. It is well-maintained by the community and has a good documentation. Thus, good to test / refer different generative techniques for 3D.

3.4.5 City Dreamer

CityDreamer4D [65] tried testing the 3D branch with static scenes but encountered issues with the docker container.

3.4.6 Urban Architect

Urban Architect [36] - Working. Can generate semantic maps and depth maps from a layout input. Which then can be used to generate the 3D scene. However, the code has to be adapted to generate using multiple GPUs due VRAM limitations.

3.4.7 CLIP

CLIP [43] is a model that can be used to evaluate the quality of generated scenarios. Tested on colab with a pretrained ViT model on ImageNET.

3.4.8 GraphDreamer

GraphDreamer [18] works but generation takes over 30 hours for 1 scenegraph. However, it cannot generate complex scenes with more than 3 nodes in the scene graph.

4 Problem and Open Question

While recent generative models have shown strong capabilities in synthesizing urban scenarios, they remain limited in their ability to model pedestrian dynamics within these environments. In particular, they struggle to establish a coherent link between spatio-temporal control over objects and the range of possible pedestrian interactions—an aspect that is essential for simulating realistic traffic scenarios.

This leads to the following open research question:

How can we leverage natural language–conditioned generative models to produce urban layouts with precise spatiotemporal control over objects and their interactions?

5 Method

The following section describes the proposed method for generating the urban scenarios.

5.1 Input and Splitting

Since no existing datasets provide paired text prompts with corresponding urban layouts and tasks, we propose leveraging the capabilities of LLMs to parse and structure input prompts. Similar approaches have been explored in prior works, such as ChatDyn [57] for action planning nd LayoutGPT [15] for layout generation.

We first process the input prompt and split it into the layout and tasks. The classes in the layout are based on the KITTI-360 [31] dataset. Here is an example of how the input prompt can be split:

Scene Specification Format

Prompt:
You are an urban scenario planning assistant. For the following urban scene prompt:

[Urban scene description]

Generate a JSON object containing the following fields:

- **static_layout:**
  A $20m \times 20m$ 2D layout map represented as an array of objects. Each object must include:
  - `id`: unique identifier
  - `type`: one of {`pole`, `traffic sign`, `smallpole`, `lamp`, `trash bin`, `ground`, `road`, `sidewalk`, `parking`, `building`, `garage`, `fence`, `gate`, `vegetation`, `terrain`, `rail track`, `wall`, `box`, `vending machine`, `traffic light`, `rider`, `bicycle`, `motorcycle`, `motorbike`, `car`, `truck`, `bus`, `van`, `trailer`, `caravan`, `person`}
  - `position`: (x, y) coordinates in meters within the 2D plane
  - `orientation`: angle in degrees (0–360)
  - `size`: width and height in meters
  - `layer`: Layer classification for interactive scenarios. Choose from the default layers:
    - **Interactive layers:**
      - `"Container"`: Objects that can contain or hold other items (e.g., trash bins, boxes, bags)
      - `"Support"`: Surfaces that can support objects being placed on them (e.g., tables, shelves, building roofs)
      - `"Movable"`: Objects that can be moved or picked up (e.g., cars, people, bicycles, small objects)
      - `"Interactable"`: Objects users can interact with or activate (e.g., traffic lights, buttons, switches, doors)
    - **Navigation layers:**

        - `"Ground"`: Walkable surfaces (e.g., road, sidewalk, parking, ground)
        - `"Wall"`: Vertical barriers (e.g., building walls, fences, barriers)
        - `"Entryway"`: Passages and access points (e.g., doors, gates, entrances)
    - **Environment layers:**
        - `"Light"`: Light sources (e.g., lamps, street lights)
        - `"Camera"`: Viewpoints and surveillance (e.g., security cameras)
        - `"State_object"`: Objects with changing states (e.g., animated elements)
        - `"Animated"`: Objects with built-in movement (e.g., fountains, flags)

  Use default layers unless scenario requirements need custom combinations.

- **dynamic_layout:**
  - `trajectories`: Array of dynamic objects representing movement over time. Each object must include:
    - `id`: Unique identifier
    - `type`: One of {`rider`, `bicycle`, `motorcycle`, `motorbike`, `car`, `truck`, `bus`, `van`, `trailer`, `caravan`, `person`}
    - `initial_position`: (x, y) at time $= 0$
    - `trajectory_description`: Array of states; each state is of the form {`"time": t, "position": [x, y]`} (with t normalized between 0 and 1)
    - `layer`: Layer classification (usually `"Movable"` for dynamic objects)

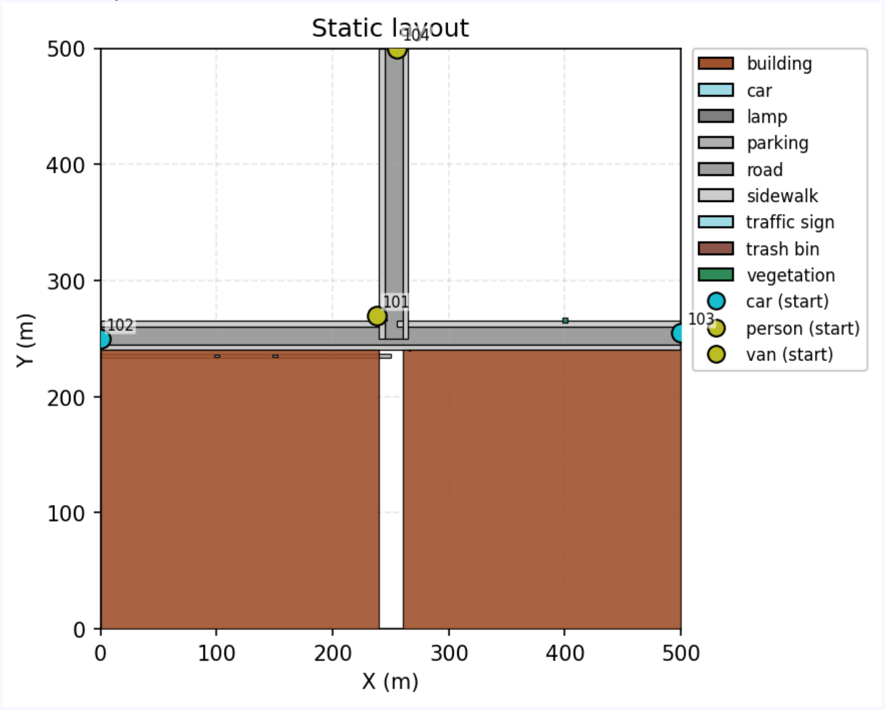
- **tasks:**
  Before creating tasks, review the static_layout objects above.
  - Task targets must reference actual object `"type"` values in static_layout.
  - Ordered list of guided actions for the pedestrian; each task must include:
    - `id`
    - `goal`: {`type, target, key_item`}
        - `type`: One of [`"get"`, `"interact"`, `"interactWith"`, `"go"`, `"place_in"`, `"place_on"`] with exact definitions and required object layers:
        - `"get"`: Target must be `layer="Movable"`
        - `"interact"/"interactWith"`: Target must be `layer="Interactable"`
        - `"go"`: Target must be `layer="Ground"`
        - `"place_in"`: Target must be `layer="Container"`
        - `"place_on"`: Target must be `layer="Support"`
        - `target`: Must match the exact `"type"` of an existing object in static_layout with the correct layer.
        - `key_item`: Optional item needed (null if not needed)
    - `constraints`: {`precedence`, `evaluation`}
    - `help`: {`baseline`, `target`, `failure`}

- ∗ failure_condition: {type, item}
- ∗ instructions: {text_en, text_fr}
- ∗ trigger: {object_id, function}

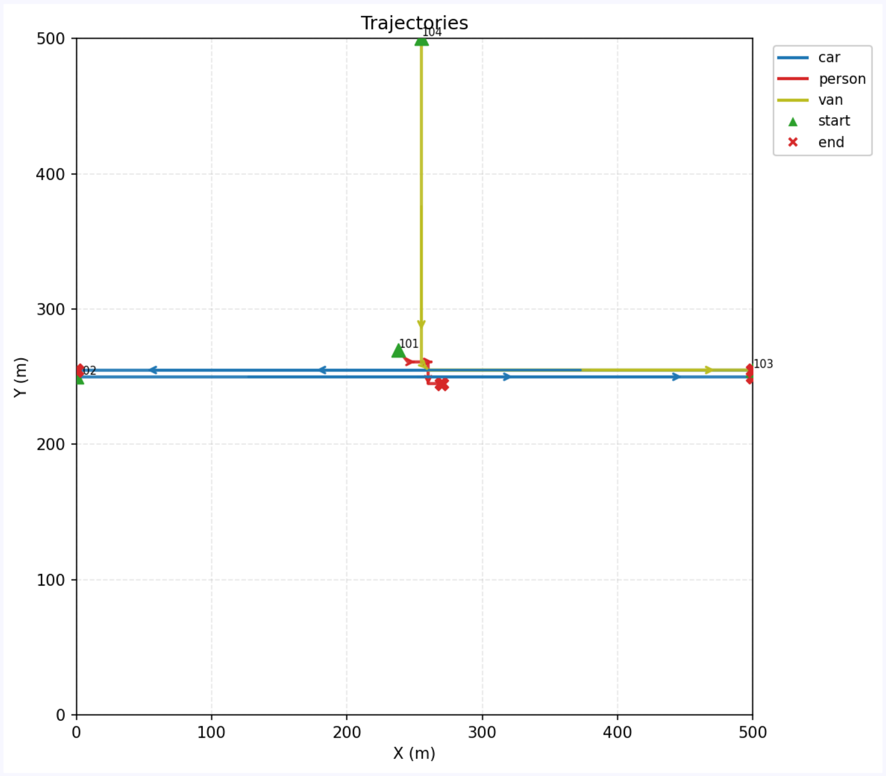
Input Example:

Afternoon at a three-way intersection with cars parked on the side. Cars arriving from all sides. Agent must yield and check for cars before crossing.

Output Example:
Static layout:



Trajectories:



Tasks:

- Walk from your starting position at [275, 275] to the edge of the crosswalk at [275, 260].
- Wait for the pedestrian signal to allow crossing the horizontal road.
- Cross the street to the opposite sidewalk, arriving at [275, 240].

JSON for the above available in appendix section 9.1

5.2 Tasks

The tasks predicted by the MLLM are a sequence of guided actions to be performed by the player or agent in the environment. Each task is defined as a structured object, similar to the following JSON format:

- **id:** a unique identifier for the task.
- **goal:** specifies the type of action, the target object, and optionally a key item needed to perform the action.
  - `type` – one of the following:

        * `get`: locate and pick up an object (target must have layer "Movable").
        * `interact`: interact with an object (target must have layer "Interactable").
        * `interactWith`: interact with a specific object using a key item (target has layer "Interactable").
        * `go`: navigate to a location (target must have layer "Ground").
        * `place_in`: place an object inside a container (target has layer "Container", requires `key_item`).
        * `place_on`: place an object on a support surface (target has layer "Support", requires `key_item`).
    - `target` – the identifier of the target object, using the GUST annotation format:
        * Ground objects (layer "Ground"): `objecttype_x.0_y.0`, e.g., `sidewalk_50.0_62.0`, `road_50.0_50.0`.
        * Other objects: `objecttype_id`, e.g., `traffic_light_4`, `table_6`, `flower_pot_8`.
    - `key_item` – optional object required to perform the task (null for simple tasks, required for `place_in`, `place_on`, and `interactWith` tasks).

