Incremental3D: Incremental 3D Scene Generation with Scene Graph for Immersive Teleoperation

Anonymous authors
Paper under double-blind review

Abstract

Graph-based 3D scene generation aims to synthesize 3D environments conditioned on scene graphs and has been widely explored in applications such as 3D gaming and interior design. However, its potential for immersive robotic teleoperation has been largely overlooked. In this setting, transmitting lightweight incremental 3D scene graphs from the robot-side to the operator-side is far more bandwidth-efficient and lower-latency than streaming raw RGB or point-cloud data. At the same time, recent advances in robot-side 3D scene-graph learning now make such incremental scene-graphs readily obtainable from RGB-D inputs. Despite this opportunity, existing scene-graph-based 3D scene generation methods are fundamentally single-shot: inserting even a single new object requires regenerating the entire scene. This global re-computation incurs prohibitive latency and renders existing approaches unsuitable for real-time immersive robotic teleoperation, where the scene graph, and therefore the scene itself, is built and generated incrementally as the robot moves through the environment. To address this limitation, we propose *Incremental3D*, the first framework capable of incremental graph-to-3D scene generation for teleoperation applications. *Incremental3D* augments an existing scene graph with a global context node that maintains a holistic representation of the evolving environment. At each update step, the global context node aggregates the global information and conditions the generation of newly added objects, enabling geometry synthesis and spatial prediction without recomputing unchanged regions. Extensive experiments demonstrate that *Incremental3D* achieves 38 Hz generation speed while maintaining high spatial accuracy, indicating its suitability for real-time teleoperation and other latency-sensitive 3D applications.

1 Introduction

The aim of graph-based 3D scene generation Dhamo et al. (2021); Zhou et al. (2019); Zhai et al. (2023; 2024); Yang et al. (2025); Chattopadhyay et al. (2023) is to synthesize 3D scenes conditioned on scene graphs, where nodes encode object attributes (e.g., position, orientation, size) and edges represent explicit inter-object relationships (e.g., same-material-as, left-of, larger-than). This paradigm has been applied in game development, interior design, and other related domains. However, its potential for immersive robotic teleoperation Tefera et al. (2024); Naceri et al. (2021); Darvish et al. (2023) has been largely overlooked. Latency (time delay) is a long-standing problem in teleoperation Farajiparvar et al. (2020); Chai & Necsulescu (2021); Sakib et al. (2023), and traditional teleoperation systems that rely on continuous image and/or 3D data streaming from the robot side to the operator require substantial bandwidth and incur significant end-to-end delays—often ranging from hundreds of milliseconds to several seconds Pop et al. (2022). Point-cloud-based streaming is more problematic, with latency extending to minutes due to the extremely high data volume. Such delays make these approaches unsuitable for responsive and safety-critical real-time teleoperation. In contrast, a low-latency, low-bandwidth pipeline may be achievable by transmitting lightweight incremental 3D scene-graph updates, as illustrated in Figure 1, instead of compressed sensory data.

At the same time, recent advances in simultaneous localization and mapping (SLAM) Hughes et al. (2022); Maggio et al. (2024) and 3D scene-graph learning Wu et al. (2021); Feng et al. (2025); Renz et al. (2025) make it feasible to obtain scene graphs directly from robot-side RGB-D or point-cloud inputs. For instance,

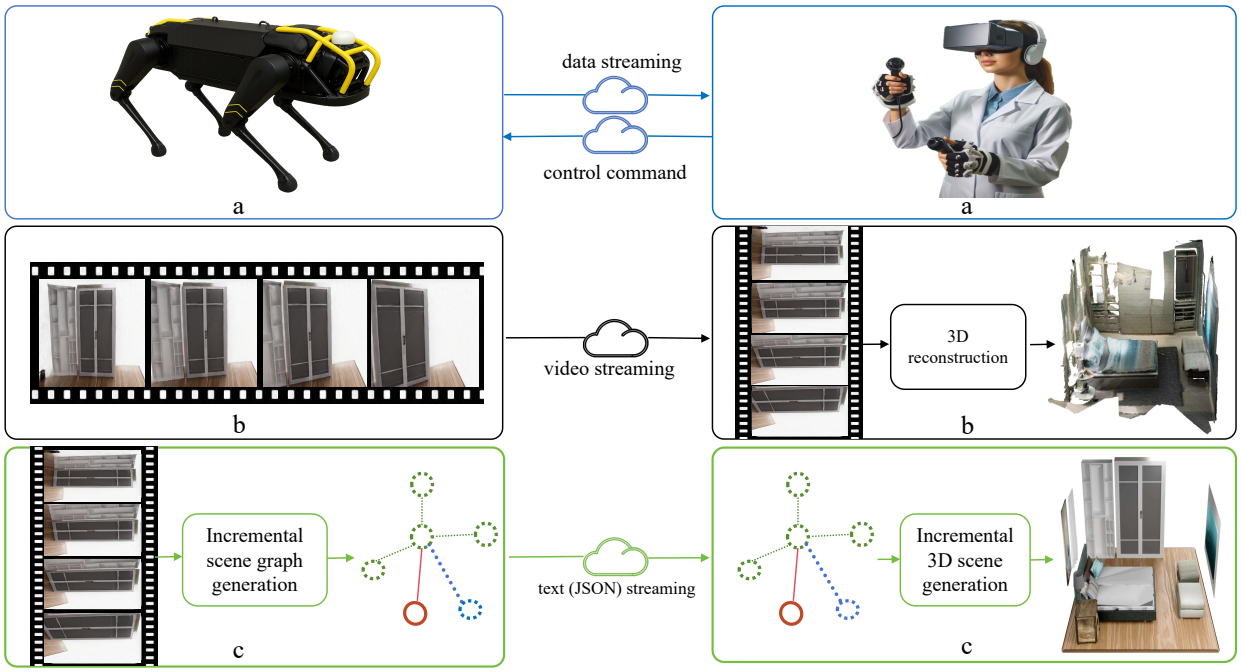


Figure 1: The typical teleoperation pipeline is shown in subfigure (a). In traditional systems, videos (even point clouds) are streamed from the robot to the operator side, and a 3D map is reconstructed to provide an immersive view, as illustrated in subfigure (b). In contrast, our system streams lightweight 3D scene graphs (in JSON format), reducing both latency and bandwidth, as shown in subfigure (c). On the robot side, incremental 3D scene graphs are constructed online from RGB-D images as the robot moves Wu et al. (2021); Feng et al. (2025); Renz et al. (2025); on the operator side, immersive 3D scenes are generated from these graphs using our proposed method. The system is expected to operate reliably in well-organized environments.

SceneGraphFusion Wu et al. (2021) incrementally constructs 3D scene graphs in indoor environments by segmenting each incoming RGB-D frame with camera trajectories and using a graph neural network (GNN) to infer semantic labels and pairwise relationships. The detailed pseudocode for the incremental scene graph construction pipelines are provided in the appendix A.1. The resulting scene-graph summaries can be efficiently transmitted to the operator side over standard 4G or WiFi connections. Thus, incremental scene graphs are already available on the robot side Wu et al. (2021), making them a practical and attractive representation for teleoperation.

Despite this opportunity, existing graph-based 3D scene generation approaches, i.e., reconstructing the 3D scenes from the generated 3D scene-graphs, seem fundamentally unsuited for teleoperation. As the robot-camera moves, the operator side of the teleoperation system receives a sequence of incremental graph updates, yet all prior methods Dhamo et al. (2021); Zhou et al. (2019); Zhai et al. (2023; 2024); Yang et al. (2025); Chattopadhyay et al. (2023) regenerate the entire 3D scene upon insertion of a single new object, causing prohibitive delays. Even approaches that support scene manipulation still reconstruct the full scene after each node addition. For example, CommonScenes Zhai et al. (2023) requires over 20 seconds to generate a single room-scale scene. This paradigm mismatch—incremental perception versus single-shot generation—renders existing models unsuitable for latency-sensitive application like immersive teleoperation. Moreover, most generative methods Dhamo et al. (2021); Zhou et al. (2019); Zhai et al. (2023; 2024); Yang et al. (2025); Chattopadhyay et al. (2023) sample new scenes from learned distributions rather than recover the original environment structure, further limiting applicability in immersive teleoperation.

