Controllable 3D Outdoor Scene Generation via Scene Graphs Supplementary Material

Contents

A. Evaluation Protocol

In our work, we evaluate the generated scenes based on three aspects: scene quality, control capacity, and user perception through a user study. Specifically, we conduct assessments of scene quality using Fréchet 3D Distance (F3D) [3], evaluate control capacity using MAE and Jaccard metrics, and select 100 pairs of scenes for a user study employing the DMOS [4] (Differential Mean Opinion Score) methodology. The specific methodologies for each evaluation are as follows.

A.1. F3D

F3D [3] is an evaluation method that extends the 2D Fréchet Inception Distance (FID) to 3D for assessing the quality of generated scenes. It measures the similarity between generated scenes and real scenes. A lower F3D value indicates a closer match to the real scenes, suggesting that the model has the capability to generate scenes that closely resemble the real ones. Following the methodology in PDD [3], we use their pre-trained 3D CNN-based autoencoder, which extracts high-dimensional features from the generated 3D scenes. The mathematical formulation of F3D is expressed as:

$$\text{F3D} = \|\mathbf{m}_g - \mathbf{m}_r\|^2 + \mathbf{Tr}\left(\mathbf{C}_g + \mathbf{C}_r - 2(\mathbf{C}_g\mathbf{C}_r)^{1/2}\right) \tag{1}$$

where $\mathbf{m}_g$ and $\mathbf{m}_r$ represent the feature means, and $\mathbf{C}_g$ and $\mathbf{C}_r$ represent the feature covariances of the generated and real scenes, respectively.

A.2. MAE and Jaccard

To evaluate whether the generated scenes match the object quantities and categories defined in the conditional scene graphs, we use Mean Absolute Error (MAE) and the Jaccard Index. MAE quantifies the accuracy of object counts in the generated scenes relative to the conditional scene graphs, while the Jaccard Index measures the degree of category overlap.

The MAE is calculated to compare the object counts for all categories across all scenes. For two distributions, $\mathbf{G}$ (ground truth) and $\mathbf{P}$ (generated scenes), with N scenes and K categories, *MAE* is defined as:

$$MAE = \frac{1}{N \cdot K} \sum_{n=1}^{N} \sum_{k=1}^{K} |G_{n,k} - P_{n,k}| \tag{2}$$

where $G_{n,k}$ and $P_{n,k}$ represent the counts of objects of category k in scene n from the ground truth and generated scenes, respectively. Lower *MAE* values indicate closer alignment between generated and real object distributions.

The Jaccard Index evaluates the category-level overlap between generated scenes and the conditional scene graphs. It is computed based on the binary presence or absence of

Table a. **Scoring Guidelines for User Study.** The table outlines the criteria used by participants to evaluate the alignment of generated scenes with the provided scene graph based on road type, object count, and relative positioning.

Score	Description
5	The road type is accurate, with the number and location of nodes being consistent.
4	The road type is roughly accurate, the number of nodes is consistent, and the relative position is relatively precise.
3	The road type is roughly accurate, with minor deviations in the node count for each category, and the relative position is relatively accurate.
2	The road type is roughly accurate, with significant deviations in the node count for each category and partial errors in the relative position.
1	The road type is incorrect, with significant deviations in the node count for each category and the relative position being relatively inaccurate.

each category in the scene graphs. For N scenes and K categories, J is expressed as:

$$J = \frac{1}{K} \sum_{k=1}^{K} \frac{|\{n : G_{n,k} > 0\} \cap \{n : P_{n,k} > 0\}|}{|\{n : G_{n,k} > 0\} \cup \{n : P_{n,k} > 0\}|} \quad (3)$$

where $\cap$ and $\cup$ represent the intersection and union of sets, respectively. A higher Jaccard Index indicates better agreement between the presence of categories in the generated and conditional scene graphs.

These metrics provide a comprehensive evaluation of the generated scenes, focusing on object quantity accuracy and categorical matching relative to the conditional scene graphs.

A.3. DMOS

In our task, where the goal is to generate 3D outdoor scenes conditioned on an input scene graph, the Differential Mean Opinion Score (DMOS) [4] serves as a crucial metric to evaluate the perceived alignment between the generated scenes and the corresponding scene graph. This metric allows us to compare the quality of generated scenes across different methods in terms of object quantity accuracy, road type consistency, and positional alignment. To conduct the evaluation, we visualize 100 pairs of generated scenes, divided into three groups for comparison: (1) scenes generated by our method ***v.s.*** scenes generated by a Large Language Model (LLM), (2) scenes generated by our method ***v.s.*** randomly generated scenes, (3) scenes generated by the LLM ***v.s.*** randomly generated scenes. For each group, 20 users have participated in the evaluation, using the input scene graph as a reference. Users have assessed through three main criteria: (1) the match between object quantities in the scene and the scene graph, (2) the match between road types in the scene and the scene graph, (3) the positional accuracy of object placement. Each group is evaluated by 20 users, who reference the input scene graph for each pair and assign scores based on three dimensions: alignment of object quantities, road type consistency, and approximate object positioning. Users rate the scenes on a scale from 1 to 5, with scoring guidelines outlined in Table a.

DMOS Calculation. For each group, we compute the DMOS by calculating the score differences between the two methods for each scene pair, as rated by all users. Suppose a group contains n scene pairs and is rated by m users; this results in $n \times m$ score comparisons. The computation for each group is as follows: (1) for each user rating, subtract the score of one method from the score of the other (e.g., our method $-$ LLM, our method $-$ Random, or LLM $-$ Random), (2) collect all $n \times m$ differences and compute the average to obtain the DMOS for the group. This approach quantifies the relative performance of the methods in terms of user preferences, highlighting differences in alignment with the input scene graph. Notably, we visualize one sample from each of the three groups in Figure a.