- **baselineTime:** time in seconds that a baseline user should take to complete the task (typically 10–30 seconds for simple tasks, 30–60 seconds for complex tasks).
- **targetTime:** maximum time in seconds allowed for the user to complete the task (typically 1.5–2$\times$ `baselineTime`).
- **Help system:** provides progressive assistance to the user at different stages.
    - `baselineHelp`: type of help provided after `baselineTime` elapses ("text", "path", or "nothing").
    - `targetHelp`: type of help provided after `targetTime` elapses ("text", "path", or "nothing").
    - `failureHelp`: type of help provided after task failure ("text", "path", or "nothing").
    - `baselineText`: English help text shown after `baselineTime` (e.g., "Walk straight along the sidewalk").
    - `targetText`: English help text shown after `targetTime` (e.g., "Reach the sidewalk safely").
    - `failureText`: English help text shown after failure (e.g., "Stay on designated walking areas").
- **Failure conditions:** defines when and how the task fails.
    - `failure`: type of failure condition ("collision" or "none").
    - `failureItem`: object that causes failure when collided with (e.g., "vehicle", "person"; null if `failure` is "none").
- **constraints:** includes task ordering and evaluation criteria.
    - `precedence`: list of task IDs that must be completed before this task can begin.

  - `evaluation`: condition that defines successful task completion (e.g., "on_sidewalk", "item_picked_up").

- **instructions:** natural language instructions for the player in multiple languages.
  - `text_en`: English instructions (e.g., "Walk to the sidewalk at position [50, 62]").
  - `text_fr`: French instructions (e.g., "Marchez vers le trottoir à la position [50, 62]").
- **trigger:** specifies the object and function that triggers task completion.
  - `object_id`: ID of the object from `static_layout`.
  - `function`: trigger function name (e.g., "on_reach", "on_interact", "on_pickup").

The structure is based on the existing GUsT-3D framework [61] and is integrated with the existing system to provide guided task execution with real-time feedback and progressive assistance.

6 Implementation

The implementation of the system is divided into two components — the **Backend (Python)** and the **Frontend (Unity)** — which together enable interaction between the LLM and the Unity-based scenario generation environment.

6.1 Backend (Python) Implementation

The backend is implemented as an HTTP server in Python that interfaces with the LLM to process requests and manage data flow between the model and the Unity client. It exposes four primary `POST` endpoints:

- **/get_json** – Generates the initial layout or configuration in JSON format based on user input or scene specifications.
- **/get_3d** – Produces 3D representations or assets.
- **/get_tasks** – Provides task-level metadata or scenario instructions to GUsT-3D

6.2 Frontend (Unity) Implementation

The frontend is developed as a part of existing GUsT-3D add-on. It communicates with the Python backend via HTTP, sending requests to the defined endpoints and parsing the responses to generate the scenarios within Unity.

7 Experiments

This section presents a log of all experiments conducted to finetune the *LLM*. Detailed evaluations of the experiments are available here for reference.

7.1 Experiment 1

In this experiment, the layout generation tasks is treated as a bounding box json generation task using the Qwen3-14B model [68]. The expected bounding box json structure is as follows:

```
{
  "static_layout": [
    {"type": "road", "bbox": [x_min, y_min, x_max, y_max]},
    {"type": "sidewalk", "bbox": [x_min, y_min, x_max, y_max]},
    ...
  ]
}

For simplicity and reducing the token size, only the static layout is considered for this experiment


\begin{verbatim}
"messages": [
  {
    "role": "user",
    "content": [
      {"type": "text", "text": "Generate a scenario with <Class List>"},
      {"type": "image", "text": "<PIL.Image>"}
    ]
  },
  {
    "role": "assistant",
    "content": [
      {"type": "json", "json": "<JSON structure>"}
    ]
  }
],
```

Parameters:

1. Quantization: 4 bit
2. Rank of LoRA: 16
3. Alpha of LoRA: 32
4. Learning Rate: 5×10^{-6} (adaptive)
5. Batch Size: 1 per GPU (4 GPUs used)
6. Number of Steps/Epochs: 2
7. Modules of the transformer finetuned: [”q_proj”, ”k_proj”, ”v_proj”, ”o_proj”, ”gate_proj”, ”up_proj”, ”down_proj”]

Results:

1. Final train/eval loss of the LLM: 0.670500 / 0.656
2. The generation performance of the model drops compared to the pretrained model.

7.2 Experiment 2

In this experiment, the layout generation tasks is treated as a polygon json generation task using the Qwen3-14B model [68]. The expected bounding box json structure is as follows:

```
{
  "static_layout": [
    {"type": "road", "polygon": [(x_1, y_1), (x_2, y_2), ...]},
    {"type": "sidewalk", "polygon": [(x_1, y_1), (x_2, y_2), ...]},
    ...
  ]
}
```

For simplicity and reducing the token size, only the static layout is considered for this experiment. In this experiment, the model is finetuned on 1/10th (7500) of the KITTI-360 [31] semantic maps dataset. We use the class list and the corresponding semantic map polygon representation pairs as the dataset similar to previous experiment for finetuning:

Parameters:

1. Quantization: 4 bit
2. Rank of LoRA: 16
3. Alpha of LoRA: 32
4. Learning Rate: 5×10^{-6} (adaptive)
5. Batch Size: 1 per GPU (4 GPUs used)
6. Number of Steps/Epochs: 2
7. Modules of the transformer finetuned: [”q_proj”, ”k_proj”, ”v_proj”, ”o_proj”, ”gate_proj”, ”up_proj”, ”down_proj”]

Results:

1. Final train loss of the LLM: 0.637200.
2. The generation performance of the model drops compared to the pretrained model similar to the previous experiment.
3. The model struggles to generate complete jsons.

7.3 Experiment 3

The previous experiments degraded the generation performance of the model compared to the pretrained model. One hypothesis is that the model gets confused between similar instructions for different layouts. Thus, in this experiment, we take a smaller subset of layouts (1227 maps) with unique permutations of classes to reduce confusion during training.

Parameters:

1. Quantization: 4 bit
2. Rank of LoRA: 16
3. Alpha of LoRA: 32
4. Learning Rate: 5×10^{-6} (adaptive)
5. Batch Size: 1 per GPU (4 GPUs used)
6. Number of Steps/Epochs: 2
7. Modules of the transformer finetuned: ["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"]

Results:

1. Final train loss of the LLM: 0.6066
2. The generation performance of the model still drops compared to the pretrained model.
3. The drop in performance is less compared to previous experiments.

7.4 Experiment 4

Since the model's performance improved in the previous experiment, this experiment is a continuation of the previous one, but with a higher epoch count.

Parameters:

1. Quantization: 4 bit
2. Rank of LoRA: 16
3. Alpha of LoRA: 32
4. Learning Rate: 5×10^{-6} (adaptive)
5. Batch Size: 1 per GPU (4 GPUs used)
6. Number of Steps/Epochs: 7
7. Modules of the transformer finetuned: ["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"]

Results:

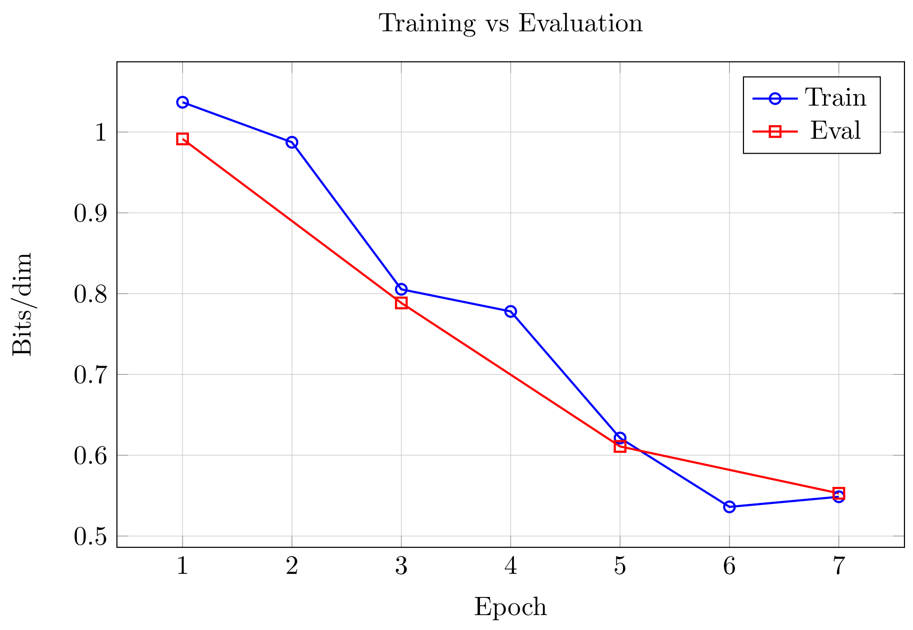


Figure 1: Loss curves for Experiment 4

1. Final train/eval loss of the LLM: 0.5486/0.5529. Loss curves shown in figure 1
2. The model generalises better than the baseline qualitatively.
3. Quantitatively, the CKL is still higher due to incomplete JSONs. The FID and KID are lower.

8 Evaluation

8.1 Quantitative Evaluation

To assess the fidelity and diversity of the generated results, we employ three widely used quantitative metrics: the Fréchet Inception Distance (FID), the Kernel Inception Distance (KID), and the Kullback–Leibler divergence Loss (CKL). Each metric captures different aspects of distributional similarity between the generated and reference data.

8.1.1 Fréchet Inception Distance (FID)

The Fréchet Inception Distance [23] measures the similarity between the real and generated data distributions in the feature space of a pretrained Inception network. It models both distributions as multivariate Gaussians and computes their distance as:

$$\mathrm{FID} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}), \tag{1}$$

where (μ_r, Σ_r) and (μ_g, Σ_g) denote the mean and covariance of the real and generated features, respectively. Lower FID scores indicate closer alignment between the two distributions, corresponding to higher visual fidelity and realism.

8.1.2 Kernel Inception Distance (KID)

The Kernel Inception Distance [5] offers a similar objective to FID but estimates the squared Maximum Mean Discrepancy (MMD) using polynomial kernels. Unlike FID, KID provides an unbiased and more stable estimator for smaller sample sizes. It is defined as:

$$\text{KID} = \mathbb{E}_{x,x'}[k(f(x), f(x'))] + \mathbb{E}_{y,y'}[k(f(y), f(y'))] - 2\mathbb{E}_{x,y}[k(f(x), f(y))], \tag{2}$$

where $k(\cdot, \cdot)$ denotes the kernel function (typically a third-degree polynomial), and $f(\cdot)$ represents the feature extractor. Lower KID values imply better perceptual alignment between generated and real samples.

8.1.3 Classwise Kullback–Leibler Divergence (CKL)

The Classwise Kullback–Leibler divergence (CKL) measures the divergence between the classwise distributions of generated samples and the ground truth, capturing how well the model preserves dependencies:

$$\text{CKL} = \mathbb{E}_{x \sim P_{\text{data}}} \left[D_{\text{KL}} \big(P_{\text{data}}(y|x) \| P_{\text{gen}}(y|x) \big) \right], \tag{3}$$

where $P_{\text{data}}(y|x)$ and $P_{\text{gen}}(y|x)$ denote the classwise distributions of the reference and generated data, respectively. Lower CKL values indicate better fidelity in modelling conditional relationships.

8.2 Qualitative Evaluation

For our experiments, we use Qwen3 to generate the 3D scene layouts and tasks based on user prompts. We use the KITTI-360 dataset [31] to train our multinomial diffusion model for refining the LLM generated semantic maps and trajectories.

8.3 Prompts

Based on existing SOTA in prompt engineering, we assess the MLLM along two complementary axes: prompt specificity and scenario plausibility. This framework is informed by SOTA practices in which prompts are structured to test the model across a spectrum of detail starting from vague to detailed, structured guideline (rubric).