To address this gap, we introduce *Incremental3D*, a framework for incremental 3D scene-generation from scene graphs, designed for latency-sensitive applications such as teleoperation. The key contribution is the

concept of *global context embedding*, inspired by the CLS (short for *classification*) token in BERT Devlin et al. (2019) and ViT Dosovitskiy et al. (2020). We adapt this mechanism to incremental scene graphs by introducing a global context node, the *CLS node*, that aggregates holistic scene information and broadcasts it to newly added nodes. When a new object arrives in the scene, *Incremental3D* conditions on both its local features and the global context embedding, enabling geometry synthesis and spatial placement without recomputing unchanged regions of the scene. A second design choice focuses on high-fidelity geometry. Since teleoperation requires manipulation-ready shapes and layouts, we adopt an autoencoder Li et al. (2023) instead of sampling-based generative models such as VAEs or diffusion models Zhai et al. (2023; 2024); Yang et al. (2025), which emphasize diversity over accuracy.

Because no incremental 3D scene datasets exist, we construct incremental sequences from SG-FRONT Zhai et al. (2024) using a principled pipeline consisting of: (1) clustering objects based on Euclidean proximity; (2) constructing a cluster graph to preserve spatial adjacency; and (3) determining an incremental insertion order using a Hamiltonian-path approximation. We further adopt a curriculum-learning strategy that gradually increases sequence length and difficulty, improving accuracy on long-horizon incremental updates. Through comparison with three graph-based scene generation baselines, our method provides a competitive solution directed towards teleoperation, achieving 38 Hz scene generation, while preserving high layout accuracy and geometric fidelity.

Our contributions are as follows:

1. A new problem setting: we introduce incremental graph-to-3D scene generation aimed at teleoperation, motivated by a low-latency system and real-time incremental scene-graph perception.
2. Incremental3D: a real-time framework capable of generating 3D scenes from incremental graph updates, achieving 38 Hz generation while preserving high spatial and geometric accuracy.
3. Introducing a curriculum-learning strategy that progressively increases scene complexity, improving accuracy on long incremental sequences.
4. Extensive experiments showing that *Incremental3D* consistently outperforms single-shot baselines in layout accuracy, geometric accuracy, and real-time performance.

2 Related Work

A. 3D Scene Graphs Construction: Representing complex 3D environments requires a structured way to capture both the objects present and the relationships that organize them. One effective strategy is to use graph-based abstractions, which model spatial and semantic interactions in a unified framework. Earlier methods in this domain predominantly focused on reconstructing a full scene graph from complete environment data, relying on point clouds Wald et al. (2020), 3D meshes Armeni et al. (2019), or image collections Gay et al. (2019). More recently, researchers have moved to *incremental* graph construction, in which the graph is updated continuously as new frames arrive. For instance, Wu *et al.* Wu et al. (2023) combine geometric segmentation with graph learning to build scene graphs online from RGB-D streams. This paradigm has been extended to large-scale environments: from room- to building-level Hughes et al. (2022); Maggio et al. (2024), and even to large-scale outdoor environments Deng et al. (2024).

B. 3D Scene Generation from Scene Graphs: Building on their strong representational capabilities, scene graphs have become a central intermediate structure for generating 3D environments. Their compact yet expressive formulation allows generative models to condition on both object-level attributes and relational constraints, enabling the synthesis of coherent, semantically grounded scenes. The Graph-to-3D Dhamo et al. (2021) was the first method to generate 3D geometry directly from a scene graph in an end-to-end way, decoding predicted shape codes with DeepSDF Park et al. (2019). CommonScenes Zhai et al. (2023) replaced the codebook with a latent diffusion model, while EchoScene Zhai et al. (2024) introduces an "information-echo" mechanism that lets nodes information-share diffusion progress. MMGDreamer Yang et al. (2025) augments text-based graphs with visual cues to improve fidelity. These methods, aimed mainly at architectural or interior design, rely on VAE Kingma et al. (2013); Kusner et al. (2017) or diffusion priors

Sohl-Dickstein et al. (2015). Random latent sampling of those methods, however, can yield diverse object appearances. In addition, they usually regenerate the entire scene even for a single new node insertion. By contrast, our method aims to provide real-time, purely local generation for 3D objects, incrementally, with high accuracy.

C. Global Context Embedding: Understanding or generating complex structures often requires more than just local information; local features alone can be ambiguous or inconsistent without knowledge of how they fit into the broader whole. Capturing information that extends beyond local neighbourhoods is therefore essential for maintaining semantic consistency, enforcing long-range dependencies, and ensuring that local predictions align with the global structure, whether in language, vision, or 3D environments. Early approaches often relied on simple descriptive statistics to obtain a global summary. For example, max pooling Reimers & Gurevych (2019) retains only the most dominant local feature, potentially discarding important contextual cues, while mean pooling assigns equal weight to all elements, failing to reflect the fact that nearby or semantically relevant objects typically exert a stronger influence on newly generated content. More advanced methods replaced these fixed aggregation rules with learnable mechanisms. Attention pooling Zhang et al. (2021), for instance, introduces a set of learnable query vectors that attend to the entire sequence via multi-head attention, selectively aggregating salient information. However, such approaches require storing all previously generated features and attending to them all, which limits scalability in incremental settings. A widely adopted alternative is the CLS token, used in BERT Devlin et al. (2019). During self-attention, the CLS token interacts with every other token, absorbing contextual information that is later used for classification. Vision Transformer (ViT) Dosovitskiy et al. (2020) employs the same device, treating the CLS embedding as a global image descriptor. This mechanism is now standard for capturing global context in Transformers Wang et al. (2024), powering tasks such as sentiment analysis, text categorization, intent recognition, etc. In our proposed framework, the CLS embedding is used to represent the entire 3D scene; newly inserted objects are generated conditionally on this global vector, enabling efficient local updates.

D. Curriculum Learning: Learning complex tasks directly from difficult examples can overwhelm a model, lead to unstable optimization, or hinder its ability to generalize. Curriculum learning (CL) Bengio et al. (2009) addresses this challenge by structuring the training process in a way that mirrors human learning: starting with simpler, more easily learnable examples and gradually progressing toward harder ones. In domains such as computer vision and natural language processing, CL has proven highly effective in improving model generalization, training stability, convergence speed, and final accuracy Wang et al. (2021).

3 Methodology

Overview The *Incremental3D* scene generation framework is seen in Figure 2. The incremental scene graph is defined as $G = (O_{\text{old}}, R_{\text{old}}; O_{\text{new}}, R_{\text{new}})$ Wu et al. (2021); Feng et al. (2025); Renz et al. (2025), where O_{old} and R_{old} denote the objects and relationships accumulated from previous timesteps, and O_{new} and R_{new} represent the newly added ones. After the operator side receives an incremental scene graph G from the robot-side of the teleoperation system(see Figure 1-c), new 3D objects could be generated in real-time based on both their local features and the global embedding, and maintain high 3D scene accuracy through *Incremental3D*. This system follows an auto-encoder design. The **encoder** (Sec. 3.1) produces embeddings for newly added nodes and constructs a global scene embedding E_g. This embedding is derived from a *CLS* node, which serves as a global context aggregator by capturing the information of newly inserted nodes and edges at each step. The **decoder** (Sec. 3.2) consumes new node embeddings together with this global scene embedding E_g to produce, for each new object, its position, size, orientation (p, s, a) and a shape-specific latent code. The resulting new object latent codes are used to retrieve semantically and geometrically matching 3D models from a database and the retrieved objects are then placed and scaled according to the predicted geometric parameters. To improve the incremental generation ability of network in further steps, it is trained by a simple curriculum learning strategy based on the number of incremental steps (Section 3.3).