B. Datasets Pre-processing

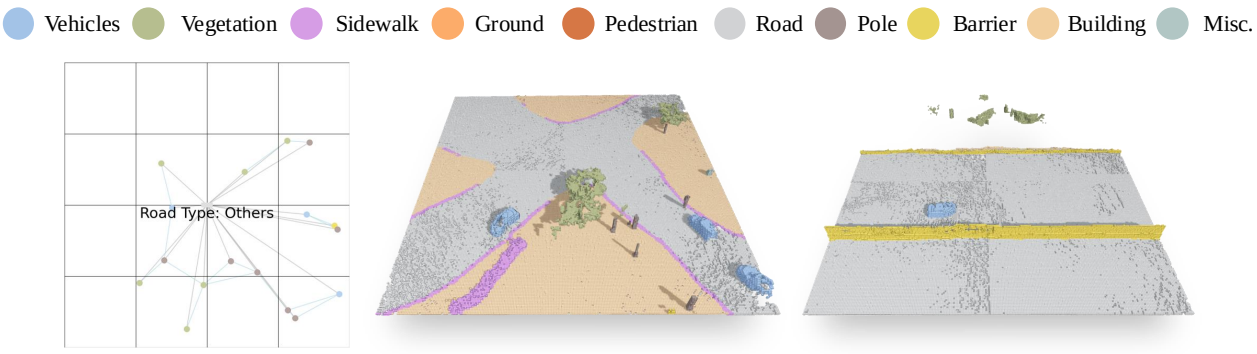
We primarily use the CarlaSC [5] dataset for experiments in our work. However, due to the lack of outdoor 3D LiDAR datasets equipped with scene graphs, we develop a corresponding scene graph dataset for CarlaSC. Using our algorithm, we create a one-to-one mapping between each outdoor scene in CarlaSC and its corresponding scene graph, which we term as *CarlaSG*.

B.1. CarlaSC

CarlaSC is a synthetic dataset containing LiDAR-scanned point cloud data. For our work, we use the voxel-representation version of the dataset. Following the official guidelines, all labels are consolidated into 11 categories, with label 0 representing *unclassified* data. Detailed label information is provided in Table b. The dataset comprises a total of 43,200 scenes, with 32,400 scenes designated for training, and the remaining scenes split for validation and testing. In our experiments, we utilize the highest-resolution version of CarlaSC, with voxel grids of size $256 \times 256 \times 16$. Additionally, we downsample the high-resolution scenes to resolutions of $64 \times 64 \times 8$ and $64 \times 64 \times 1$, which are used for the 3D Discrete Diffusion and 2D Discrete Diffusion stages, respectively.

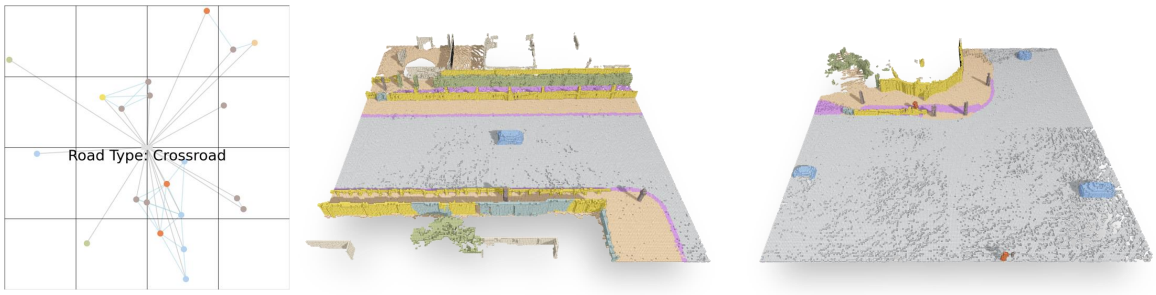
B.2. 3D Semantics to 3D Instances.

We process the original semantic-only 3D scenes into instance-level representations by applying a connected component algorithm combined with handcrafted rules. This approach ensures the distinction between individual objects while addressing challenges posed by LiDAR scanning artifacts and noisy raw data.



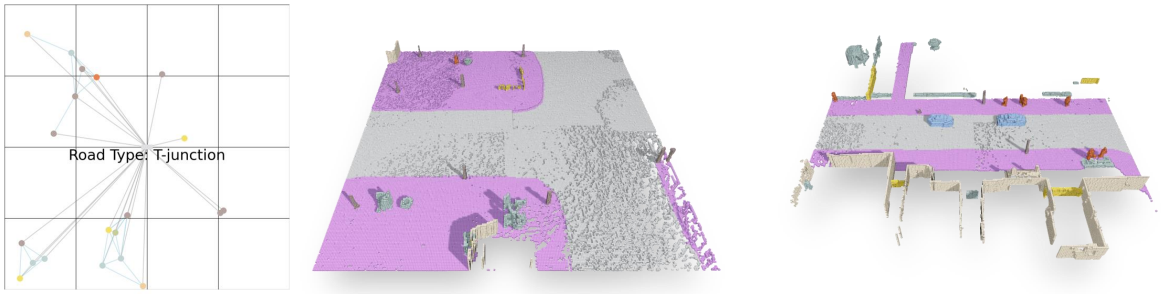
Scene Graph Scene 1 Scene 2

(a) Ours ***vs.*** LLM



Scene Graph Scene 1 Scene 2

(b) Random ***vs.*** LLM



Scene Graph Scene 1 Scene 2

(c) Ours ***vs.*** Random

Figure a. **Examples of Generated Scenes for User Study Evaluation.** Each group compares scene generation methods with different baselines.

B.3. Road Types

We merge the *Ground* and *Sidewalk* labels in the original dataset with the *Road* label and collectively refer to them as *Road*. Based on the general shapes of the original Road data, we split all scenes into six types, as shown in Table c.