- **Prompt specificity:** Measures the model's robustness to variations in instruction detail [39, 66].
  - *Vague prompts:* Underspecified instructions for generating urban scenarios, where the model must infer missing details such as city layout, traffic patterns, or population distribution. For example, "Generate an urban scenario with streets and buildings" leaves most design aspects open to the model.

  - *Criteria Specific prompts:* Prompts that specify the desired properties or evaluation criteria for the urban scenario, without detailing exact layouts or steps. For instance, "Generate an urban scenario that is realistic, has safe traffic flow, and includes diverse building types" guides the model toward satisfying these properties while leaving implementation and arrangement open.
  - *Detailed prompts (Rubric):* Fully specified instructions with explicit objectives, constraints, and optional step-by-step guidance for urban scene generation. This may include a rubric enumerating required features, such as "Generate an urban scenario with at least three traffic intersections with controlled signals, a sidewalk along the road and a crosswalk every 50 metres, and pedestrian pathways connecting all major areas. Agent arrives from the north, presses on the pedestrian light and then crosses the road when green." Optional Chain-of-Thought instructions can guide stepwise reasoning for layout design.

- **Scenario plausibility:** Evaluates the realism, logical coherence, and potential risk of the model-generated tasks [41, 80].
  - *Illogical scenarios:* Contain contradictory or impossible instructions to probe reasoning limitations . Example, "Create a scenario with no roads with busy traffic. Agent must cross using the crosswalk."
  - *Realistic scenarios:* Plausible, safe instructions representing typical interactions. Example, "Create a scenario with multiple cars and bicycles. Agent must wait for the pedestrian signal before crossing."
  - *Critical scenarios:* Edge-case or high-risk tasks revealing latent model limitations. Example, Jaywalking, Parking lot navigation, etc.

Prompt Grid Design To systematically explore model behavior, each task is tested using prompts that span the two axes. Conceptually, this forms a 2D evaluation grid:

Prompt Specificity	Illogical	Realistic	Critical
Vague	P1	P2	P3
Detailed	P4	P5	P6

Here, P1–P6 represent prompts designed for each combination of specificity and scenario plausibility.

8.3.1 User studies

Appendix

9 Examples

9.1 Layout JSON

```
{
  "static_layout": [
    {
      "id": 1,
      "type": "road",
      "position": [
        250,
        250
      ],
      "size": [
        500,
        20
      ],
      "orientation": 0,
      "layer": "Ground"
    },
    {
      "id": 2,
      "type": "road",
      "position": [
        250,
        250
      ],
      "size": [
        20,
        500
      ],
      "orientation": 90,
      "layer": "Ground"
    },
    {
      "id": 3,
      "type": "sidewalk",
      "position": [
        250,
        230
      ],
      "size": [
        500,
        10
  ],
  "orientation": 0,
  "layer": "Ground"
},
{
  "id": 4,
  "type": "sidewalk",
  "position": [
    250,
    270
  ],
  "size": [
    500,
    10
  ],
  "orientation": 0,
  "layer": "Ground"
},
{
  "id": 5,
  "type": "sidewalk",
  "position": [
    230,
    250
  ],
  "size": [
    10,
    500
  ],
  "orientation": 90,
  "layer": "Ground"
},
{
  "id": 6,
  "type": "sidewalk",
  "position": [
    270,
    250
  ],
  "size": [
    10,
    500
  ],
  "orientation": 90,
  "layer": "Ground"
},
{
  "id": 7,
  "type": "building",
  "position": [
    50,
    50
  ],
  "size": [
    100,
    100
  ],
  "orientation": 0,
  "layer": "Wall"
},
{
  "id": 8,
  "type": "building",
  "position": [
    450,
    50
  ],
  "size": [
    100,
    100
  ],
  "orientation": 0,
  "layer": "Wall"
},
{
  "id": 9,
  "type": "building",
  "position": [
    50,
    450
  ],
  "size": [
    100,
    100
  ],
  "orientation": 0,
  "layer": "Wall"
},
{
  "id": 10,
  "type": "building",
  "position": [
      450,
      450
    ],
    "size": [
      100,
      100
    ],
    "orientation": 0,
    "layer": "Wall"
  },
  {
    "id": 11,
    "type": "traffic light",
    "position": [
      240,
      220
    ],
    "size": [
      2,
      5
    ],
    "orientation": 0,
    "layer": "Interactable"
  },
  {
    "id": 12,
    "type": "traffic light",
    "position": [
      260,
      280
    ],
    "size": [
      2,
      5
    ],
    "orientation": 180,
    "layer": "Interactable"
  },
  {
    "id": 13,
    "type": "traffic light",
    "position": [
      220,
      260
    ],
    "size": [
    2,
    5
  ],
  "orientation": 270,
  "layer": "Interactable"
},
{
  "id": 14,
  "type": "traffic light",
  "position": [
    280,
    240
  ],
  "size": [
    2,
    5
  ],
  "orientation": 90,
  "layer": "Interactable"
},
{
  "id": 15,
  "type": "vegetation",
  "position": [
    150,
    150
  ],
  "size": [
    5,
    5
  ],
  "orientation": 0,
  "layer": "Environment"
},
{
  "id": 16,
  "type": "vegetation",
  "position": [
    350,
    150
  ],
  "size": [
    5,
    5
  ],
  "orientation": 0,
    "layer": "Environment"
  },
  {
    "id": 17,
    "type": "vegetation",
    "position": [
      150,
      350
    ],
    "size": [
      5,
      5
    ],
    "orientation": 0,
    "layer": "Environment"
  },
  {
    "id": 18,
    "type": "vegetation",
    "position": [
      350,
      350
    ],
    "size": [
      5,
      5
    ],
    "orientation": 0,
    "layer": "Environment"
  },
  {
    "id": 19,
    "type": "ground",
    "position": [
      250,
      250
    ],
    "size": [
      20,
      20
    ],
    "orientation": 0,
    "layer": "Ground"
  }
],
"dynamic_layout": {
```

```
"trajectories": [
  {
    "id": 101,
    "type": "car",
    "initial_position": [
      100,
      250
    ],
    "layer": "Movable",
    "trajectory_description": [
      {
        "time": 0.0,
        "position": [
          100,
          250
        ]
      },
      {
        "time": 0.5,
        "position": [
          250,
          250
        ]
      },
      {
        "time": 0.7,
        "position": [
          250,
          235
        ]
      },
      {
        "time": 0.9,
        "position": [
          265,
          250
        ]
      }
    ]
  },
  {
    "id": 102,
    "type": "car",
    "initial_position": [
      250,
      400
    ],
    "layer": "Movable",
    "trajectory_description": [
      {
        "time": 0.0,
        "position": [
          250,
          400
        ]
      },
      {
        "time": 0.5,
        "position": [
          250,
          250
        ]
      },
      {
        "time": 0.7,
        "position": [
          235,
          250
        ]
      },
      {
        "time": 0.9,
        "position": [
          250,
          265
        ]
      }
    ]
  },
  {
    "id": 103,
    "type": "bicycle",
    "initial_position": [
      300,
      230
    ],
    "layer": "Movable",
    "trajectory_description": [
      {
        "time": 0.0,
        "position": [
          300,
        230
      ]
    },
    {
      "time": 0.3,
      "position": [
        300,
        250
      ]
    },
    {
      "time": 0.6,
      "position": [
        250,
        250
      ]
    },
    {
      "time": 0.8,
      "position": [
        235,
        250
      ]
    }
  ]
},
{
  "id": 104,
  "type": "person",
  "initial_position": [
    300,
    215
  ],
  "layer": "Movable",
  "trajectory_description": [
    {
      "time": 0.0,
      "position": [
        300,
        215
      ]
    },
    {
      "time": 0.3,
      "position": [
        300,
            230
          ]
        },
        {
          "time": 0.6,
          "position": [
            300,
            250
          ]
        },
        {
          "time": 0.8,
          "position": [
            300,
            270
          ]
        }
      ]
    }
  ]
},
"tasks": [
  {
    "id": 1,
    "goal": {
      "type": "go",
      "target": "sidewalk_300.0_230.0",
      "key_item": null
    },
    "constraints": {
      "precedence": [],
      "evaluation": "arrive_at_target"
    },
    "help": {
      "baseline": "Head towards the sidewalk to wait for the cyclist.",
      "target": "Reach the designated waiting area.",
      "failure": "Do not enter the road until safe."
    },
    "failure_condition": {
      "type": "collision",
      "item": "car"
    },
    "instructions": {
      "text_en": "Go to the sidewalk at [300, 230] and wait.",
      "text_fr": "Allez sur le trottoir \u00e0 [300, 230] et attendez."
    },
  "trigger": {
    "object_id": 3,
    "function": "on_reach"
  }
},
{
  "id": 2,
  "goal": {
    "type": "interact",
    "target": "traffic_light_11",
    "key_item": null
  },
  "constraints": {
    "precedence": [
      1
    ],
    "evaluation": "light_green"
  },
  "help": {
    "baseline": "Observe the traffic light and wait for it to turn green.",
    "target": "Wait for the pedestrian signal to be green.",
    "failure": "Crossing on red is dangerous."
  },
  "failure_condition": {
    "type": "violation",
    "item": "traffic_light"
  },
  "instructions": {
    "text_en": "Wait for the traffic light at [240, 220] to turn green before proceeding.",
    "text_fr": "Attendez que le feu de circulation \u00e0 [240, 220] passe au vert."
  },
  "trigger": {
    "object_id": 11,
    "function": "on_light_green"
  }
},
{
  "id": 3,
  "goal": {
    "type": "go",
    "target": "road_250.0_250.0",
    "key_item": null
  },
  "constraints": {
    "precedence": [
      2
    ],
    "evaluation": "arrive_at_target"
  },
  "help": {
    "baseline": "Cross the intersection carefully.",
    "target": "Cross the road safely.",
    "failure": "Avoid vehicles."
  },
  "failure_condition": {
    "type": "collision",
    "item": "car"
  },
  "instructions": {
    "text_en": "Cross the intersection at [250, 250] when the light is green.",
    "text_fr": "Traversez l'intersection \u00e0 [250, 250] lorsque le feu est vert."
  },
  "trigger": {
    "object_id": 19,
    "function": "on_reach_road"
  }
},
{
  "id": 4,
  "goal": {
    "type": "go",
    "target": "sidewalk_270.0_250.0",
    "key_item": null
  },
  "constraints": {
    "precedence": [
      3
    ],
    "evaluation": "arrive_at_target"
  },
  "help": {
    "baseline": "Continue across the intersection to the other side.",
    "target": "Reach the other sidewalk.",
    "failure": "Stay on the pedestrian path."
  },
  "failure_condition": {
    "type": "collision",
    "item": "car"
  },
  "instructions": {
    "text_en": "Continue to the sidewalk at [270, 250].",
    "text_fr": "Continuez vers le trottoir \u00e0 [270, 250]."
```

```
      },
      "trigger": {
        "object_id": 6,
        "function": "on_reach"
      }
    }
  ]
}
```

10 Misc

10.1 Meeting Notes

10.1.1 01/08/2025

- **Clarify Image Usage:** Specify what kinds of images are being referred to in the input for the models (e.g., diagrams, real-world photos, layouts).
- **Add details on scenegraph:** Provide more information about the scenegraphs (e.g., format, usage, etc.) in the context of the models.
- **Avoid Overly High-Level Descriptions:** Some explanations are too abstract.
- **Improve Focus on Vocabulary:** Review and refine terminology. Ensure technical or domain-specific terms are clearly defined and used consistently.
- **Identify and Specify Missing Interaction Types:** Clearly outline which user/system interactions are missing or underexplored.
- **Sharpen Research Question (RQ):** Make the RQ more concrete.
- **Better Categorization of Literature:** Reorganize cited papers using clear categories such as research themes, methodologies, etc.