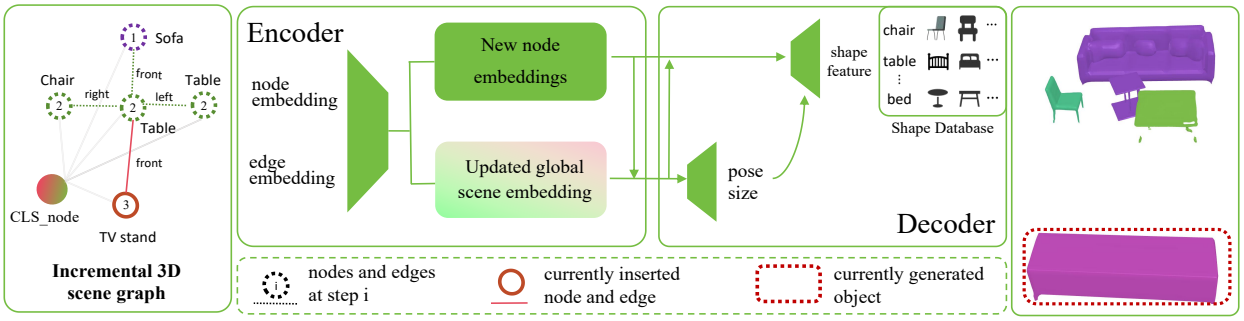


Figure 2: Proposed *Incremental3D* Scene Generation Framework. After receiving incremental 3D scene graphs from the robot-side of a teleoperation system, the framework can generate 3D scenes incrementally, in real-time and with high accuracy Wu et al. (2021); Renz et al. (2025). At each insertion step, a CLS node captures information of newly added nodes and edges to form a global 3D scene embedding in the encoder. Then, new objects are generated incrementally with high accuracy, through database retrieval, based on their own features combined with this global embedding in the decoder.

3.1 3D Scene–Graph Encoder

The encoder E_c is a *triplet* graph–convolutional network (T–GCN) that maps a 3D incremental scene-graph $G = (O_{\text{old}}, R_{\text{old}}; O_{\text{new}}, R_{\text{new}})$ to (i) new node embeddings ϕ_{v_i} and (ii) a scene–level latent E_g. Each node is $n_i = \big(o_i, p_i, s_i, a_i, \text{shape}_i\big)$, where o_i is the semantic class and $(p_i, s_i, a_i, \text{shape}_i)$ are position, size, orientation and shape features, respectively.

Inspired by the CLS token in BERT Devlin et al. (2019) and ViT Dosovitskiy et al. (2020)—which introduces an extra token attending to all words or patches to capture global context—we add a CLS node to the incremental 3D scene graph and connect it bidirectionally to all newly inserted object nodes. Moreover, old nodes that are directly connected to the new ones are also linked to the CLS node, enabling it to capture information from the newly formed edges. The resulting augmented subgraph is then processed by a graph neural network (GNN) . For every directed triplet $(v_i \xrightarrow{e_{i\to j}} v_j)$ of GNN, a layer l performs three operations:

1. Message passing and Edge updates

$$\big(\psi_i^l,\ \phi_{e_{i\to j}}^{l+1},\ \psi_j^l\big) = g_1\big(\phi_{v_i}^l,\ \phi_{e_{i\to j}}^l,\ \phi_{v_j}^l\big), \tag{1}$$

   with a shared MLP g_1.

2. Neighbor aggregation

$$\rho_i^l = \frac{1}{M_i}\left(\sum_{j\in\mathcal{R}_{\text{out}}} \psi_{\text{out},ij}^l + \sum_{j\in\mathcal{R}_{\text{in}}} \psi_{\text{in},ji}^l\right), \tag{2}$$

   where M_i is the total number of incident relations of node i. R_{in}, R_{out} are the set of edges of the node as out(in)-bound objects.

3. Node update

$$\phi_{v_i}^{l+1} = \phi_{v_i}^l + g_2\big(\rho_i^l\big), \tag{3}$$

   with another MLP g_2.

At each insertion step, the CLS node gathers the information from new nodes and new edges, updating its hidden state to yield a compact, up-to-date scene embedding E_g.

3.2 3D Scene Decoder

The input of decoder is class label o_i, node embedding ϕ_{v_i}, and global scene embedding E_g. The output consists of the predicted bounding box, $(p_i,\ s_i,\ a_i)$, and shape code, $shape_i$, for the new object. During this stage, bidirectional edges are established between the CLS node and the newly added nodes, enabling the decoder to exploit global scene context for object generation. To preserve the information carried by the new edges, the CLS node is also connected bidirectionally to old nodes that are directly linked to the new nodes. The resulting subgraph is then processed by two consecutive triplet graph convolutional network (GCN) blocks:

- **Box-GCN** (D_b): predicts the object position p, orientation a, and size s from the new node features and the global scene embedding.
- **Shape-GCN** (D_s): predicts a shape code conditioned on the layout predicted by Box-GCN D_b, the new node features, and the global scene embedding. The resulting new object latent codes are used to query a 3D asset database to retrieve semantically and geometrically consistent models, which are then instantiated in the scene with the predicted poses and scales.

3.3 Training Objectives

At every incremental step, gradients are propagated *only* through the newly inserted objects. For the N new objects, the following loss is minimised:

$$L_{\text{layout}} = \frac{1}{N}\sum_{i=1}^{N}\Big(\|\hat{\mathbf{b}}_{\alpha,i} - \mathbf{b}_{\alpha,i}\|_1 \ + \ \|\hat{\alpha}_i - \alpha_i\|_1\Big), \tag{4}$$

$$L_{\text{shape}} = \frac{1}{N}\sum_{i=1}^{N}\|\hat{\mathbf{f}}_i^s - \mathbf{f}_i^s\|_1, \tag{5}$$

$$L_{\text{total}} = \lambda_1 L_{\text{layout}} \ + \ \lambda_2 L_{\text{shape}}. \tag{6}$$

L_{total} consists of two parts: L_{layout} and L_{shape}. λ_1 and λ_2 are weighting factors. L_{layout} reconstructs the 3D bounding-box parameters. Here $\mathbf{b}_{\alpha,i} \in \mathbb{R}^6$ denotes the box center position and size, α_i the yaw angle, and N the total number of boxes; hats indicate ground-truth values. L_{shape} reconstructs the object shape in feature space based on semantic label, object size, and relationship of 3D scene graph. Here, f_i is the feature of the object.

Furthermore, the curriculum learning strategy organizes the training scene samples from easy to hard, where the level of difficulty is defined based on the number of incremental steps across batch.

4 Experimental Setup

This section outlines the approach used to implement the evaluation methodology for *Incremental3D*, starting with generating the incremental 3D scene dataset to enable training and testing of the proposed approach.

4.1 Incremental Dataset

The incremental 3D scene sequences are constructed using the SG–FRONT dataset Zhai et al. (2024), which is an extension of the refined 3D–FRONT dataset Fu et al. (2021) where each room is annotated with a complete 3D scene graph. The dataset covers four categories of scenes: 4,041 *bedroom*, 813 *living room*, 900 *dining room*, and *all*. The *all* category is composed of a mixed subset of scenes sampled from the other three categories. The training, validation, and test splits follow the same partitioning as the original dataset Zhai et al. (2024).

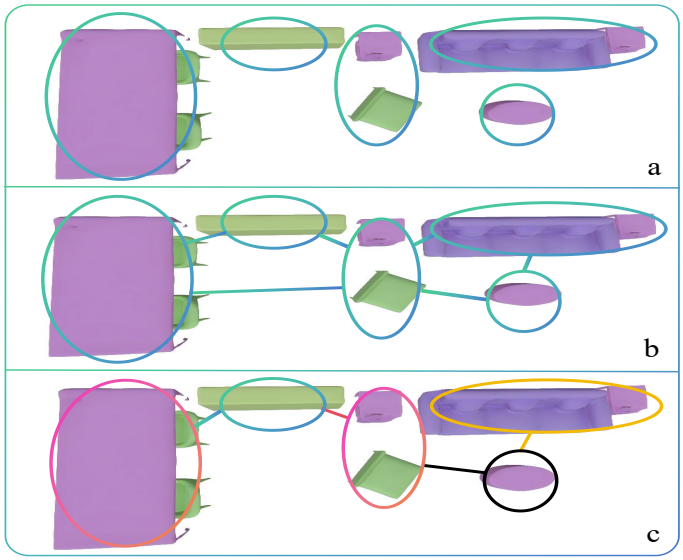


Figure 3: Pipeline of incremental dataset creation. The process includes clustering, graph construction, and insertion order computation, as illustrated in subfigures (a–c).