B.4. 3D Instances to BEV Scene Graph Data

We transform 3D instance scenes into scene graph data according to our predefined rules. Each scene graph consists of a `nodes` section and an `edges` section. The `nodes` section records every object in the scene, including its *centroid* to represent its position, its *label ID*, and its *road type*. The `edges` section defines connections between objects based on a fixed proximity threshold of 50 voxels, indicating spatial relationships between objects. Additionally, we incorporate road type information into each scene graph, linking non-road objects to road-type nodes to provide global context and enhance structural coherence. This

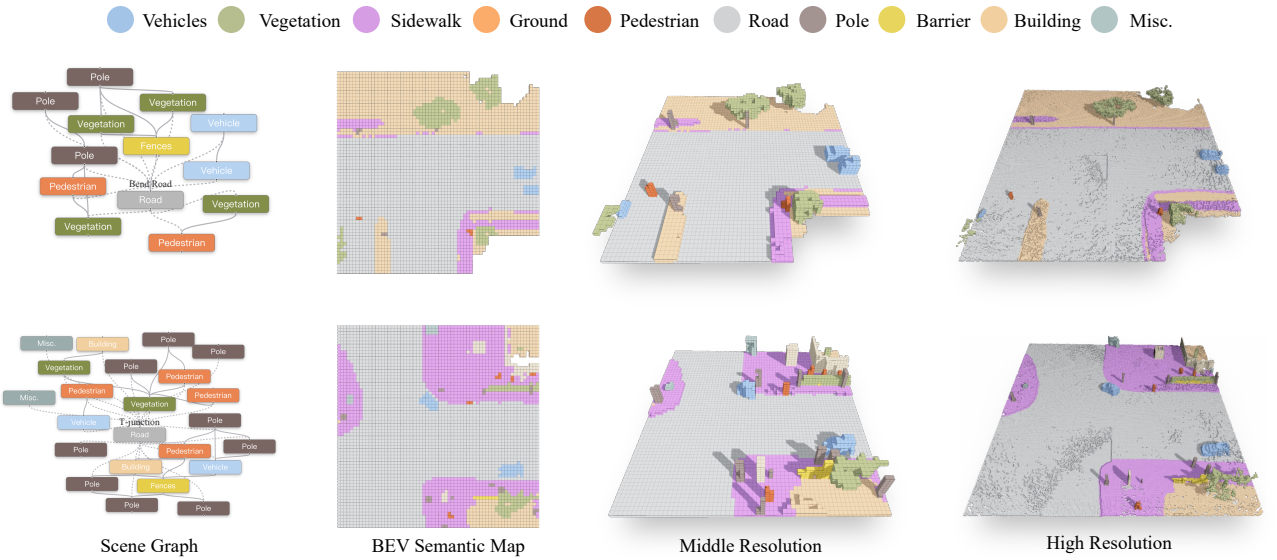


Figure b. **Full Generation Process.** The resolutions of Middle Resolution 3D scene and High Resolution 3D scene are $64 \times 64 \times 8$ and $256 \times 256 \times 16$ respectively.

Table b. **Label Information for CarlaSC Dataset.** The table lists the 10 consolidated categories used in the CarlaSC dataset, including their corresponding indices and labels. Label 0 (*unclassified*) is excluded from the table for clarity.

Index	Label	Index	Label
1	Building	6	Road
2	Fences	7	Ground
3	Other	8	Sidewalk
4	Pedestrian	9	Vegetation
5	Pole	10	Vehicle

Table c. **Road Type Labels.** The table lists the indices and corresponding labels for road types.

Index	Label	Index	Label
0	Unclassified	3	Crossroad
1	Straight Road	4	Bend Road
2	T-junction Road	5	Others

process ensures that each scene graph comprehensively represents all valid objects, their attributes, and their spatial connections within the scene.

C. Hyperparameters Setting

In this section, we provide a detailed overview of the training configurations and hyperparameter settings for various components of our framework. Specifically, We provide a detailed explanation of the joint training process for the 2D Discrete Diffusion Model and the GNN, the training for Localization Head, the 3D Discrete Diffusion Model, and the models used for evaluation. Additionally, we explain the inference phase settings and training of the baseline Graph Language Model for comparative experiments.

C.1. Scene Graph to BEV Semantic Map

Training 2D Discrete Diffusion with GNN. In our method, we jointly train the 2D Discrete Diffusion model and the GNN model to generate BEV Semantic Maps, which serve as the condition for subsequent stages of 3D scene generation. These BEV Semantic Maps have a resolution of $64 \times 64 \times 1$, capturing essential spatial and semantic information in a compact 2D representation. During the joint training process, the parameters of the localization head (LOC) are frozen, and Auxiliary Tasks are employed to enhance the control capabilities. The learning rates for the diffusion tasks, edge reconstruction tasks, and node classification tasks are set as 2.5×10^{-3}, 8.0×10^{-4}, and 8.0×10^{-4}, respectively. Data augmentation is applied to both diffusion and GNN during training. For the diffusion model, augmentation includes operations such as flipping and rotation, while for the GNN model, the ground truth data undergoes node shuffling to introduce variability. To further improve the quality of scene generation, 10% unconditional data is included in the diffusion model training. Additionally, a 30% feature mask is applied to the GNN input data to simulate scenarios where some users might choose not to provide positional information for certain nodes. We use a batch size of 256 and conduct all training on a single RTX 3090 GPU. The training process is conducted for 1,000 epochs.

Localization Head Post-training. After completing the joint training of the 2D Discrete Diffusion model and the GNN model, we freeze their parameters and train the LOC separately. The learning rate for this training phase is set

to 4×10^{-3}, with a batch size of 256. Similar to the joint training process, a 30% feature mask is applied to the GNN input data to simulate scenarios where positional information for certain nodes might be missing. The localization head is trained for 100 epochs.

Inference. During the inference phase, we use Gumbel Softmax with a temperature of 2.0 to introduce controlled randomness into the generated scenes, enhancing diversity while maintaining coherence. The diffusion model is configured with 100 diffusion steps for the generation process, ensuring high-quality outputs.

C.2. 3D Discrete Diffusion

We follow the methodology outlined in PDD [3], dividing the 3D Discrete Diffusion process into two stages: (a) BEV Semantic Map to middle-resolution 3D scenes, and (b) middle-resolution 3D scenes to high-resolution 3D scenes.