10.1.2 14/08/2025

- **RQ still too high level:** Provide more information about the scenegraphs (e.g., format, usage, etc.) in the context of the models.
- **Clarify the missing pieces in the SOTA:**
- **How to address:** diversity -¿ generative models, realism -¿ ontology, dynamic -¿ tasks
- **Start with WorkPlan:**
- **Remove relation between GUsT-3D and generation**

10.1.3 04/09/2025

- **Semantic Information Missing:** Find a way to match the scene to Kitti's prior.
- **Input unclear:** Scenegraph requires too much information. Need to simplify the input.
- **Use Layout:** Graphdreamer based score distillation doesn't work, use layout instead.

10.1.4 29/09/2025

- **Organize docs and notes:** Clarify the layout preferably with images.
- **Add missing citations:** Add missing citations for stuff like scene background generation in the workplan.
- **Detail:** Detail the missing sections more.
- **Git:** Add the code to gitlab.

10.1.5 08/10/2025

- **Validate with more prompts:** Validate the system with additional prompts - realistic, illogical, variations of the same prompts, prompts that recreate Florent's scenarios, decomposing prompts which are "difficult" for a pedestrian (jaywalking, parking lot entrance), etc.
- **Add in the doc:** the prompts discussed in the meeting, diffusion architecture, more details on validation, add examples for failure conditions, SOTA on prompt engineering for multimodal content.

10.1.6 21/10/2025

- **Check frame rate:** Was 10 Hz (not mentioned anywhere, but they used the same 10Hz cameras as the original KITTI dataset)
- **Log all runs:** Log all train/eval losses.
- **Cross check validation:** Need to test the model performance based on the input bboxes. To know whether the model was over/underfitting.
- **Check performance on unseen:** Test model performance based on unseen input bboxes.
- **Last week runs:** Send logs of last week's runs.
- **Adapt input:** Make Diffusion input more similar to the MLLM output or vice-versa OR fine-tune the Diffusion model.

10.1.7 04/11/2025

- **Better way to create finetune data for the VLM:**
- **Validation ideas:** Think of user testing, Think of metric for prompt to scene expectation, same prompt for multiple scenes, 100 scenes for user testing, think of more qualitative metrics for evaluation.
- **add explanations with examples to FID, KID and CKL:**
- **Think of the role of vlm and diffusion models**:
- **Check synthetic prompts** :
- **Fix scale of the maps to 20x20m** :

10.1.8 24/11/2025

- **Organise the results in the doc:**
- **Add KID and CKL tests in the doc:**
- **Compare the results of previous and new methods as well as gemini:**
- **Add all system and user prompts to the doc:**

10.2 Urban Scenario Generation

This section discusses pre-existing models used for generating scenarios in urban environments. These models can be broadly classified into procedural and deep generative approaches.

10.2.1 Procedural Methods

Procedural methods leverage algorithms and rules to generate urban scenarios, often resulting in highly customizable environments. They can be further divided into two categories:

Classic Procedural Generation use rules or constraints to generate scenes. The most popular procedural approach used for urban scenarios is Scenic [17], which allows users to define scenarios using a probabilistic programming language. Similarly, MetaUrban [63] is a popular urban scenario generation framework to create urban micromobility scenarios using the Metadrive [28] simulator.

LLM-based Procedural Generation can be used to enhance procedural generation with natural language understanding. For example, ScenicNL [14] and ChatScene [73] use LLMs to generate scenic code from prompts and then define the scenarios in Scenic. TTSG[47] is another framework that uses an to plan a traffic scenario using an LLM in JSON and then render it using CARLA [12]. CityX [76] is multi-agent framework that uses LLM to assemble assets using the procedural content generation (PCG) library as well as plan actions for the agents in the scenario.

Even though procedural approaches can be highly customizable and allow for precise control over the generated scenarios, they typically suffer from a lack of realism, as they rely on predefined rules that may not capture the full complexity of real-world environments. However, they can be useful in creating diverse scenarios that adhere to specific constraints or requirements. For example, a scenario with a crossroad with a specific number of lanes, a certain type of road surface, and a defined layout of buildings. However, no real crossroad would exist in the world that conforms to these specifications.

10.2.2 Deep-Generative Methods

Deep-Generative Approaches have recently become popular for various sorts of 3D generation. To create a complete scenario, some techniques use generative models such as GANs, VAEs or Diffusion models in combination with procedural techniques. These can be broadly classified into the following categories:

Procedural Environments with Deep-Generative Dynamics generate the static aspects of the environment using procedural techniques, while the dynamic aspects such as vehicles and pedestrians are generated using generative models. For example, Chatdyn [57] uses CARLA [12] to construct the traffic environment, then populate it with pedestrians and vehicles, each equipped with an LLM agent to generate a high-level scenario plan. This high-level plan is then executed in a low-level PedExecutor using Text2Motion [21] for pedestrians and VehExectutor using a physics-based, history-aware reinforcement learning controller to produce vehicle trajectories.

Procedural Dynamics with Deep-Generative Environments generate the dynamic aspects such as vehicles and pedestrians using procedural techniques. UrbanWorld [51] converts 2.5D urban layout data into a structured 3D city with separated assets, applying depth-aware, multi-view diffusion-based texture rendering and UV inpainting to achieve high-fidelity visuals. It also uses an urban MLLM for designing the world and provides dynamic elements which are planned using a random tree path planning algorithm. CityDreamer4D [64] modularly integrates autoregressive token-based generation, neural rendering (e.g., NeRF-style volumetrics), and procedural traffic modeling to synthesize large-scale, time-varying 3D cities.

Complete Deep-Generative Approaches recreate the static as well as the dynamic aspects using the learnt representation. Infinicube [37] constructs a large, dynamic 3D voxel world from input HD maps, vehicle positions, and text prompts; then, it generates photorealistic driving videos and reconstructs the scene into a manipulable 3D environment by fusing voxel- and video-based representations. UniScene [27] UniScene generates driving scenes in three modalities—semantic occupancy, multi-view video, and LiDAR—by first producing a controllable, temporally consistent occupancy sequence from BEV layouts. This occupancy then serves as unified geometric–semantic guidance to synthesize realistic videos and point clouds, ensuring cross-modal consistency and editability.

10.3 Scenegraph-Controlled Diffusion

Despite the advancements in urban scenario generation, very few methods have explored the use of scenegraphs to control the generation process. In recent years, the achievements in text-to-image generation has enabled the advancement of text-to-3D generation using Score Distillation Sampling (SDS) [42] which optimizes a 3D model by aligning 2D images rendered at arbitrary viewpoints with the distribution derived from a text-conditioned diffusion model. Subsequent works including ProlificDreamer [56] introduced Variational Score Distillation (VSD) which addresses the over-regularization and mode collapse issues of SDS by introducing a variational formulation that jointly optimizes the 3D representation and a learnable Gaussian noise distribution, enabling more faithful geometry and richer texture generation. GraphDreamer [18] uses the the SDS process with scene-graphs to enable more structured and controllable scene generation. However, it uses a simplified static scenegraph representation as shown in Visual genome [26] in which, nodes represent the objects in the scene, edges represent the relationship between these nodes and attriubtes represent the properties of the node. Example, Elderly (attribute) man (Node) wearing (edge) a hat(node). This limits the ability to generate dynamic scenarios where the relationships between objects change over time.

10.4 Scenegraph-guided Generation

Scenegraph-guided 3D generation is the process of creating 3D environments by leveraging scene graphs, which are structured representations of the objects and their relationships within a scene. A major advantage of scenegraph-guided generation is that it provides a clear relationship between the elements which allows for better generation than other techniques such as simple text prompts or bounding boxes. Scenethesis [32] uses a Vision-Language Model (VLM) to create a scenegraph with parent-child relationships and localizes objects using 3D bounding boxes. Work by Liu et al. [34] defines scenegraphs as graphs where instance nodes represent countable objects with semantic and positional features, a singleton road node encodes global scene structure, and edges capture both physical proximity among instances and connectivity to the road. X-Scene [69] is another work that uses LLMs (Large-Language Models) to create a scenegraph with nodes (objects) and edges (relationships) to facilitate the generation process. Graphdreamer [18] employs scenegraphs structured around the Visual Genome [26] format, where nodes represent objects with associated attributes, and edges encode the relationships between these objects to guide the generation process. Despite the advancements in scenegraph-guided generation, all of these works focus on static scenes and do not consider dynamic scenarios where the state of the objects changes over time.

10.5 Grounding: Input for Scene Generation

Generative models have already been applied to urban scenario generation, where models synthesize plausible urban environments, pedestrian layouts, or vehicle positions from various types of input. An input in the context of a generative model is any data—such as a prompt, image, or scenario graph—provided to influence the output. Grounding, on the other hand, is the process of linking elements of that input to specific, coherent representations in the generated scenario, such as ensuring the "footpath" appears visually plausible, is correctly positioned on the "road", and respects spatial relationships or physical constraints. Grounding ensures the generated scenario isn't just randomly composed but meaningfully aligned with the intended semantics of the input. The common ways to do grounding are,

10.5.1 Rules and Constraints

Rules can be used to give a prescriptive logic that defines how to generate or modify a scenario. Rule-based systems are necessarily procedural, meaning they follow a set of predefined steps or algorithms to create scenarios. Table 5 lists some of the recent rule-based models for scenario generation. Although these systems allow users to define rules for generating scenarios—such as the placement of buildings, roads, and other elements—they often require an intermediate representation which captures the real-world environments. Creating such representations can be challenging, as it requires a deep understanding of the underlying rules and relationships between different elements. Additionally, rule-based systems can be limited in their ability to generate diverse and realistic scenarios, as they rely on predefined rules that may not capture the full complexity of real-world environments.traffic

Model	**Technique**	**Output**
CityEngine [40]	Procedural Modeling with Rules	3D scene
Infinigen [44]	Procedural Modeling with Rules	3D scene
MetaUrban [63]	Description Scripts for Scene Layout	3D Scenario

Table 5: Rule-based Models for Urban Scenario Generation

Constraints define conditions that must be satisfied to get a valid scenario. Scenic [17] is a probabilistic programming language that allows users to define constraints for generating scenarios. It uses a declarative approach to specify the properties of the scenario, such as the layout, objects, and their relationships. These can then be rendered using a frontend such as CARLA [12]. However, it suffers from the same limitations of rule-based systems, as it can only generate scenarios that fit within the defined constraints, potentially missing out on the richness and variability of real-world environments.

10.6 Diffusion-based 4D Generation

Diffusion models have emerged as a powerful approach for generating high-quality 3D content by iteratively refining a random noise input into a coherent output. DreamFusion [42] initially introduced text-to-3D synthesis by optimizing a neural radiance field so that rendered views, when noised and denoised by a pretrained text-conditioned diffusion model, produce identical gradients via Score Distillation Sampling (SDS). This framework was extended to dynamic scenes in MaV3D, which employs video-based SDS to animate a time-conditioned radiance field into a 4D scene [52]. However, MaV3D's reliance on a single diffusion prior leads to trade-offs between appearance fidelity, 3D consistency, and motion realism. 4D-fy [3] addresses this by hybridizing three SDS signals—3D-aware image SDS for geometry, standard image SDS for texture, and video SDS for motion—alternating updates to preserve all three qualities. Animate124 [77] further refines single-image animation into 4D using a coarse-to-fine 4D grid backbone optimized first with 2D and 3D image priors, then with video diffusion, and finally with a personalized ControlNet fine-tuning stage to prevent semantic drift. More recently, Trans4D [71] leverages a multimodal large language model to perform physics-aware 4D scene planning—generating object trajectories, rotations, and transition times—and introduces a Transition Network that predicts point-wise appearance or disappearance probabilities to realize complex geometry-aware transitions such as a missile transforming into an explosion cloud. Despite their impressive generative capabilities, these diffusion-based 4D synthesis methods still lack the explicit, structured control afforded by scene graphs, making it difficult to enforce complex object relationships or constraints at generation time.