The pipeline for incremental dataset creation is illustrated in Figure 3. It consists of three main steps: 1. Given a room, the objects are clustered using DBSCAN Khan et al. (2014). 2. Each cluster is treated as a node, with edges connecting spatially adjacent clusters to form a graph. 3. The cluster farthest from the global centroid is chosen as the entry point. A visiting order over the remaining clusters is then determined by a nearest-neighbour greedy tour followed by a single 2–opt refinement, producing a near-optimal Hamiltonian path Sipser (1996) in $O(N^2)$ time.

Traversing this ordered list yields an incremental sequence of steps and at each step, the current cluster is regarded as the *new-node* set. The edge of the 3D scene-graph, whose subject and object both lie within the same cluster, and the directly connected cluster to it are preserved, while all others are discarded. Our processed relations are sparse, better reflecting real-world scene graph construction Wu et al. (2021); Hughes et al. (2022); Maggio et al. (2024); Deng et al. (2024), in cases where computational resources are limited and camera fields-of-view are restricted, preventing dense graphs. For subsequent training, the indices of the retained relationships are re-mapped. The resulting sequences cover the entire room without spatial jumps or node revisits.

4.2 Compared Baselines

CommonScenes. Current graph-based 3D scene generation methods Dhamo et al. (2021); Zhou et al. (2019); Zhai et al. (2023; 2024); Yang et al. (2025); Chattopadhyay et al. (2023) regenerate the entire scene at each step. Among them, *CommonScenes* Zhai et al. (2023) is a representative state-of-the-art approach and serves as our primary comparison baseline.

Baseline 1. Baseline 1 isolates the effect of *full regeneration*, in which the entire 3D scene is reconstructed from scratch at every incremental step. To ensure a fair comparison, we remove the main source of latency in *CommonScenes*—its online 3D shape generation—and replace it with a shape-retrieval pipeline. Furthermore, while *CommonScenes* employs diffusion- and VAE-based models to encourage scene diversity, Baseline 1 substitutes these components with an autoencoder to produce reconstructions that more closely match the ground truth. Baseline 1 retains the same *full-regeneration* paradigm: whenever new nodes are inserted, *all* existing nodes are re-predicted, including their positions, sizes, orientations, and shape features.

Baseline 2. Baseline 2 evaluates the importance of the CLS node and its associated global embedding. At each incremental step, newly inserted nodes generate objects based solely on their *local* neighborhood. In the

encoder, new nodes communicate only with directly connected old nodes, without receiving indirect or global contextual information. In the decoder, object generation for new nodes is conditioned solely on their own representations and those of their directly connected neighbors. As in Baseline 1, 3D shapes are obtained via retrieval rather than online generation. This baseline therefore represents a purely local incremental update mechanism that lacks global scene awareness.

4.3 Implementation Details

We conduct the training, evaluation, and visualization of *Incremental3D*, baseline 1, and baseline 2 on a single NVIDIA GeForce RTX 3070 GPU with 8 GB memory. We adopt the Adam optimizer with a learning rate of 1e-4 to train the network. λ_1 and λ_2 are set as 5 and 1 respectively. *CommonScenes* Zhai et al. (2023) is evaluated on an A6000 GPU, since loading CLIP features Radford et al. (2021) requires substantial memory.

4.4 Evaluation Metrics

Running time is used as the primary evaluation metric for assessing the speed of 3D scene generation, since a high update frequency is critical for user experience in immersive teleoperation. At 10Hz, interactive control exhibits noticeable lag, leading to user disorientation and imprecise robot manipulation Zhang (2020); Wang et al. (2023). In contrast, at 30Hz, users can perform tasks such as grasping and navigation for telerobotics Tefera et al. (2024); Pop et al. (2022).

An accurate environment model is essential for enabling robots to interact effectively with the physical world Parosi et al. (2023). Therefore, we prioritize accuracy as another key evaluation criterion, covering both layout accuracy and shape accuracy.

Layout accuracy measures the geometric errors between predicted and ground-truth layouts, including position difference E_{pos}, size difference E_{size}, and angle difference E_{ang}.

$$E_{\text{pos}} = \frac{1}{N}\sum_{i=1}^{N}\left\|p_{\text{pre}}^{(i)} - p_g^{(i)}\right\|_2 \tag{7}$$

$$E_{\text{size}} = \frac{1}{N}\sum_{i=1}^{N}\left\|s_{\text{pre}}^{(i)} - s_g^{(i)}\right\|_2 \tag{8}$$

$$E_{\text{ang}} = \frac{1}{N}\sum_{i=1}^{N}|a_{\text{pre}}^{(i)} - a_g^{(i)}| \tag{9}$$

Shape accuracy measures the similarity between the generated object mesh and the ground truth mesh. It is assessed using two metrics that evaluate the quality of the generated objects. Chamfer Distance (CD) Bakshi et al. (2023) measures the average nearest-neighbor distance between two point-clouds and reflects the global geometric alignment, as defined below:

$$\text{CD} = \frac{1}{N}\sum_{i=1}^{N}\min_j \|p_i - g_j\|_2^2 + \frac{1}{N}\sum_{j=1}^{N}\min_i \|g_j - p_i\|_2^2 \tag{10}$$

F-score Katz et al. (2007) evaluates the degree of local geometric alignment between two meshes by combining precision P and recall R, where d_1 and d_2 denote the nearest distances from predicted points to ground truth points and vice versa—a match is counted if the distance is below a threshold t. In our experiments, we set $t = 0.1$ meters.

$$F = \frac{2PR}{P+R}, \tag{11}$$

$$P = \frac{\#\{d_1 < t\}}{N}, \quad R = \frac{\#\{d_2 < t\}}{N} \tag{12}$$

5 Results and Analysis

5.1 Running time

Table 1 reports the running time (from 3D scene graph input to 3D mesh output) of the baselines and *Incremental3D*. Even with such high-end hardware, *CommonScenes* still takes more than 20 seconds to infer a single whole 3D scene from a scene graph and it is not practical for real-time robot tasks. Baseline 1 regenerates all objects at each node insertion, limiting its update rate to 10.02 Hz—an insufficient frequency that causes latency and user discomfort in immersive environments. Baseline 2 can achieve real-time generation of 3D scenes, but its scene accuracy remains unsatisfactory (see layout and shape accuracy evaluation). *Incremental3D* achieves a generation frequency of 38Hz on average in different scenarios, enabling smooth interaction.

Method	**Bedroom**	**Living room**	**Dining room**	**All**
CommonScenes	24.676	21.390	24.950	28.633
Baseline 1	0.079	0.114	0.116	0.090
Baseline 2	**0.021**	**0.028**	**0.027**	**0.024**
Incremental3D	0.022	0.029	0.028	0.026

Table 1: Comparison of methods on running time and the unit is seconds.

Method	**Bedroom**			**Living room**			**Dining room**			**All**		
	pos	size	angle	pos	size	angle	pos	size	angle	pos	size	angle
CommonScenes	25.4	25.9	10.0	36.8	39.4	15.0	34.1	56.1	12.8	52.2	52.2	17.7
Baseline 1	4.8	4.6	3.4	15.6	9.0	4.3	14.1	7.1	7.8	3.3	3.1	2.7
Baseline 2	3.9	2.9	3.3	7.9	7.6	4.9	24.0	10.7	4.4	3.9	3.7	2.9
Incremental3D	**1.9**	**1.6**	**1.9**	**4.8**	**2.5**	**1.4**	**5.2**	**3.9**	**1.9**	**2.3**	**1.6**	**1.6**

Table 2: Comparison of methods on layout accuracy evaluation, including position, size, and angle error between generated objects and ground truth (*lower is better*). The unit for position, size and angle is cm, cm, and degree, respectively.