Training Hyperparameters. We independently train two models for the two stages of the 3D Discrete Diffusion process. In the first stage, the diffusion model uses the BEV Semantic Map as the condition to generate 3D scenes with resolution $64 \times 64 \times 8$. In the second stage, the diffusion model takes the middle-resolution 3D scenes generated in the first stage as input and produces high-resolution 3D scenes with a voxel grid size of $256 \times 256 \times 16$. We set the batch sizes to 32 for the first stage and 8 for the second stage. Both stages are trained on 4 A100 GPUs, with a learning rate of 10^{-3}. Each model is trained for 1,000 epochs. During training, we apply data augmentation techniques, including rotation and flipping, consistent with the approach described in Sec C.1.

Inference Hyperparameters. For the inference process in both stages of the 3D Discrete Diffusion pipeline, we set the number of diffusion steps to 100.

C.3. Large Language Model

In our experiments, we implement a Large Language Model (LLM) [6, 7] as a baseline for comparison with our method. A 2D deconvolution layer is employed to downstream the dimensionality of the text embeddings to a size compatible with our 2D Discrete Diffusion model, enabling it to serve as a control condition for scene generation.

Embedding Extraction. To leverage a language model for extracting scene graph embeddings, we first process each scene graph in the CarlaSG dataset into a JSON-formatted file, as illustrated in Table d. After this formatting step, the structured JSON data is programmatically converted into a natural language description following the template:

> In a scene with a road type of `<road_type>`, there are `<num>` `<label_name>` objects located at `<centroids>`, and `<num>` `<label_name>` objects located at `<centroids>`, and ....
>
> Among them, the `<label_name>` at `<centroid>` and the `<label_name>` at `<centroid>` are close to each other, and the `<label_name>` at `<centroid>` and the `<label_name>` at `<centroid>` are also close to each other, and ....

Subsequently, we use OpenAI's official `text-embedding-3-large` model to process each scene description, obtaining a corresponding embedding for every scene. Each embedding has a dimensionality of 3072.

Table d. **Example of a formatted scene graph in JSON format.** Showing nodes with attributes such as `instance_id`, `label_id`, `centroid`, and `road_type`, and edges defining relationships between instances.

```
{
    "nodes": [
        {
            "instance_id": 1,
            "label_id": 1,
            "centroid": (28, 34, 3),
            "road_type": 0
        },
        {
            "instance_id": 2,
            "label_id": 6,
            "centroid": (56, 124, 4),
            "road_type": 1
        },
        ...
    ],
    "edges": [
        {
            "instance_id_1": 1,
            "instance_id_2": 6
        },
        {
            "instance_id_1": 2,
            "instance_id_2": 6
        },
        ...
    ]
}
```

Training Hyperparameters. We implement a 2D deconvolution layer to reduce the original 3072-dimensional text embedding to 32 dimensions, aligning it with the input requirements of our 2D Discrete Diffusion model. During the training phase, we set the learning rate to 10^{-3}, the batch size to 256, and train the model for 1,000 epochs.

C.4. Evaluation Models

We evaluate the quality of the generated scenes using mIoU, MA, and F3D [3] metrics. To enable these evaluations, we train corresponding models tailored for each metric. Specifically, we follow the methodology outlined in [3] to train a SparseUNet [1] model for semantic segmentation, which is used to compute mIoU and MA. Additionally, we train a 3D CNN model to extract high-dimensional features for F3D evaluation.

SparseUNet. SparseUNet is employed for voxel-based semantic segmentation, enabling the calculation of mIoU and MA metrics. The model is trained on the training set of the CarlaSC dataset. We use cross-entropy as the loss function. During training, we use a batch size of 16, SGD as the optimizer, and a cosine scheduler for learning rate adjustment. The learning rate is set to 10^{-2}, and the model is trained for 30 epochs.

3D CNN. As described in Sec A.1, the calculation of F3D requires training a 3D CNN network for feature extraction. We utilize the CarlaSC dataset as the training dataset and adopt balanced cross-entropy as the loss function. The training process employs a batch size of 16, SGD as the optimizer, and a cosine scheduler for learning rate adjustment. The learning rate is set to 10^{-2}, and the model is trained for 30 epochs.

C.5. Text Interaction

In our Interactive System, we enable the generation of scene graphs through text interaction by invoking Large Language Model (LLM) APIs. In addition to user-provided prompts, we incorporate system-level prompts that are combined with the user's input as part of the LLM request. The system-level prompt for generating a scene graph is shown in Table e.

D. Additional Experiment Results

D.1. Qualitative Results

We present additional qualitative comparisons of scenes generated using scene graphs as conditions under both the LLM and our method in Figure c and Figure d. The results demonstrate that our method effectively and accurately captures the information provided in the scene graph, including object quantities and road types. In contrast, the LLM approach exhibits significant mismatches, such as missing objects and incorrect road types. These discrepancies highlight the limitations of the LLM in faithfully aligning the generated scenes with the input scene graph conditions, emphasizing the superior capability of our method in controlled 3D scene generation.

D.2. Full Generation Process

We visualize the intermediate outputs of our controlled 3D scene generation method in Figure b. In our approach, the scene graph is first processed using the GNN and Allocation Module to produce a BEV Embedding Map, which guides the 2D Discrete Diffusion model to generate the BEV Semantic Map. The generated BEV Semantic Map serves as the condition for generating middle-resolution 3D scenes. Subsequently, the middle-resolution 3D scene is used as the condition for producing high-resolution 3D scenes. Both the middle-resolution and high-resolution stages employ 3D Discrete Diffusion models for generation. This multi-stage pipeline enables the progressive generation of high-quality 3D scenes that align with the input scene graph.

E. Interactive System

We design an interactive system to enable users to intuitively and efficiently add, delete, and modify parameters within a scene graph. The system also allows users to directly generate the final 3D scene from the defined scene graph within the interface. Detailed operation steps are provided in the demonstration video file submitted as part of the supplementary materials.