10.7 Datasets

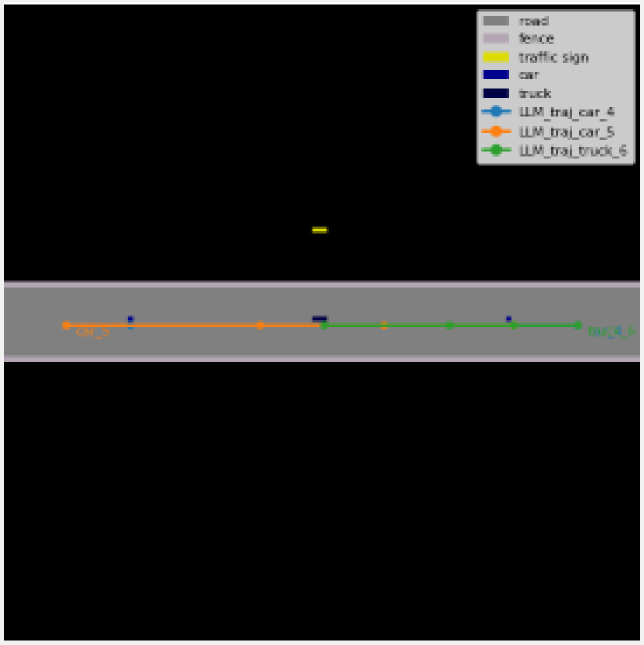
In this section, we review the datasets that are used to train generative models for road crossing scenarios. Table 6 lists some of the datasets commonly used for training and evaluating generative models in urban environments.

Table 6: Overview of selected datasets for foundation model-based scenario generation and analysis.

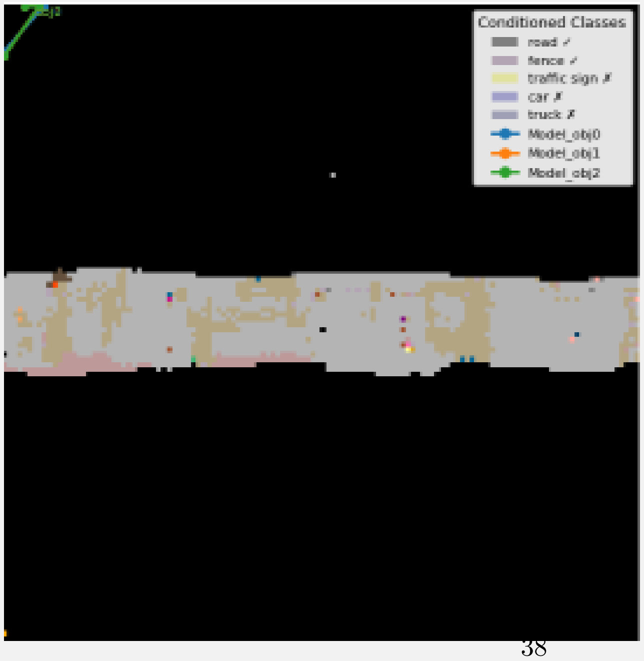
Dataset	**Year**	**View**	**Source**
SIND [67]	2022	BEV	Real
Waymo Open [53]	2020	FPV	Real
Argoverse [8][58]	2023	BEV/FPV	Real
nuScenes [7]	2022	FPV	Real
KITTI [20]	2012	FPV	Real
Cityscapes [9]	2016	FPV	..
HoliCity [79]	2020	FPV	..
OmniCity [29]	2023	FPV	..
GoogleEarth [64]	2024	BEV	Real
OSM [64]	2024	BEV	Real
CarlaSC [59]	2022	BEV/FPV	Synthetic
CityTopia [65]	2025	BEV/FPV	..

10.8 Multinomial Diffusion for Semantic Map Generation

Input example:
LLM predicted bounding boxes for the prompt: *Road with multiple lanes, guardrails with 2 cars and a truck.*



Output example:
The output is a top-down semantic map representing the static elements of the 3D scene to be generated.



The next step is to generate a static 3D scene top-down semantic map from the bouding boxes predicted by the MLLM.

1. Preparation of Conditional Inputs Given the predicted bounding box layout generated by the MLLM, we build a structured *conditional representation* that serves as the spatial input to the diffusion model. Each bounding box entry in the layout corresponds to a semantic instance (e.g., *road*, *building*, *car*), which is mapped to its corresponding KITTI-360 class ID using a predefined class dictionary.

We then convert the entire layout into a *spatial mask* (conditioning tensor). This process rasterizes all object instances into a dense 2D class map of fixed spatial resolution ($H \times W$), where each pixel encodes the semantic label of the corresponding region. The result is a tensor

$$\mathbf{M} \in \mathbb{R}^{1\times H\times W},$$

representing the input condition for the diffusion process.

If dynamic agents and their motion trajectories are available from the MLLM output, we additionally construct a *trajectory tensor*

$$\mathbf{T} \in \mathbb{R}^{1\times N_{\mathrm{dyn}}\times T\times 2},$$

where N_{dyn} denotes the number of moving entities and T the number of timesteps. Each trajectory consists of normalized coordinates in map space, which are projected into pixel coordinates consistent with the spatial mask dimensions. To ensure temporal alignment, all trajectories are padded or truncated to a uniform sequence length.

These conditional inputs—the **semantic spatial mask** and, the **dynamic trajectory tensor**—form the conditioning context for the diffusion model's generative process.

2. Multinomial Diffusion for Layout Synthesis. Based on the discrete diffusion framework introduced in [24], we design a multinomial diffusion model for semantic layout synthesis. The model uses a U-Net that iteratively denoises top-down semantic maps from the KITTI-360 dataset, conditioned on both static spatial masks and dynamic trajectories. This formulation enables the joint modeling of scene semantics and motion priors.

Forward (noising) process. Let $\mathbf{x}_0 \in \{1,\ldots,K\}^{H\times W}$ denote the clean semantic layout, where each pixel corresponds to one of K semantic categories. The forward diffusion process gradually corrupts $\mathbf{x}_0$ through a sequence of categorical random variables $\{\mathbf{x}_t\}_{t=1}^{T}$ sampled via a fixed multinomial transition kernel:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathrm{Cat}\left(\mathbf{x}_t;\, (1-\beta_t)\,\mathbf{1}_{\mathbf{x}_{t-1}} + \beta_t\,\frac{\mathbf{1}}{K}\right), \tag{4}$$

where $\beta_t \in [0,1]$ denotes the noise schedule and $\mathbf{1}_{\mathbf{x}_{t-1}}$ is the one-hot encoding of $\mathbf{x}_{t-1}$. Repeated application of this transition progressively replaces class labels with uniform noise, such that at step T the distribution approaches:

$$q(\mathbf{x}_T \mid \mathbf{x}_0) = \mathrm{Cat}\left(\mathbf{x}_T;\, \bar{\alpha}_T\,\mathbf{1}_{\mathbf{x}_0} + (1-\bar{\alpha}_T)\,\frac{\mathbf{1}}{K}\right), \tag{5}$$

where $\bar{\alpha}_T = \prod_{s=1}^{T}(1-\beta_s)$.

Reverse (denoising) process. The reverse process reconstructs $\mathbf{x}_0$ from $\mathbf{x}_T$ by learning the posterior distribution

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) = \mathrm{Cat}(\mathbf{x}_{t-1}; \mathbf{p}_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}))\,, \tag{6}$$

where $\mathbf{c}$ represents all conditioning inputs (see below). At each timestep t, the U-Net predicts categorical logits $\mathbf{h}_t$ conditioned on the noisy input, timestep embedding, spatial mask, and motion features:

$$\mathbf{h}_t = f_\theta(\mathbf{x}_t,\, t,\, \mathbf{M},\, \mathbf{T}), \tag{7}$$

$$\mathbf{p}_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) = \mathrm{softmax}(\mathbf{h}_t). \tag{8}$$

Training objective and bits-per-dimension. The model is trained by minimizing the evidence lower bound (ELBO), which corresponds to the expected cross-entropy between the predicted denoised distribution and the true posterior:

$$\mathcal{L}_{\mathrm{ELBO}} = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}_t, \mathbf{x}_0)}[\mathrm{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}))]\,. \tag{9}$$

This objective is implemented as a per-pixel cross-entropy loss. To measure log-likelihood quality, we report the ELBO in *bits-per-dimension (bpd)*:

$$\mathrm{bpd} = \frac{\mathcal{L}_{\mathrm{ELBO}}}{H \times W \times \log 2}, \tag{10}$$

where lower bpd values indicate higher likelihood and better generative quality.

Sampling. During inference, we begin from a uniform categorical noise map $\mathbf{x}_T$ and iteratively sample

$$\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}), \quad t = T, \ldots, 1,$$

to produce a semantic layout consistent with the MLLM-generated layout.

Conditioning inputs to the U-Net. At each denoising step, the U-Net receives:

- **Noisy semantic map:** $\mathbf{x}_t \in \{0, \ldots, K-1\}^{B \times H \times W}$, embedded into a continuous feature space of dimension d;
- **Temporal embedding:** a sinusoidal timestep embedding $\tau(t) \in \mathbb{R}^d$, injected into each ResNet block;
- **Static spatial conditioning:** a semantic mask $\mathbf{M} \in \{0, \ldots, K-1\}^{B \times H_s \times W_s}$ projected via

  $$\Phi_{\mathrm{spatial}} : \mathbb{R}^{B \times K \times H_s \times W_s} \to \mathbb{R}^{B \times d \times H \times W};$$

- **Dynamic conditioning:** Trajectories $\mathbf{T}_{\mathrm{in}} \in \mathbb{R}^{B \times N_{\mathrm{dyn}} \times T_{\mathrm{in}} \times 2}$ encoded into:

  $$\mathbf{E}_{\mathrm{obj}} \in \mathbb{R}^{B \times N_{\mathrm{dyn}} \times d}, \qquad \mathbf{e}_{\mathrm{motion}} \in \mathbb{R}^{B \times d},$$

  where $\mathbf{e}_{\mathrm{motion}}$ modulates U-Net blocks and $\mathbf{E}_{\mathrm{obj}}$ is used by a shared MLP for auxiliary trajectory prediction.

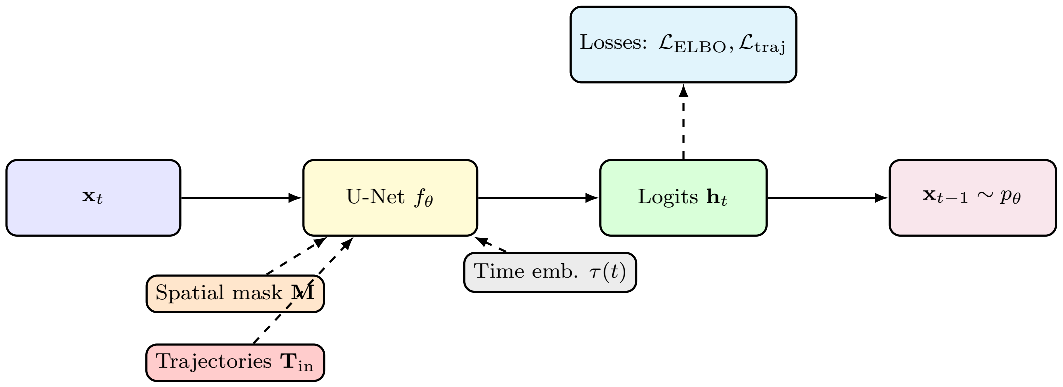

Losses: $\mathcal{L}_{\text{ELBO}}, \mathcal{L}_{\text{traj}}$
$\mathbf{x}_t$
U-Net f_θ
Logits $\mathbf{h}_t$
$\mathbf{x}_{t-1} \sim p_\theta$
Time emb. $\tau(t)$
Spatial mask $\mathbf{M}$
Trajectories $\mathbf{T}_{\text{in}}$

Figure 2: Overview of the multinomial diffusion U-Net for layout synthesis. The model denoises categorical maps $\mathbf{x}_t$ conditioned on spatial and dynamic inputs, optimizing ELBO and auxiliary trajectory losses.