Method	**Bedroom**		**Living room**		**Dining room**		**All**	
	CD	F-score	CD	F-score	CD	F-score	CD	F-score
CommonScenes	0.180	0.135	0.104	0.274	0.136	0.199	0.145	0.238
Baseline 1	0.091	0.443	**0.077**	0.503	0.076	0.506	0.080	0.485
Baseline 2	0.0547	0.582	0.094	0.497	0.086	0.493	0.079	0.540
Incremental3D	**0.0502**	**0.615**	0.080	**0.509**	**0.075**	**0.507**	**0.068**	**0.549**

Table 3: Comparison of methods on shape accuracy evaluation using Chamfer Distance (*lower is better*) and F-score (*higher is better*).

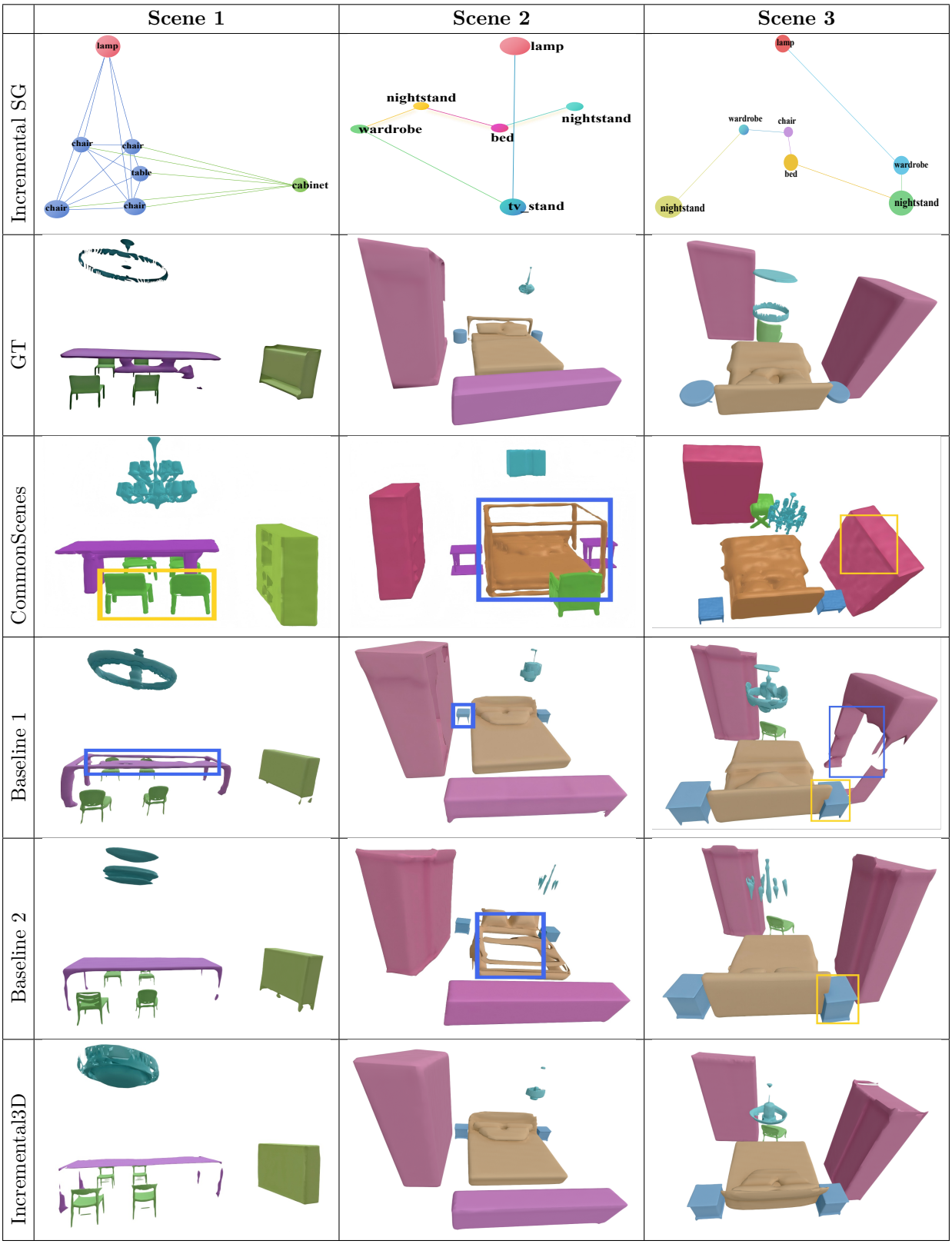


Figure 4: Qualitative comparison across three scenes. Each cell shows one reconstructed 3D scene. Columns correspond to different scenes and rows to different methods. Red boxes highlight layout errors, and blue boxes indicate shape inaccuracies relative to the ground truth.

5.2 Layout accuracy

Table 2 summarizes the layout accuracy of different methods. *CommonScenes* shows the lowest accuracy, with average errors of 37.16 cm in position, 43.4 cm in size, and 13.88° in orientation. Baseline 1 achieves errors of 9.45 cm (position), 5.95 cm (size), and 4.55° (orientation). Baseline 2 reports 9.93 cm, 6.63 cm, and 3.88°, respectively. *Incremental3D* attains the highest accuracy, with average errors of only 3.45 cm in position, 2.40 cm in size, and 1.70° in orientation.

Figure 4 provides qualitative layout comparison results. For *CommonScenes*, for example, the chair highlighted in yellow in scene 1 is placed on the right side of the table, whereas the chairs in the ground truth are located on the left. In the third scene, the nightstand (yellow rectangle) collides with the bed in baseline 1 and baseline 2. In contrast, the layouts produced by *Incremental3D* are closest to the ground truth.

Although CommonScenes generates reasonable layouts overall, it exhibits systematic deviations from the ground truth in size, position, and orientation. Its VAE is trained by maximizing the evidence lower bound (ELBO), which encourages distributionally plausible generations rather than precise reconstruction of a specific scene. Baseline 1 and Baseline 2 aim to reconstruct original environments from scene graphs and achieve higher layout accuracy. However, Baseline 1 generates new objects only from local information: despite recomputing node features for both new and existing nodes at each insertion, shallow GNNs are inherently limited in capturing information from indirectly connected nodes Di Giovanni et al. (2024); Giraldo et al. (2023). For instance, a 3-layer GNN can propagate information at most within a 3-hop neighborhood. Such one-shot methods are difficult to obtain the global scene embedding, especially in large-scale environments. Baseline 2 restricts new nodes to communicate only with directly connected neighbors, and again generates objects solely from local context. In contrast, *Incremental3D* conditions generation on global scene information through the CLS node. This global embedding aggregates cues such as room symmetry, negative space, collision constraints, and style, enabling more accurate and coherent 3D layout generation compared to the baselines.

5.3 Shape accuracy

Table 3 reports the quantitative comparison results for shape accuracy. On average, shapes generated by *CommonScenes* show the poorest global and local geometric alignment with the ground truth, exhibiting the largest mean Chamfer Distance (0.123 m) and the lowest mean F-score (0.21). Baseline 1 achieves a Chamfer Distance of 0.081 m and an F-score of 0.48 across four scenarios, while baseline 2 reports 0.078 m and 0.52, respectively. *Incremental3D* achieves the best global and local geometry alignment with the ground truth, reaching a Chamfer Distance of 0.068 m and an F-score of 0.55.

Figure 4 provides qualitative comparisons of object shapes. In Scene 2, *CommonScenes* produces a noticeably different bed compared to the ground truth. For baseline 1, the table in scene 1 and the wardrobe in scene 3 are clearly misaligned with the reference meshes. Similarly, for baseline 2, the bed in scene 2 does not match the ground truth. In contrast, *Incremental3D* generates object meshes that are closest to the ground truth.