E.1. Scene Graph Design Interface

Figure e shows our scene graph design interface. There are five key functional areas: *Toolbar* (Area 1): This area allows users to create new nodes, select the road type for the current scene, and includes a *Generate* button to produce the BEV Semantic Map based on the defined scene graph. *Workspace* (Area 2): Users can construct the scene graph in this area and select the positional parameters for each node, enabling precise scene layout adjustments. *Status Area* (Area 3): This section provides project-related information, such as the current project name and settings. *History Area* (Area 4): Users can save constructed scene graphs as history records and restore them at any time for further editing. *Text-to-Scene Graph Generation Area* (Area 5): Users can input text to describe the desired scene graph, enabling control through natural language interaction.

E.2. BEV Semantic Map Interface

After completing the construction of the scene graph and clicking the *Generate* button, the BEV Semantic Map inferred using the scene graph as the condition will be displayed, as shown in Figure f. Area 3 showcases the generated BEV Semantic Map, while Area 1 and Area 3 remain the *toolbar* and the *status bar*, respectively. If users wish to fine-tune the generated BEV Semantic Map, they can select the corresponding label brush in Area 1 and make adjustments directly in Area 3. Note that this fine-tuning functionality is currently in beta testing. Once satisfied, users

can proceed to generate the middle-resolution 3D scene.

E.3. 3D Outdoor Scene Display Interface

Once users confirm the BEV Semantic Map they wish to use for generating the final 3D scene, they can first generate the middle-resolution 3D scene (Figure g) and then proceed to generate the high-resolution 3D scene (Figure h). Area 1 serves as the status area, while Area 2 is dedicated to displaying the 3D models.

F. Additional Discussion

F.1. Related Work and Indoor-Outdoor Differences

Indoor and outdoor scene generation differ fundamentally, making it impractical to directly adopt indoor pipelines as baselines for our method. We summarize these differences in three aspects:

- **Spatial Characteristics:** Indoor scenes are enclosed and structured, whereas outdoor scenes are expansive, unconstrained, and contain a greater variety of objects. This results in fundamentally different scene layouts and spatial distributions.
- **Data Representation:** Outdoor scenes primarily rely on LiDAR-captured point clouds, which naturally align with 3D voxel representations but lack textures. In contrast, indoor scenes use RGB-D or multi-view data for textured surface reconstruction. Outdoor environments also introduce unique challenges, such as non-discrete backgrounds (e.g., roads, terrain) and incomplete structures (e.g., buildings), where occlusions and discontinuities make surface-based representations ineffective.
- **Scene Generation Targets:** Outdoor scene generation primarily employs segmentation-like methods to model scene-level topology, capturing relationships between background elements and objects. In contrast, indoor generation relies on detection-like approaches, which focus on recognizing and placing discrete object instances with structured relationships.

F.2. The Choice of Evaluation Metrics

Given that our data representation and generation task closely align with PDD [3], a published and open-source work on outdoor scene generation, we follow its evaluation framework. Specifically, we adopt F3D, mIoU, and MA to assess scene generation quality and semantic consistency.

F3D is an adaptation of the Fréchet Inception Distance (FID) [2] designed for semantic 3D data. Unlike FID, which operates on RGB images, F3D is tailored for 3D outdoor semantic scenes that typically lack texture. We highlight that using RGB image-based FID is unsuitable for our task, as outdoor voxel-based representations do not contain color information. In contrast, F3D evaluates feature distributions in a way that better reflects structural and semantic fidelity. Therefore, adopting F3D alongside mIoU and MA provides a more meaningful and task-specific evaluation of generated outdoor scenes.

Table e. **System-level prompt for generating scene graphs.** The prompt ensures structured and accurate scene graph generation based on user descriptions, adhering to predefined rules for object categories, positions, and relationships.

System Prompt of Generation Process

<System>: Given the user's description of a scene, generate a scene graph in the following JSON format:

```
{
  "nodes": [
     {
        "instance_id": <instance_id>,
        "label_id": <label_id>,
        "position": <according to rules>,
        "road_type": <only for the road node, according to rules>
     },
     ...
  ],
  "edges": [
     {
        "instance_id_1": <instance_id>,
        "instance_id_2": <instance_id>
     },
     ...
  ]
}
```

Rules for Generation:

1. Scene Graph Structure:
  a. Each object in the scene is represented as a node under "nodes".
  b. Each pair of objects with close proximity is represented as an edge under "edges".

2. Object Categories:
  a. The label_id must correspond to one of the following predefined categories. Ignore objects that fall outside this range:
  {1: Building, 2: Fence, 3: Other, 4: Pedestrian, 5: Pole, 6: Road, 7: Ground, 8: Sidewalk, 9: Vegetation, 10: Vehicle}.

3. Instance IDs:
  Assign a unique instance_id to every node, starting from 1 and incrementing for each new object.

4. Position Assignment:
  Divide the scene into 16 quadrants as follows:
  [13][14][15][16]
  [9][10][11][12]
  [5][6][7][8]
  [1][2][3][4]
  Assign the position based on the user's description of object locations. If the user does not specify, assign the position freely.

5. Proximity and Edges:
  If two objects are described as being close to each other, ensure their positions are adjacent and establish an edge connection.
  Use the instance_id of the two objects to define this edge.

6. Road Node Requirements:
  a. There must always be exactly one node with label_id: 6 (representing "Road").
  b. If the user does not specify the road type, assign a random road_type from the following options:
  {1: Straight Road, 2: T-junction Road, 3: Crossroad, 4: Bent Road, 5: Others}.
  c. All objects must establish an edge connection with the road node.

7. Output Restrictions:
  The output must strictly conform to the above JSON format. No additional text, comments, or explanations should be included.

Generate the scene graph based on the above requirements.

<User>: ......