Network outputs. The U-Net produces:

- **Per-pixel logits:** logits $\in \mathbb{R}^{B\times K\times H\times W}$ representing categorical distributions for denoising;
- **Auxiliary trajectory predictions:** $\widehat{\mathbf{T}}_{\text{refined}} \in \mathbb{R}^{B\times N_{\text{dyn}}\times M\times 2}$, predicted via a shared MLP.

3. Output The final output is a top-down semantic map representing the static elements of the 3D scene to be generated. This map can then be used with score distillation sampling (as shown in Urban Architect [36]) to create the full 3D scene.

11 Experiments

This section presents a log of all experiments conducted to train the *multinomial diffusion model*. Detailed parameters of the experiments are available here for reference. Figure 9 shows the whisker plots of the eval losses across experiments. Table 7 visualises the validation results for each experiment, whereas Table 8 visualises the LLM conditioning results for each experiment. Figure 6 shows the legend of the generated maps.

11.1 Experiment 1: `notebook_test`

Goal: Train the model based on the degraded semantic maps extracted from the KITTI-360 layouts.
Parameters:

- Latent Dimension: 32
- Timesteps: 100
- Epochs: 100
- Number of maps: 100

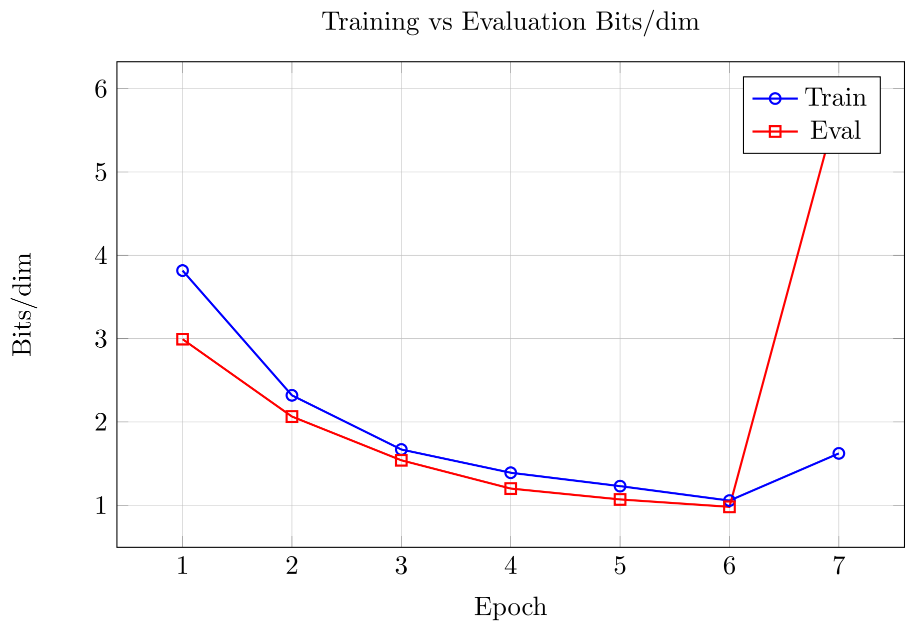


Figure 3: Loss curves for Experiment 1

Results:

- The losses oscillated significantly due to a high learning rate as shown in Figure 3.
- Training failed.

11.2 Experiment 2: `notebook_test_stable`

Goal: Based on the previous results. The goal of this run was to stabilise the training as well as check the training and inference for the trajectory generation code.

Parameters:

- Class weights: added based for trajectories based on the frequency of dynamic classes.
- Latent Dimension: 64
- Timesteps: 1000
- Number of maps: 15000
- File stride: 5
- Epochs: 50

Changes Made:

- Decreased learning rate by a factor of 10 for more stable convergence.
- Added input augmentations (rotations and flips) to improve dataset variability.
- Increased diffusion embedding dimension from 32 to 64 to mitigate class swapping.
- Increased diffusion timesteps from 100 to 1000 (improving quality at the cost of training/inference time).
- Implemented the following enhancements:
  - **Trajectory logic:** Introduced to handle dynamic elements.
  - **Class weights:** Applied to dynamic classes to handle class imbalance.
  - **File stride:** Sampling every nth file to increase variability.
  - **Frame skip:** Similar to stride, but applied to trajectory sequences.
  - **Sequence length:** Defined the number of frames per trajectory.

Results:

- The training became significantly more stable compared to the previous run. The lower validation loss compared to the training loss could be due to a smaller number of epochs. Additionally, the validation loss is measured after each training epoch. Figure 4 shows the loss curves for each epoch.
- The static generation generates artefacts (such as the dynamic components) as shown in tables 7 and 8.
- The trajectory tensors had much lower values and didn't conform to the spatial mask.

11.3 Experiment 3: `run_20251023_020445`

Goal: Conduct a complete static layout generation run after incorporating insights from prior experiments and discussions.

Changes Made:

- Ignored class weights and trajectory-related computations to focus solely on static layout generation.
- Replaced bounding-box extraction with a grid-based approach to convert exact shape masks into bounding boxes. This approach aligns better with the blocky, large-object nature of LLM-generated outputs.

Parameters:

- Class weights: added based for trajectories based on the frequency of dynamic classes.
- Latent Dimension: 64
- Timesteps: 1000

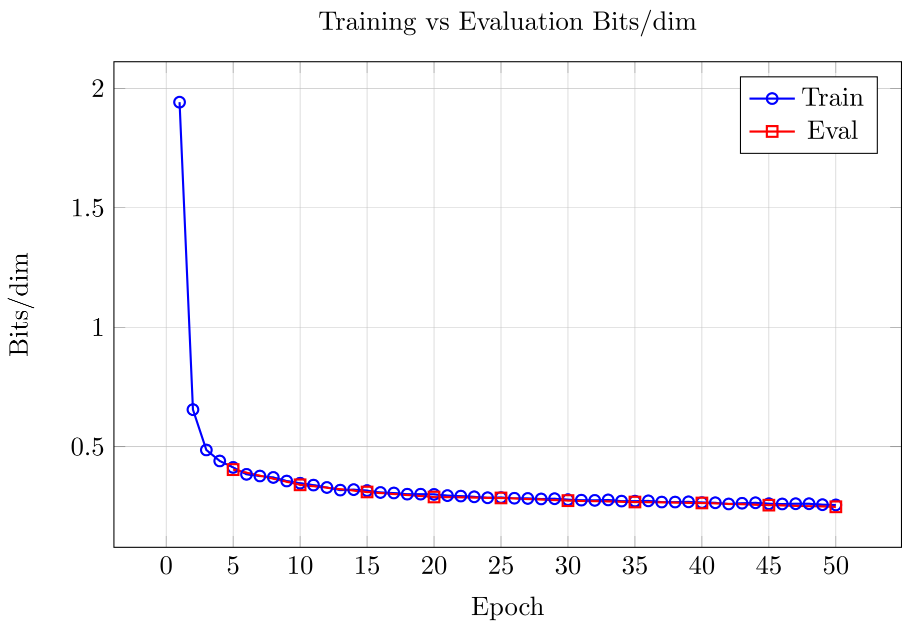


Figure 4: Loss curves for Experiment 2

- Number of maps: 7500
- File stride: 10
- Epochs: 25

Results:

1. Final training and validation loss of 0.25 and 0.24. The training and validation loss curves are shown in Figure 7.
2. The model does regenerate the main structures in the images, but it also introduces dynamic objects, such as cars, into the static map. Figure 6 shows the legend of the generated maps. In future, A better approach could be to mask out these dynamic objects completely from the training data for static inference.

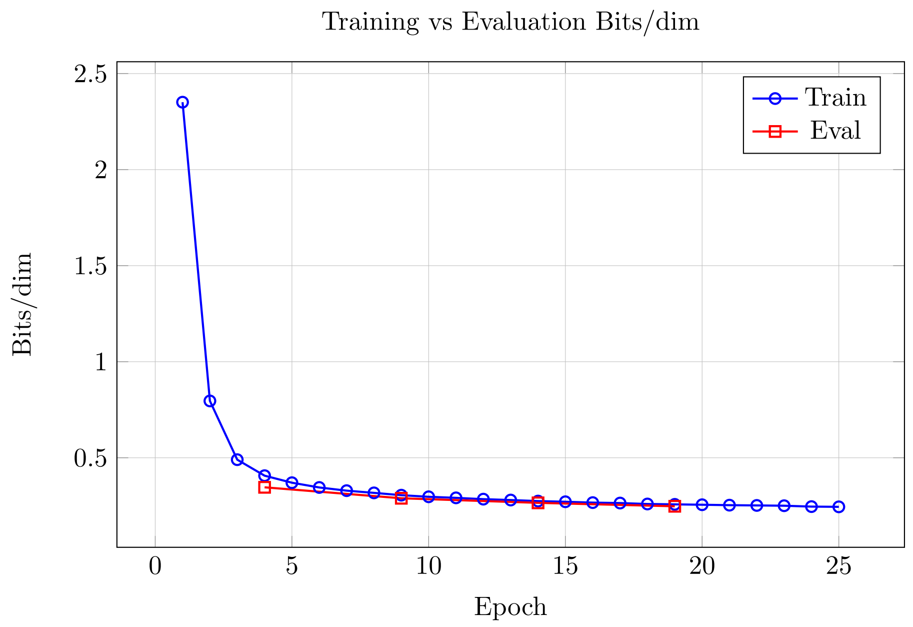


Figure 5: Loss curves for Experiment 3

11.4 Experiment 4: `run_20251025_031240`

Goal: Conduct a complete static layout generation run after incorporating insights from prior experiments and discussions.

Parameters:

- Class weights: added based for trajectories based on the frequency of dynamic classes.
- Latent Dimension: 64
- Timesteps: 1000
- Number of maps: 5000
- File stride: 14
- Epochs: 50

Changes Made:

- Ignored class weights and trajectory-related computations to focus solely on static layout generation.
- Replaced bounding-box extraction with a grid-based approach to convert exact shape masks into bounding boxes. This approach aligns better with the blocky, large-object nature of LLM-generated outputs.

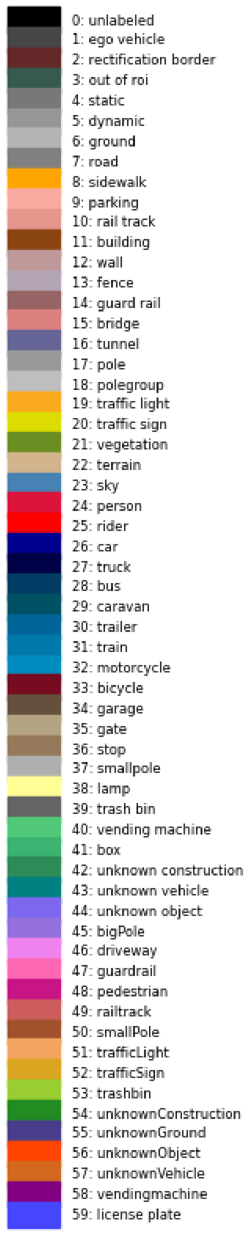


Figure 6: Legend for generated maps.