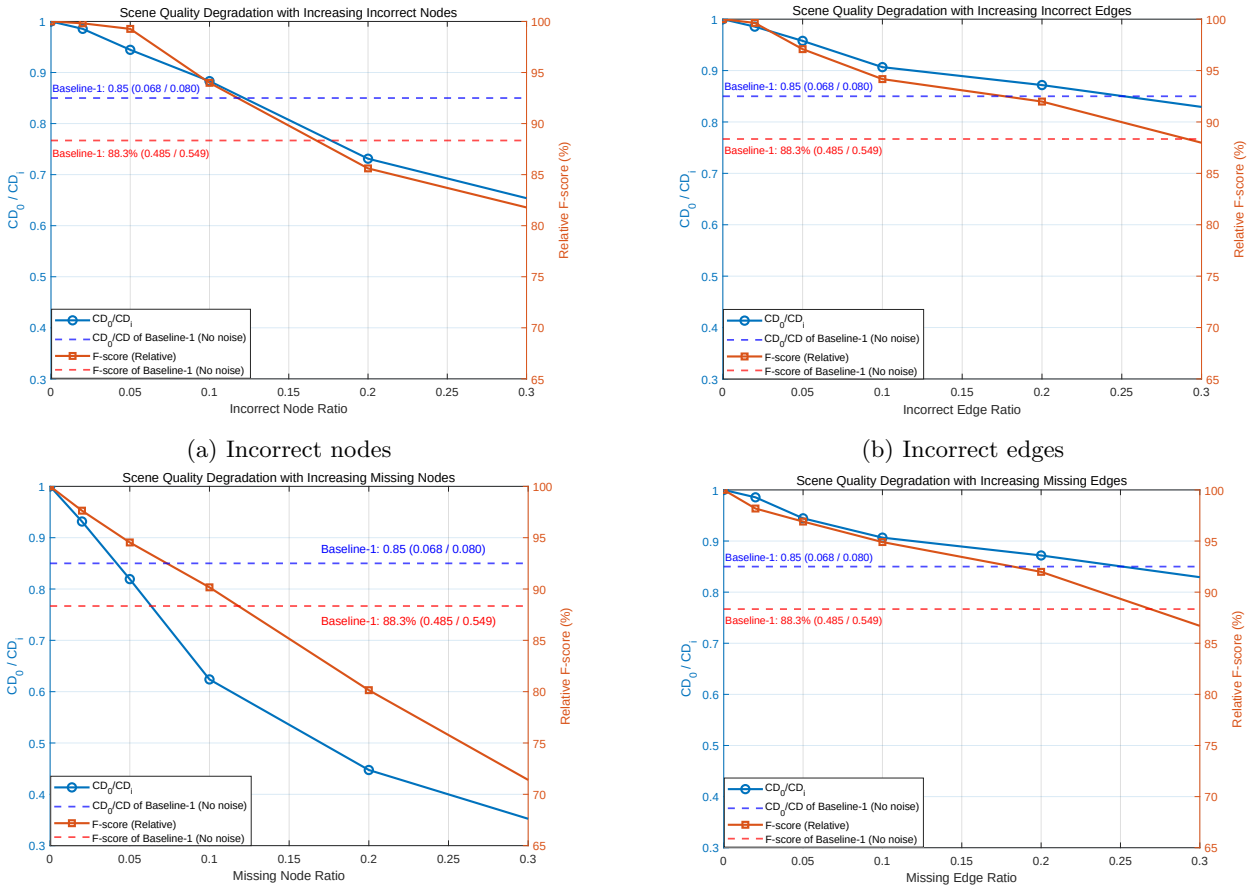
CommonScenes generates plausible and diverse shapes by explicitly modeling a distribution of object geometries conditioned on the scene graph. Since shapes are sampled through a diffusion process in a VQ-VAE latent space, different random seeds or sampling trajectories yield diverse yet valid meshes rather than an exact ground-truth replica. Baseline 1 and baseline 2 achieve better geometric alignment than *CommonScenes*, but the shapes of newly inserted objects rely solely on local information (see layout accuracy section). By comparison, *Incremental3D* leverages the CLS node to inject room-level priors into each new object's conditional distribution, thereby reducing uncertainty in shape prediction and avoiding information decay during deep propagation. As a result, *Incremental3D* converges more effectively to shapes that align with the ground truth.

5.4 Robustness Evaluation

In open-world environments, accurate 3D scene graph construction remains an unsolved problem. Small objects are often missed, and objects under severe occlusion are frequently misclassified. To characterize

Category	Running time	Position	Size	Angle	CD	F-score
Bedroom	0.024	4.3	4.8	3.8	0.067	0.559
Living room	0.031	7.5	5.1	5.7	0.073	0.496
Dining room	0.032	9.4	6.2	4.4	0.071	0.507
All	0.025	5.4	4.6	3.9	0.087	0.530

Table 4: Performance of ablation experiment without curriculum learning across different room categories.



(a) Incorrect nodes

(b) Incorrect edges

(c) Missing nodes

(d) Missing edges

Figure 5: Robustness evaluation under 4 different types of scene graph noise. The horizontal axis denotes the noise level, and the vertical axis represents the global and local geometric similarity between the generated objects and the ground-truth meshes. Incorrect or missing edges have a minor impact on overall scene quality, whereas incorrect node labels lead to a more noticeable degradation. Missing nodes cause more severe performance drops. Under low-noise conditions, Incremental3D achieves scene quality comparable to noise-free Baseline–1.

the operational boundaries of the proposed method, we conduct a series of robustness experiments under noisy 3D scene graph inputs. The level of perception noise varies with the environment. For example, CLIO Maggio et al. (2024) reports a node label accuracy of approximately 98% in apartment environments, while performance degrades to around 73% node precision in office environments and 79% in building-scale environments. To systematically evaluate robustness, we inject synthetic noise into the input 3D scene graphs during data loading. Specifically, we apply noise levels of 2%, 5%, 10%, 20%, and 30% across four settings that simulate common perception errors: incorrect object labels, incorrect edges, missing edges, and

missing nodes. For more details about the noise injection methods, refer to the appendix A.2. All robustness evaluations are conducted on the *all* split of the incremental 3D–FRONT dataset, which includes a mixture of living rooms, bedrooms, and dining rooms.

Chamfer Distance (CD) and F-score are used to evaluate global geometry similarity and local geometry similarity between the generated shape and the ground truth shape. The original numerical results for different noise levels are provided in the appendix A.2. A larger CD indicates worse scene quality. To visualize the degradation of global scene consistency as noise increases, we report a normalized metric $\mathrm{CD}_0/\mathrm{CD}_i$, where CD_0 denotes noise-free CD, and i iterates over the 5 noise levels. To measure how local geometry similarity degrades with noise, we use $\text{F-score}_i/\text{F-score}_0$ to illustrate the degradation of perception, where F-score_0 is the noise-free F-score. For missing-node scenarios, the absent objects are represented by a single point when computing the CD against the ground-truth mesh, and the corresponding F-score is set to zero.

Figure 5 summarizes the results under different noise conditions. We also report the performance of Baseline–1 under noise-free inputs for reference. The results show that incorrect or missing edges have only a minor impact on overall scene quality, whereas incorrect node labels lead to more noticeable degradation. For example, with 30% incorrect node labels, the local geometry similarity drops by 18.2%. Missing nodes cause a more severe degradation, indicating substantial loss of structural information; for instance, when 30% of nodes are missing, the local geometry similarity decreases by 28.6%. Nevertheless, under low-noise conditions typical of well-organized indoor environments, the proposed method remains robust. For example, with 10% incorrect node labels, *Incremental3D* achieves scene quality comparable to Baseline–1 under noise-free inputs. Similarly, when 5% of nodes are missing, the resulting scene quality remains close to that of Baseline–1 with clean scene graphs.

5.5 Ablations

Baseline 2 can be regarded as an ablation experiment of CLS node, where the CLS node is removed and new nodes communicate only with their directly connected neighborhood at each incremental step. Results in Table 1, 2, and 3 have demonstrated the importance of the CLS node for incremental scene generation.

To further evaluate the effectiveness of curriculum learning, this ablation experiment retains the CLS node for global context aggregation but is trained without applying the curriculum learning strategy. In this setting, the number of steps is randomized across different batch scenes. Table 4 summarizes its performance in terms of running time and scene accuracy. The average running time across all categories is 0.028 seconds. Compared with *Incremental3D*, layout accuracy degrades on average from (3.45 cm, 2.40 cm, 1.70°) in position, size, and angle errors to (6.65 cm, 5.18 cm, 4.45°), while shape accuracy decreases from (0.068, 0.55) in Chamfer Distance and F-score to (0.0745, 0.52).