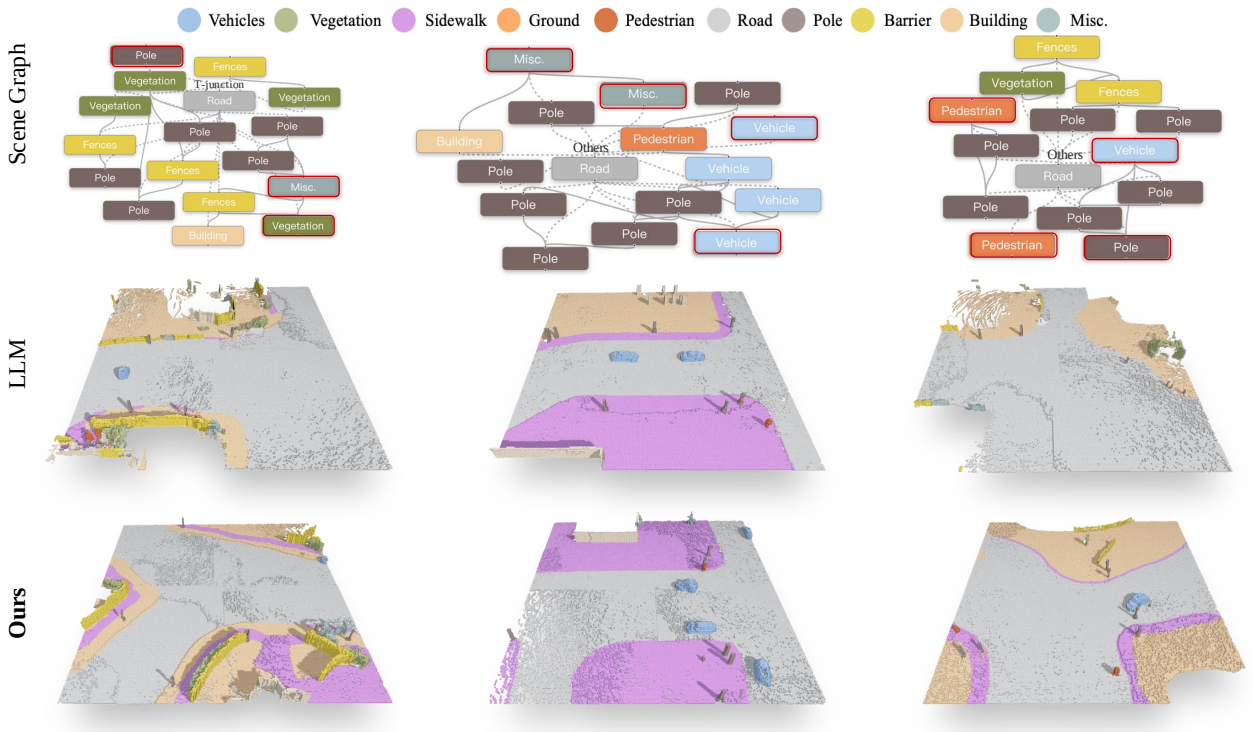


Figure c. **Controlling 3D Outdoor Scene Generation with Scene Graphs.** We compare baseline method – LLM generation. We mark the objects missed by the baseline LLM in red boxes.

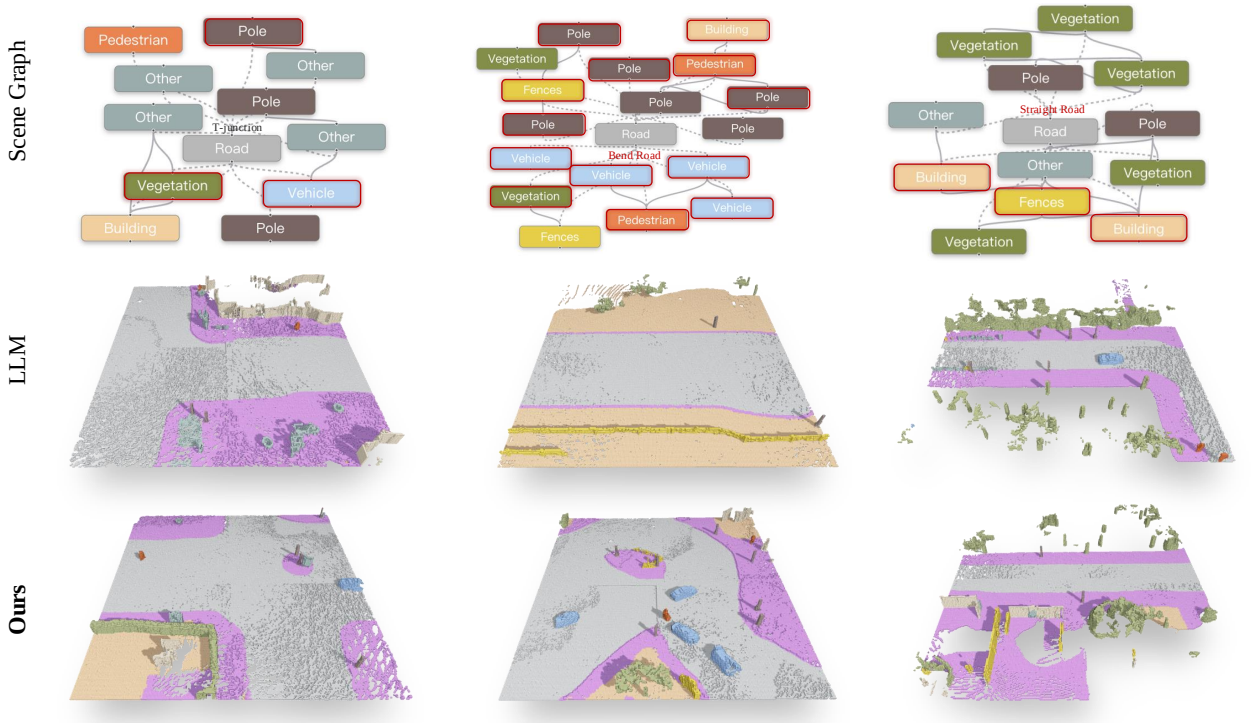


Figure d. **Controlling 3D Outdoor Scene Generation with Scene Graphs.** We compare baseline method – LLM generation. We mark the objects missed by the baseline LLM in red boxes.