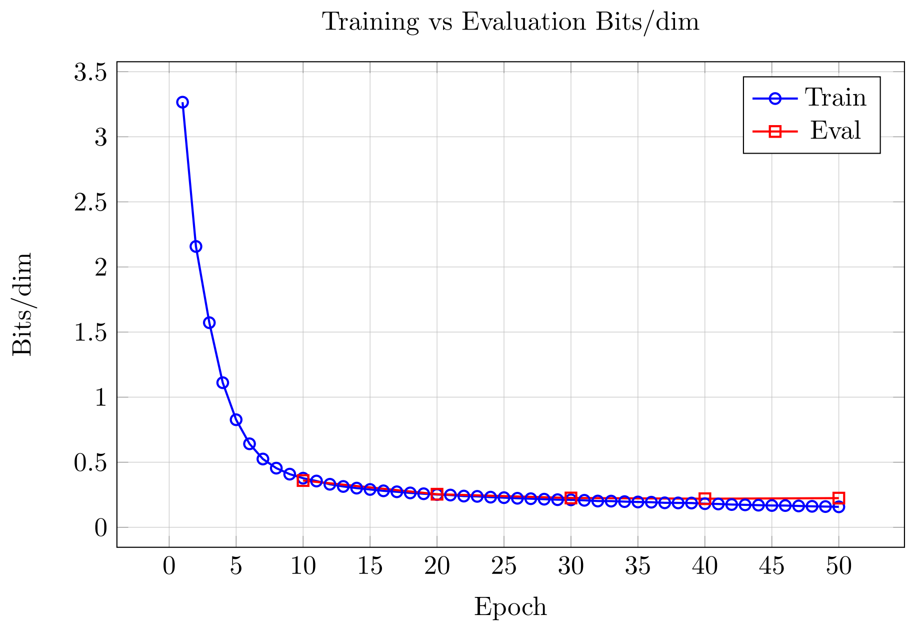


Figure 7: Loss curves for Experiment 4

Results:

1. Final training and validation loss of 0.157 and 0.224. An additional test loss of 0.229803 was calculated on the test dataset after the training The training and validation loss curves are shown in Figure 7. The much-lower training loss in this run suggests that the model overfitted to the training data compared to previous which is also reflected by the increase in validation loss in last 10 epochs.
2. The model does not introduce dynamic objects, and also infills static objects for certain maps. However, some artefacts are still present in the generated maps as shown in tables 7 and 8.

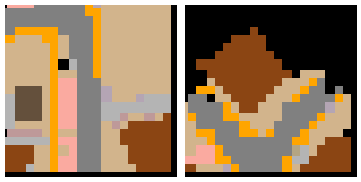
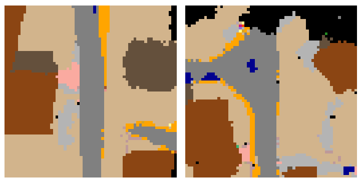
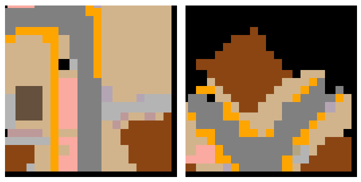
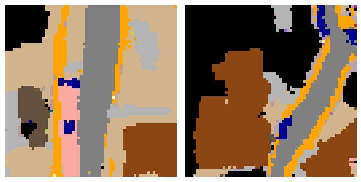
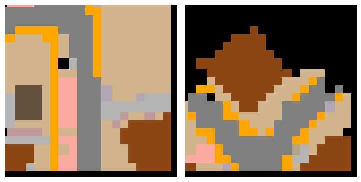
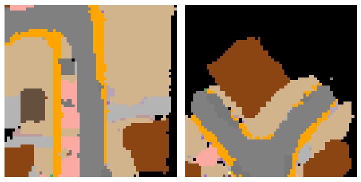
11.5 Model Architecture

Figure 8 illustrates the proposed model architecture for generating dynamic urban scenarios. The model takes a scenario prompt as input, which is processed by a large language model (LLM) to extract two key components: layout and tasks. Each component is then handled by specialized modules to generate the final 3D dynamic scene.

References

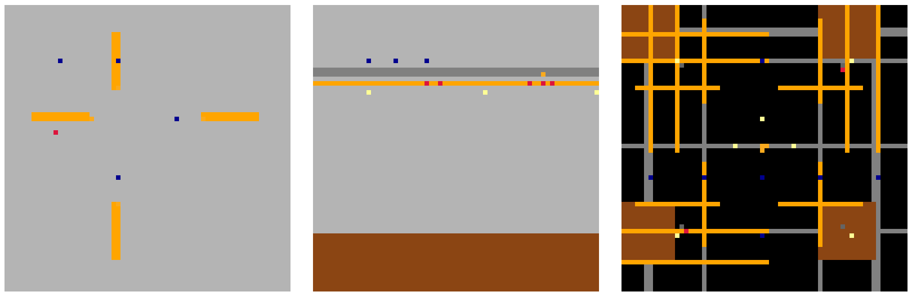
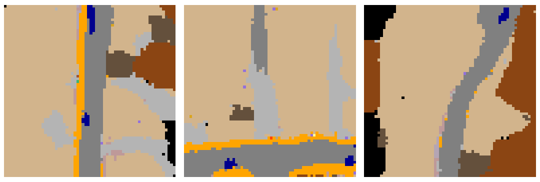
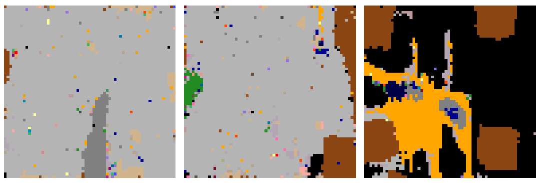
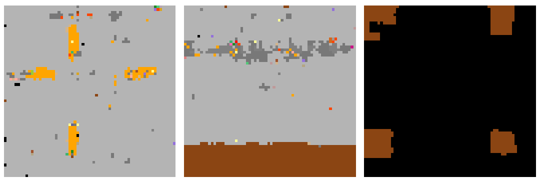
[1] E. Alonso, A. Jelley, V. Micheli, A. Kanervisto, A. J. Storkey, T. Pearce, and F. Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

Table 7: Comparison of Validation Conditioning, Ground Truth, and Inferred Images

Exp. No.	**Conditioning**	**Inferred Image**
1	Training Unstable	
2		
3		
4		

Top: Ground truth semantic maps used as conditioning input. Below: The conditionings and inferred outputs from the diffusion model across four experiments.

Table 8: Comparison of LLM Conditioning and Inferred Images

LLM Conditioning	
	
Exp. No.	**Inferred Image**
1	Training Unstable
2	
3	
4	

Top: LLM-generated semantic maps used as conditioning input. Below: Inferred outputs from the diffusion model across four experiments.

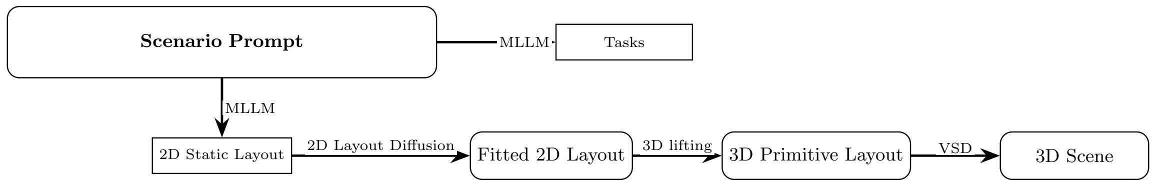


Figure 8: Model Architecture

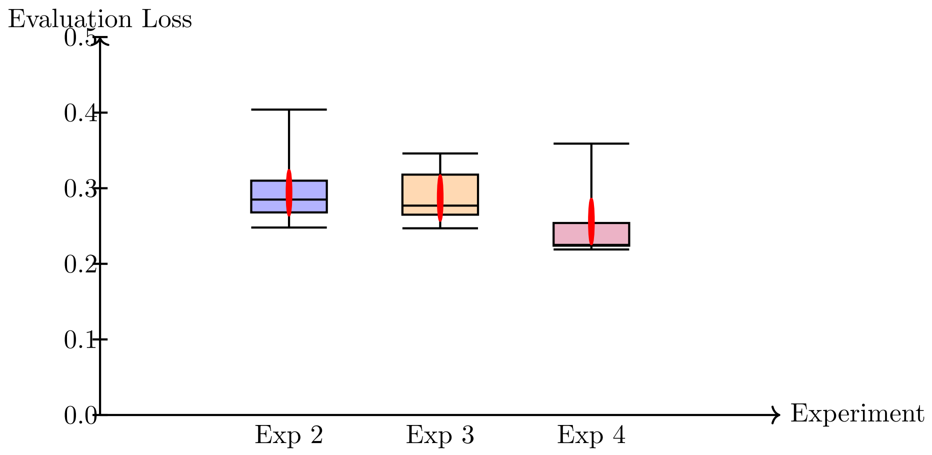


Figure 9: Whisker plot of eval losses for Experiments 2, 3 and 4

[2] S. Bahmani, J. J. Park, D. Paschalidou, X. Yan, G. Wetzstein, L. Guibas, and A. Tagliasacchi. Cc3d: Layout-conditioned generation of compositional 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7171–7181, 2023.

[3] S. Bahmani, I. Skorokhodov, V. Rong, G. Wetzstein, L. Guibas, P. Wonka, S. Tulyakov, J. J. Park, A. Tagliasacchi, and D. B. Lindell. 4d-fy: Text-to-4d generation using hybrid score distillation sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7996–8006, 2024.

[4] M. A. Bautista, P. Guo, S. Abnar, W. Talbott, A. Toshev, Z. Chen, L. Dinh, S. Zhai, H. Goh, D. Ulbricht, et al. Gaudi: A neural architect for immersive 3d scene generation. *Advances in Neural Information Processing Systems*, 35:25102–25116, 2022.

[5] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[6] J. Bruce, M. D. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[7] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In

Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 11621–11631, 2020.

[8] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8748–8757, 2019.

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[10] B. Deng, R. Tucker, Z. Li, L. Guibas, N. Snavely, and G. Wetzstein. Streetscapes: Large-scale consistent street view generation using autoregressive video diffusion. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[11] J. Deng, W. Chai, J. Huang, Z. Zhao, Q. Huang, M. Gao, J. Guo, S. Hao, W. Hu, J.-N. Hwang, et al. Citycraft: A real crafter for 3d city generation. *arXiv preprint arXiv:2406.04983*, 2024.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[13] P. Dourish. *Where the action is: the foundations of embodied interaction.* MIT press, 2001.

[14] K. Elmaaroufi, D. Shanker, A. Cismaru, M. Vazquez-Chanlatte, A. Sangiovanni-Vincentelli, M. Zaharia, and S. A. Seshia. Scenicnl: generating probabilistic scenario programs from natural language. *arXiv preprint arXiv:2405.03709*, 2024.

[15] W. Feng, W. Zhu, T.-j. Fu, V. Jampani, A. Akula, X. He, S. Basu, X. E. Wang, and W. Y. Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36:18225–18250, 2023.

[16] Y. Feng, J. Jiang, J. Ren, W. Li, R. Li, and X. Fan. Text-guided editable 3d city scene generation. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

[17] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia. Scenic: a language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation*, pages 63–78, 2019.