6 Limitation and Future Work

While *Incremental3D* demonstrates competitive performance on simulated incremental sequences, its evaluation is limited by the absence of publicly available incremental real-world 3D scene datasets. Our experiments are conducted on synthetic data derived from SG-FRONT Zhai et al. (2024), which does not fully capture the sensor noise and perception errors typically present in real robot teleoperation systems.

Moreover, in open-world environments, accurate scene graph construction remains an unsolved problem, particularly in the presence of small objects and severe occlusions, which often lead to missing or incorrect object labels. Consequently, the proposed system is primarily intended for well-organized indoor environments, where object layouts are relatively structured and scene graphs can be constructed with reasonable reliability. Furthermore, to achieve real-time performance, *Incremental3D* adopts a database-shape-retrieval-based strategy rather than direct shape synthesis, which inevitably limits its open-set generalization ability.

In future work, we plan to construct an incremental 3D scene dataset including RGB-D scans, 3D scene graphs, 3D models of 3D scenes, and implement the completed immersive teleoperation pipeline on this dataset that integrates: (1) robot-side incremental scene-graph construction, (2) text-based scene-graph

encoding and streaming, and (3) operator-side incremental graph-based decoding and 3D scene generation. Ultimately, we aim to deploy *Incremental3D* on real robot teleoperation tasks, conducting user studies to understand its relative effect on teleoperation task performance.

7 Conclusions

In this work, we introduce *Incremental3D*, the first framework that adapts scene-graph-based generation to the incremental and latency-sensitive nature of immersive robotic teleoperation. By incorporating a global CLS node and updating only newly added objects, *Incremental3D* achieves real-time 38 Hz scene synthesis, while maintaining high spatial and geometric accuracy. Our results demonstrate that incremental graph-to-3D generation is both feasible and effective, opening new possibilities for telepresence, telerobotics, remote collaboration, and other real-time 3D applications.

References

Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5664–5673, 2019.

Ainesh Bakshi, Piotr Indyk, Rajesh Jayaram, Sandeep Silwal, and Erik Waingarten. Near-linear time algorithm for the chamfer distance. *Advances in Neural Information Processing Systems*, 36:66833–66844, 2023.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Vivian Chai and Dan-Sorin Necsulescu. Time delay investigation in telerobotic surgery. In *ICINCO*, pp. 375–382, 2021.

Aditya Chattopadhyay, Xi Zhang, David Paul Wipf, Himanshu Arora, and René Vidal. Learning graph variational autoencoders with constraints and structured priors for conditional indoor 3d scene generation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 785–794, 2023.

Kourosh Darvish, Luigi Penco, Joao Ramos, Rafael Cisneros, Jerry Pratt, Eiichi Yoshida, Serena Ivaldi, and Daniele Pucci. Teleoperation of humanoid robots: A survey. *IEEE Transactions on Robotics*, 39(3): 1706–1727, 2023.

Yinan Deng, Jiahui Wang, Jingyu Zhao, Xinyu Tian, Guangyan Chen, Yi Yang, and Yufeng Yue. Opengraph: Open-vocabulary hierarchical 3d graph representation in large-scale outdoor environments. *IEEE Robotics and Automation Letters*, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Helisa Dhamo, Fabian Manhardt, Nassir Navab, and Federico Tombari. Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16352–16361, 2021.

Francesco Di Giovanni, T Konstantin Rusch, Michael M Bronstein, Andreea Deac, Marc Lackenby, Siddhartha Mishra, and Petar Veličković. How does over-squashing affect the power of gnns? *Transactions on Machine Learning Research*, 2024.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Parinaz Farajiparvar, Hao Ying, and Abhilash Pandya. A brief survey of telerobotic time delay mitigation. *Frontiers in Robotics and AI*, 7:578805, 2020.

Mingtao Feng, Chenbo Yan, Zijie Wu, Weisheng Dong, Yaonan Wang, and Ajmal Mian. History-enhanced 3d scene graph reasoning from rgb-d sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10933–10942, 2021.

Paul Gay, James Stuart, and Alessio Del Bue. Visual graphs from motion (vgfm): Scene understanding with object geometry reasoning. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pp. 330–346. Springer, 2019.

Jhony H Giraldo, Konstantinos Skianis, Thierry Bouwmans, and Fragkiskos D Malliaros. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 566–576, 2023.

N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. In *Robotics: Science and Systems (RSS)*, 2022.

Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. In *ACM SIGGRAPH 2007 papers*, pp. 24–es. 2007.

Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pp. 232–238. IEEE, 2014.

Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pp. 1945–1954. PMLR, 2017.

Pengzhi Li, Yan Pei, and Jianqiang Li. A comprehensive survey on design and application of autoencoder in deep learning. *Applied Soft Computing*, 138:110176, 2023.

Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. Clio: Real-time task-driven open-set 3d scene graphs. *IEEE Robotics and Automation Letters*, 2024.

Abdeldjallil Naceri, Dario Mazzanti, Joao Bimbo, Yonas T Tefera, Domenico Prattichizzo, Darwin G Caldwell, Leonardo S Mattos, and Nikhil Deshpande. The vicarios virtual reality interface for remote robotic teleoperation: Teleporting for intuitive tele-manipulation. *Journal of Intelligent & Robotic Systems*, 101 (4):80, 2021.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

Riccardo Parosi, Mattia Risiglione, Darwin G Caldwell, Claudio Semini, and Victor Barasuol. Kinematically-decoupled impedance control for fast object visual servoing and grasping on quadruped manipulators. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8. IEEE, 2023.

Bogdan Pop, Cristian Moldovan, Florin Florescu, and Ioan Emil Popescu. Study of latency for a teleoperation system with video feedback using wi-fi and 4g-lte networks. In *2022 International Conference on Communications, Information, Electronic and Energy Systems (CIEES)*, pp. 1–4. IEEE, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Marian Renz, Felix Igelbrink, and Martin Atzmueller. Integrating prior observations for incremental 3d scene graph prediction. *arXiv preprint arXiv:2509.11895*, 2025.

Nazmus Sakib, Kenneth C Gahan, and Craig A Woolsey. Time delay mitigation in aerial telerobotic operations using heterogeneous stereo-vision systems. *Journal of Aerospace Information Systems*, 20(9): 526–545, 2023.

Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

YT Tefera, I Sarakoglou, SN Deore, Y Kim, V Barasuol, M Villa, S Anastasi, DG Caldwell, N Tsagarakis, C Semini, et al. Robot teleoperativo: Collaborative cybernetic systems for immersive remote teleoperation. In *2024 IEEE Conference on Telepresence*, pp. 1–4. IEEE, 2024.

Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Ao Wang, Fengyuan Sun, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. [cls] token tells everything needed for training-free efficient mllms. *arXiv preprint arXiv:2412.05819*, 2024.

Jialin Wang, Rongkai Shi, Wenxuan Zheng, Weijie Xie, Dominic Kao, and Hai-Ning Liang. Effect of frame rate on user experience, performance, and simulator sickness in virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2478–2488, 2023.

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576, 2021.

Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7515–7525, 2021.

Shun-Cheng Wu, Keisuke Tateno, Nassir Navab, and Federico Tombari. Incremental 3d semantic scene graph prediction from rgb sequences. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5064–5074, 2023.

Zhifei Yang, Keyang Lu, Chao Zhang, Jiaxing Qi, Hanqi Jiang, Ruifei Ma, Shenglin Yin, Yifan Xu, Mingzhe Xing, Zhen Xiao, et al. Mmgdreamer: Mixed-modality graph for geometry-controllable 3d indoor scene generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 9391–9399, 2025.

Guangyao Zhai, Evin Pınar Örnek, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. Commonscenes: Generating commonsense 3d indoor scenes with scene graph diffusion. *Advances in Neural Information Processing Systems*, 36:30026–30038, 2023.