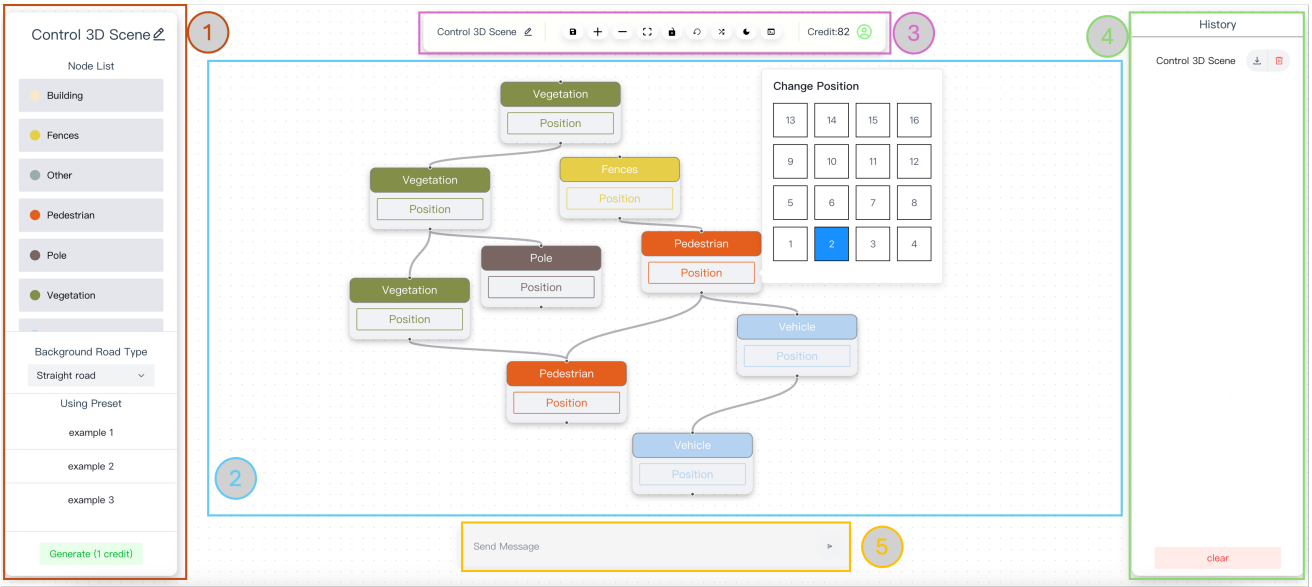


Figure e. **Interactive Scene Graph Design Interface.** Area 1: *Toolbar*, Area 2: *Workspace*, Area 3: *Status Area*, Area 4: *History Area*, 5: *Text-to-Scene Graph Generation Area*.

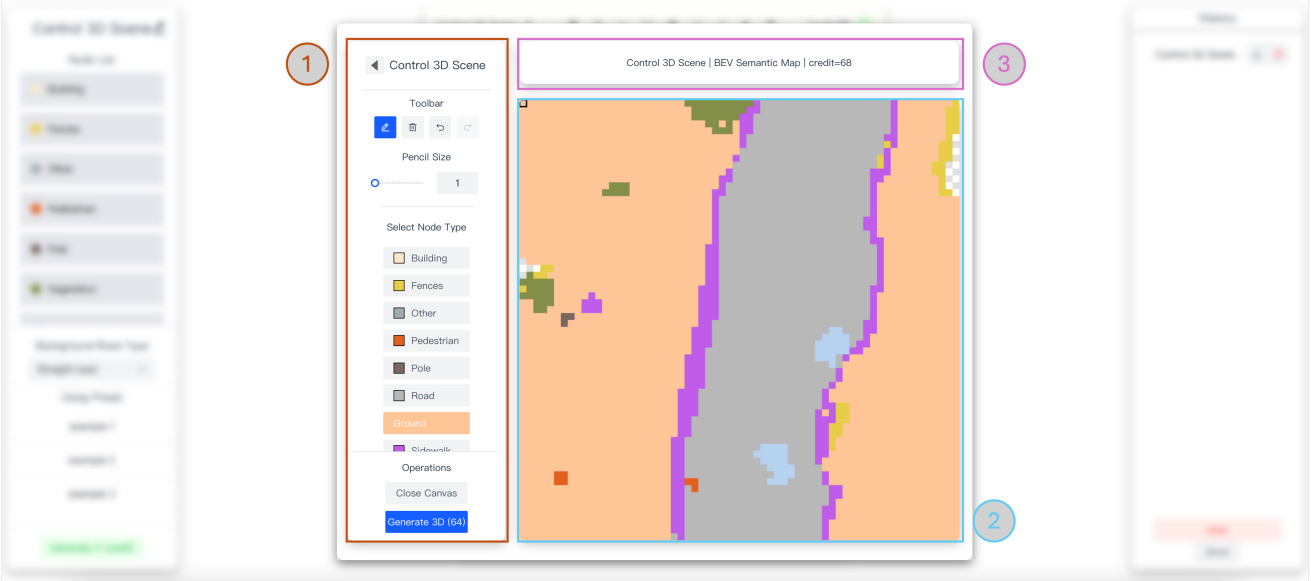


Figure f. **Interactive BEV Semantic Map Interface.** Area 1: *Toolbar*, Area 2: *Workspace*, Area 3: *Status Area*.