[18] G. Gao, W. Liu, A. Chen, A. Geiger, and B. Schölkopf. Graphdreamer: Compositional 3d scene synthesis from scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21295–21304, 2024.

[19] R. Gao, K. Chen, Z. Li, L. Hong, Z. Li, and Q. Xu. Magicdrive3d: Controllable 3d generation for any-view rendering in street scenes. *arXiv preprint arXiv:2405.14475*, 2024.

[20] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[21] C. Guo, Y. Mu, M. G. Javed, S. Wang, and L. Cheng. Momask: Generative masked modeling of 3d human motions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1900–1910, 2024.

[22] M. Hartmann, M. Viehweger, W. Desmet, M. Stolz, and D. Watzenig. "pedestrian in the loop": An approach using virtual reality. In *2017 XXVI International Conference on Information, Communication and Automation Technologies (ICAT)*, pages 1–8. IEEE, 2017.

[23] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[24] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

[25] Y. Jin, R. Yang, Z. Yi, X. Shen, H. Peng, X. Liu, J. Qin, J. Li, J. Xie, P. Gao, et al. Surrealdriver: Designing llm-powered generative driver agent framework based on human drivers' driving-thinking data. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 966–971. IEEE, 2024.

[26] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[27] B. Li, J. Guo, H. Liu, Y. Zou, Y. Ding, X. Chen, H. Zhu, F. Tan, C. Zhang, T. Wang, et al. Uniscene: Unified occupancy-centric driving scene generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 11971–11981, 2025.

[28] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 45(3):3461–3475, 2022.

[29] W. Li, Y. Lai, L. Xu, Y. Xiangli, J. Yu, C. He, G.-S. Xia, and D. Lin. Omnicity: Omnipotent city understanding with multi-level and multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17397–17407, 2023.

[30] Z. Li, H.-X. Yu, W. Liu, Y. Yang, C. Herrmann, G. Wetzstein, and J. Wu. Wonderplay: Dynamic 3d scene generation from a single image and actions. *arXiv preprint arXiv:2505.18151*, 2025.

[31] Y. Liao, J. Xie, and A. Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *Pattern Analysis and Machine Intelligence (PAMI)*, 2022.

[32] L. Ling, C.-H. Lin, T.-Y. Lin, Y. Ding, Y. Zeng, Y. Sheng, Y. Ge, M.-Y. Liu, A. Bera, and Z. Li. Scenethesis: A language and vision agentic framework for 3d scene generation. *arXiv preprint arXiv:2505.02836*, 2025.

[33] L. Liu, S. Chen, S. Jia, J. Shi, Z. Jiang, C. Jin, W. Zongkai, J.-N. Hwang, and L. Li. Graph canvas for controllable 3d scene generation. *arXiv preprint arXiv:2412.00091*, 2024.

[34] Y. Liu, X. Li, Y. Zhang, L. Qi, X. Li, W. Wang, C. Li, X. Li, and M.-H. Yang. Controllable 3d outdoor scene generation via scene graphs. *arXiv preprint arXiv:2503.07152*, 2025.

[35] Y.-T. Liu, Y.-C. Guo, V. Voleti, R. Shao, C.-H. Chen, G. Luo, Z. Zou, C. Wang, C. Laforte, Y.-P. Cao, et al. Threestudio: A modular framework for diffusion-guided 3d generation. *cg. cs. tsinghua. edu. cn*, 2023.

[36] F. Lu, K.-Y. Lin, Y. Xu, H. Li, G. Chen, and C. Jiang. Urban architect: Steerable 3d urban scene generation with layout prior. *arXiv preprint arXiv:2404.06780*, 2024.

[37] Y. Lu, X. Ren, J. Yang, T. Shen, Z. Wu, J. Gao, Y. Wang, S. Chen, M. Chen, S. Fidler, et al. Infinicube: Unbounded and controllable dynamic 3d driving scene generation with world-guided video models. *arXiv preprint arXiv:2412.03934*, 2024.

[38] K. Mukoya, E. Weng, R. Choudhury, and K. Kitani. Jaywalkervr: A vr system for collecting safety-critical pedestrian-vehicle interactions. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9600–9607. IEEE, 2024.

[39] B. Murugadoss, C. Poelitz, I. Drosos, V. Le, N. McKenna, C. S. Negreanu, C. Parnin, and A. Sarkar. Evaluating the evaluator: Measuring llms' adherence to task evaluation instructions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 19589–19597, 2025.

[40] Y. I. Parish and P. Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308, 2001.

[41] M. Parmar, N. Patel, N. Varshney, M. Nakamura, M. Luo, S. Mashetty, A. Mitra, and C. Baral. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. *arXiv preprint arXiv:2404.15522*, 2024.

[42] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[43] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[44] A. Raistrick, L. Lipson, Z. Ma, L. Mei, M. Wang, Y. Zuo, K. Kayan, H. Wen, B. Han, Y. Wang, et al. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12630–12641, 2023.

[45] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[46] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[47] B.-K. Ruan, H.-T. Tsui, Y.-H. Li, and H.-H. Shuai. Traffic scene generation from natural language description for autonomous vehicles with large language model. *arXiv preprint arXiv:2409.09575*, 2024.

[48] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[49] A. Savkin, R. Ellouze, N. Navab, and F. Tombari. Unsupervised traffic scene generation with synthetic 3d scene graphs. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1229–1235. IEEE, 2021.

[50] S. Schneider and K. Bengler. Virtually the same? analysing pedestrian behaviour by means of virtual reality. *Transportation research part F: traffic psychology and behaviour*, 68:231–256, 2020.

[51] Y. Shang, Y. Lin, Y. Zheng, H. Fan, J. Ding, J. Feng, J. Chen, L. Tian, and Y. Li. Urbanworld: An urban world model for 3d city generation. *arXiv preprint arXiv:2407.11965*, 2024.

[52] U. Singer, S. Sheynin, A. Polyak, O. Ashual, I. Makarov, F. Kokkinos, N. Goyal, A. Vedaldi, D. Parikh, J. Johnson, et al. Text-to-4d dynamic scene generation. *arXiv preprint arXiv:2301.11280*, 2023.

[53] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[54] T. T. M. Tran, C. Parker, and M. Tomitsch. A review of virtual reality studies on autonomous vehicle–pedestrian interaction. *IEEE Transactions on Human-Machine Systems*, 51(6):641–652, 2021.

[55] D. Valevski, Y. Leviathan, M. Arar, and S. Fruchter. Diffusion models are real-time game engines. *arXiv preprint arXiv:2408.14837*, 2024.

[56] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in neural information processing systems*, 36:8406–8441, 2023.

[57] Y. Wei, J. Wang, Y. Du, D. Wang, L. Pan, C. Xu, Y. Feng, B. Dai, and S. Chen. Chatdyn: Language-driven multi-actor dynamics generation in street scenes. *arXiv preprint arXiv:2412.08685*, 2024.

[58] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*, 2023.

[59] J. Wilson, J. Song, Y. Fu, A. Zhang, A. Capodieci, P. Jayakumar, K. Barton, and M. Ghaffari. Motionsc: Data set and network for real-time semantic mapping in dynamic environments. *IEEE Robotics and Automation Letters*, 7(3):8439–8446, 2022.

[60] H. Wu, D. H. Ashmead, H. Adams, and B. Bodenheimer. Using virtual reality to assess the street crossing behavior of pedestrians with simulated macular degeneration at a roundabout. *Frontiers in ICT*, 5:27, 2018.

[61] H.-Y. Wu, F. Robert, T. Fafet, B. Graulier, B. Passin-Cauneau, L. Sassatelli, and M. Winckler. Designing guided user tasks in vr embodied experiences. *Proceedings of the ACM on Human-Computer Interaction*, 6(EICS):1–24, 2022.

[62] H.-Y. Wu, F. A. S. Robert, F. F. Gallo, K. Pirkovets, C. Quere, J. Delachambre, S. Ramanoël, A. Gros, M. Winckler, L. Sassatelli, et al. Exploring, walking, and interacting in virtual reality with simulated low vision: a living contextual dataset (2023), 2023.

[63] W. Wu, H. He, Y. Wang, C. Duan, J. He, Z. Liu, Q. Li, and B. Zhou. Metaurban: A simulation platform for embodied ai in urban spaces. *arXiv e-prints*, pages arXiv–2407, 2024.

[64] H. Xie, Z. Chen, F. Hong, and Z. Liu. Citydreamer: Compositional generative model of unbounded 3d cities. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9666–9675, 2024.

[65] H. Xie, Z. Chen, F. Hong, and Z. Liu. Citydreamer4d: Compositional generative model of unbounded 4d cities. *arXiv e-prints*, pages arXiv–2501, 2025.

[66] F. Xu, Q. Lin, J. Han, T. Zhao, J. Liu, and E. Cambria. Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[67] Y. Xu, W. Shao, J. Li, K. Yang, W. Wang, H. Huang, C. Lv, and H. Wang. Sind: A drone dataset at signalized intersection in china. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2471–2478. IEEE, 2022.

[68] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[69] Y. Yang, A. Liang, J. Mei, Y. Ma, Y. Liu, and G. H. Lee. X-scene: Large-scale driving scene generation with high fidelity and flexible controllability. *arXiv preprint arXiv:2506.13558*, 2025.

[70] Y. Yang, F. Yin, J. Fan, X. Chen, W. Li, and G. Yu. Scene123: One prompt to 3d scene generation via video-assisted and consistency-enhanced mae. *arXiv preprint arXiv:2408.05477*, 2024.

[71] B. Zeng, L. Yang, S. Li, J. Liu, Z. Zhang, J. Tian, K. Zhu, Y. Guo, F.-Y. Wang, M. Xu, et al. Trans4d: Realistic geometry-aware transition for compositional text-to-4d synthesis. *arXiv preprint arXiv:2410.07155*, 2024.

[72] J. Zhang, X. Li, Z. Wan, C. Wang, and J. Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 30(12):7749–7762, 2024.

[73] J. Zhang, C. Xu, and B. Li. Chatscene: Knowledge-enabled safety-critical scenario generation for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15459–15469, 2024.

[74] J. Zhang, Q. Zhang, L. Zhang, R. R. Kompella, G. Liu, J. Li, and B. Zhou. Urban scene diffusion through semantic occupancy map. *arXiv preprint arXiv:2403.11697*, 2024.

[75] S. Zhang, Y. Zhang, Q. Zheng, R. Ma, W. Hua, H. Bao, W. Xu, and C. Zou. 3d-scenedreamer: Text-driven 3d-consistent scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10170–10180, 2024.

[76] S. Zhang, M. Zhou, Y. Wang, C. Luo, R. Wang, Y. Li, Z. Zhang, and J. Peng. Cityx: Controllable procedural content generation for unbounded 3d cities. *arXiv preprint arXiv:2407.17572*, 2024.

[77] Y. Zhao, Z. Yan, E. Xie, L. Hong, Z. Li, and G. H. Lee. Animate124: Animating one image to 4d dynamic scene. *arXiv preprint arXiv:2311.14603*, 2023.

[78] M. Zhou, Y. Wang, J. Hou, S. Zhang, Y. Li, C. Luo, J. Peng, and Z. Zhang. Scenex: Procedural controllable large-scale scene generation. *arXiv preprint arXiv:2403.15698*, 2024.

[79] Y. Zhou, J. Huang, X. Dai, S. Liu, L. Luo, Z. Chen, and Y. Ma. Holicity: A city-scale data platform for learning holistic 3d structures. *arXiv preprint arXiv:2008.03286*, 2020.

[80] K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, Y. Zhang, N. Zhenqiang Gong, et al. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv e-prints*, pages arXiv–2306, 2023.