Guangyao Zhai, Evin Pınar Örnek, Dave Zhenyu Chen, Ruotong Liao, Yan Di, Nassir Navab, Federico Tombari, and Benjamin Busam. Echoscene: Indoor scene generation via information echo over scene graph diffusion. In *European Conference on Computer Vision*, pp. 167–184. Springer, 2024.

Chen Zhang. Investigation on motion sickness in virtual reality environment from the perspective of user experience. In *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pp. 393–396. IEEE, 2020.

Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. Poolingformer: Long document modeling with pooling attention. In *International Conference on Machine Learning*, pp. 12437–12446. PMLR, 2021.

Yang Zhou, Zachary While, and Evangelos Kalogerakis. Scenegraphnet: Neural message passing for 3d indoor scene augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7384–7392, 2019.

A Appendix

A.1 Incremental 3D scene graph construction pipeline

Algorithm 1: Incremental 3D scene graph construction based on 3D perception

Input : RGB-D frames $\{(I_k, D_k)\}_{k=1}^{K}$, camera poses $\{T_{w\leftarrow c}^{k}\}_{k=1}^{K}$
Output: Incremental 3D scene graph sequence $\{G_k\}_{k=1}^{K}$

Initialize global scene graph $G_0 \leftarrow (\mathcal{V}_0, \mathcal{E}_0)$
for $k \leftarrow 1$ **to** K **do**
  `// 1) Per-frame segmentation`
  $\mathcal{S}_k \leftarrow \mathrm{Segment}(I_k, D_k)$
  `// 2) Extract per-object features`
  foreach $s \in \mathcal{S}_k$ **do**
    $F_s \leftarrow \mathrm{ExtractFeat}(s, T_k, D_k)$ `// pose, size, color, embedding, etc`
  `// 3) Search for Neighbor Graph of current frame`
  $G_n \leftarrow \mathrm{SearchNeighbors}(T_k)$
  `// 4) Predict the scene graph of the current frame using a GNN based on the neighbor graph and node feature`
  $(\mathcal{V}_k, \mathcal{E}_k) \leftarrow \mathrm{GNN}(G_n, F_s)$
  `// 5) Fuse current subgraph into the global scene graph`
  $G_k \leftarrow \mathrm{FuseSubgraph}(G_{k-1}, \mathcal{V}_k, \mathcal{E}_k)$
return $\{G_k\}_{k=1}^{K}$

Algorithm 2: Incremental scene graph construction based on 2D perception

Input : RGB-D frames $\{(I_k, D_k)\}_{k=1}^{K}$
Output: Incremental scene graph sequence $\{G_k\}_{k=1}^{K}$

Initialize global scene graph $G_0 \leftarrow (\mathcal{V}_0, \mathcal{E}_0)$
for $k \leftarrow 1$ **to** K **do**
  `// 1) 2D instance segmentation (e.g., SAM)`
  $\mathcal{S}_k \leftarrow \mathrm{Segment}(I_k)$
  `// 2) Camera pose estimation`
  $P_k \leftarrow \mathrm{EstimatePose}(I_k, D_k)$
  `// 3) Per-instance labeling and 3D attribute estimation`
  foreach $s \in \mathcal{S}_k$ **do**
    `// Semantic labeling (e.g., CLIP)`
    $\ell_s \leftarrow \mathrm{Classify}(s)$
    `// Estimate position` p`, orientation` R`, and size` d `of objects`
    $(p_s, R_s, d_s) \leftarrow \mathrm{Estimate3D}(\ell_s, P_k, I_k, D_k)$
    $v_s \leftarrow \mathrm{CreateNode}(\ell_s, p_s, R_s, d_s)$
    $\mathcal{V}_k \leftarrow \mathcal{V}_{k-1} \cup \{v_s\}$
  `// 4) Rule-based relationship inference for the current frame based on layout, etc`
  foreach $(v_i, v_j) \in \mathcal{V}_k \times \mathcal{V}_k,\ i \neq j$ **do**
    $r_{ij} \leftarrow \mathrm{InferRules}(v_i, v_j)$ `// left/right, front/behind, on, near, ...`
    if $r_{ij} \neq \varnothing$ **then**
      $\mathcal{E}_k \leftarrow \mathcal{E}_k \cup \{(v_i, r_{ij}, v_j)\}$
  `// 5) Fuse current-frame subgraph into the global incremental graph`
  $G_k \leftarrow \mathrm{FuseSubgraph}(G_{k-1}, \mathcal{V}_k, \mathcal{E}_k)$
return $\{G_k\}_{k=1}^{K}$

A.2 Robustness evaluation

Algorithm 3: Injecting Random Incorrect Node Label Noise

Input : Correct node labels $\{c_i\}_{i=1}^{N}$;
noise rate p_{node}
Output: Corrupted node labels $\{\tilde{c}_i\}_{i=1}^{N}$

Init: $\tilde{c}_i \leftarrow c_i \;\; \forall i \in \{1, \ldots, N\}$

1. Determine number of corrupted nodes
$k \leftarrow \lfloor p_{\text{node}} \cdot N \rceil$

2. Randomly sample nodes to corrupt
Select subset $\mathcal{S} \subseteq \{1, \ldots, N\}$ with $|\mathcal{S}| = k$ uniformly

3. Apply random label replacement
foreach $i \in \mathcal{S}$ **do**
  $\tilde{c}_i \leftarrow \texttt{Sample}(\mathcal{C} \setminus \{c_i\})$

return $\{\tilde{c}_i\}_{i=1}^{N}$

Algorithm 4: Injecting Random Wrong Edge Label Noise

Input : Relation triples $\mathcal{T} = \{(s_j, p_j, o_j)\}_{j=1}^{E}$;
noise rate p_{edge}
Output: Corrupted relation triples $\tilde{\mathcal{T}}$

Init: $\tilde{\mathcal{T}} \leftarrow \mathcal{T}$

1. Determine number of corrupted edges
$k \leftarrow \lfloor p_{\text{edge}} \cdot E \rceil$

2. Randomly sample edges to corrupt
Select subset $\mathcal{E} \subseteq \{1, \ldots, E\}$ with $|\mathcal{E}| = k$ uniformly

3. Apply random predicate replacement
foreach $j \in \mathcal{E}$ **do**
  $(s_j, p_j, o_j) \leftarrow \tilde{\mathcal{T}}[j]$
  $\tilde{p}_j \leftarrow \texttt{Sample}(\mathcal{P} \setminus \{p_j\})$
  $\tilde{\mathcal{T}}[j] \leftarrow (s_j, \tilde{p}_j, o_j)$

return $\tilde{\mathcal{T}}$

Table 5: Robustness evaluation under different scene graph noise types.

Noise (%)	Wrong Nodes		Wrong Edges		Missing Nodes		Missing Edges	
	CD ↓	F-score ↑	CD ↓	F-score ↑	CD ↓	F-score ↑	CD ↓	F-score ↑
0	0.068	0.549	0.068	0.549	0.068	0.549	0.068	0.549
2	0.069	0.548	0.069	0.547	0.073	0.536	0.069	0.539
5	0.072	0.545	0.071	0.533	0.083	0.519	0.072	0.532
10	0.077	0.516	0.075	0.517	0.109	0.495	0.075	0.521
20	0.093	0.470	0.078	0.505	0.152	0.440	0.078	0.505
30	0.104	0.449	0.082	0.483	0.193	0.392	0.082	0.476

This appendix provides additional details on the noise injection strategies applied to 3D scene graphs, along with the corresponding experimental results for the robustness evaluation.

We inject controlled noise into the input scene graphs on a per-scene basis during data loading. For each scene, when reading node labels, a predefined proportion of correct object category IDs is randomly replaced with alternative category IDs (see Algorithm 3). When reading relation triples, a predefined proportion of correct predicate IDs is randomly replaced with alternative relation IDs (see Algorithm 4). To simulate missing-edge noise, we randomly select and remove a specified fraction of edges from the incremental scene graph. To simulate missing-node noise, we randomly remove a specified fraction of nodes and simultaneously delete all edges incident to the removed nodes.