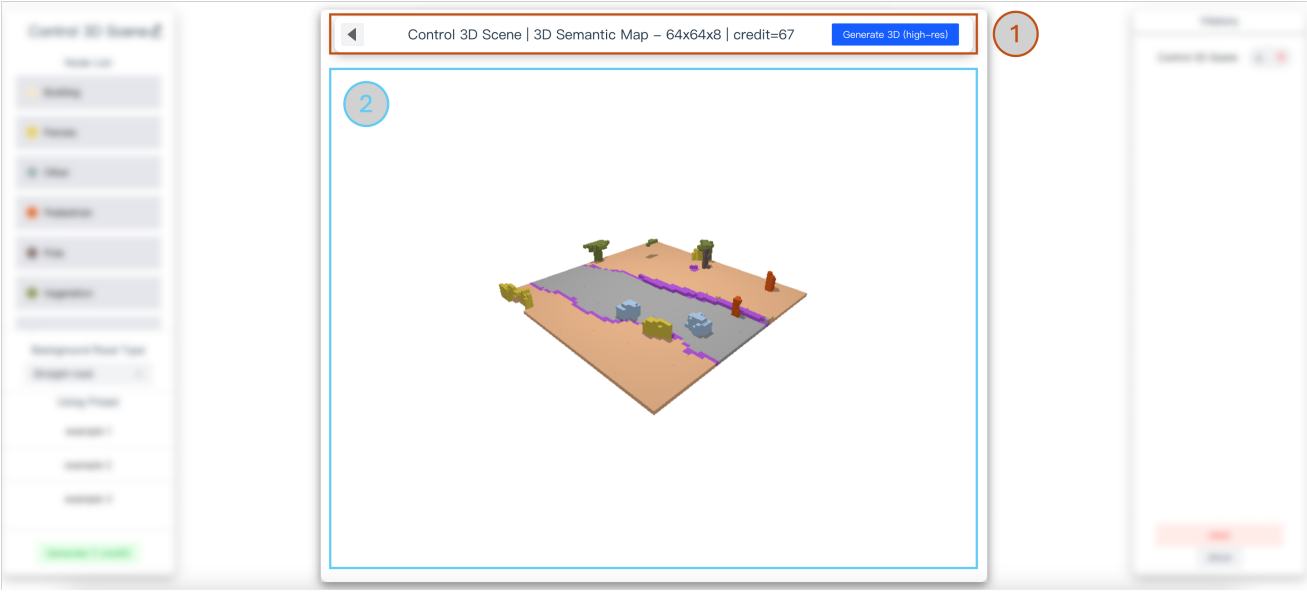


Figure g. **3D Middle-Resolution Outdoor Scene Display Interface.** Area 1: *Status Area*, Area 2: *Model Display Area*. The middle-resolution of 3D outdoor scenes is $64 \times 64 \times 8$.

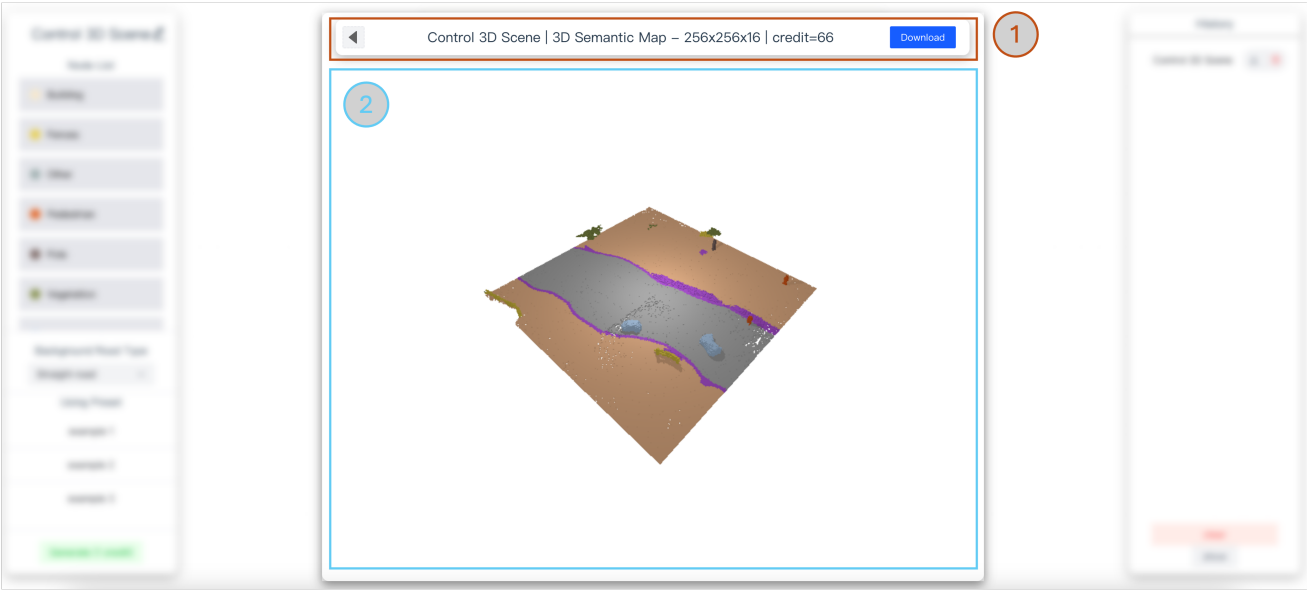


Figure h. **3D High-Resolution Outdoor Scene Display Interface.** Area 1: *Status Area*, Area 2: *Model Display Area*. The high-resolution of 3D outdoor scenes is $256 \times 256 \times 16$.

References

[1] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 6

[2] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 2017. 7

[3] Yuheng Liu, Xinke Li, Xueting Li, Lu Qi, Chongshou Li, and Ming-Hsuan Yang. Pyramid diffusion for fine 3d large scene generation. *ECCV*, 2024. 1, 5, 6, 7

[4] L.A. Thorpe and B.R. Shelton. Subjective test methodology: Mos vs. dmos in evaluation of speech coding algorithms. In *Proceedings., IEEE Workshop on Speech Coding for Telecommunications,*, 1993. 1, 2

[5] Joey Wilson, Jingyu Song, Yuewei Fu, Arthur Zhang, Andrew Capodieci, Paramsothy Jayakumar, Kira Barton, and Maani Ghaffari. Motionsc: Data set and network for real-time semantic mapping in dynamic environments. *IEEE Robotics and Automation Letters*, 2022. 2

[6] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*, 2024. 5

[7] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*, 2023. 5