This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2025.3616751

SGSG: Stroke-Guided Scene Graph Generation

Qixiang Ma, Runze Fan, Lizhi Zhao, Jian Wu, Sio-Kei Im, and Lili Wang

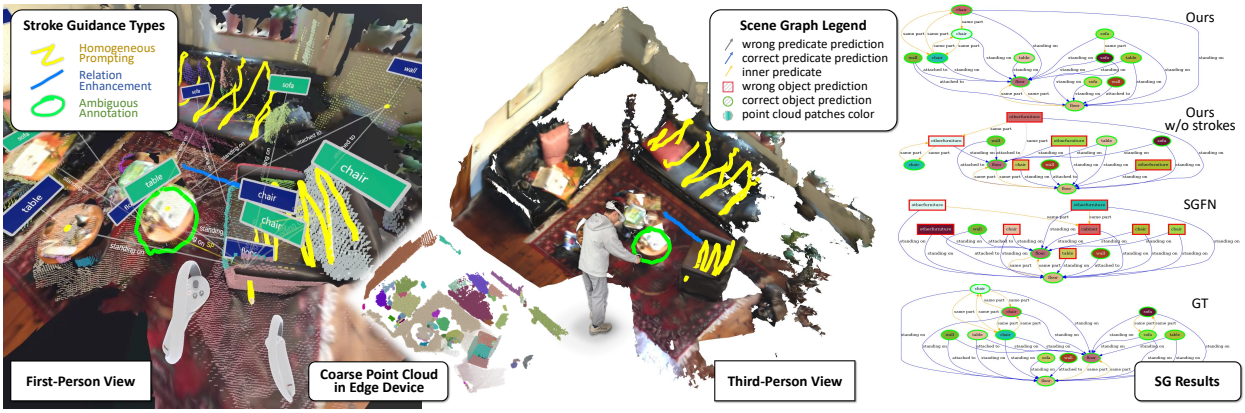


Fig. 1: SGSG in action: In the XR first-person view, using controllers, the user can inspect the scene graph (nodes and edges) and point clouds anchored in physical space, and draw strokes to refine semantic predictions. Updated semantics are shown in green, while correctly inferred inner predicates *"same part" (SP)* are shown in yellow. The third-person view demonstrates three stroke guidance types: binding nodes, modifying edges, and hinting nodes. Based on coarse point cloud patch inputs, SGSG unifies fragmented object semantics, corrects ambiguities, and improves predicate prediction. User guidance generalizes beyond annotated regions, addressing similar errors (e.g., misclassified *"sofa"*).

Abstract—3D scene graph generation is essential for spatial computing in Extended Reality (XR), providing structured semantics for task planning and intelligent perception. However, unlike instance-segmentation-driven setups, generating semantic scene graphs still suffer from limited accuracy due to coarse and noisy point cloud data typically acquired in practice, and from the lack of interactive strategies to incorporate users' spatialized and intuitive guidance. We identify three key challenges: designing controllable interaction forms, involving guidance in inference, and generalizing from local corrections. To address these, we propose SGSG, a Stroke-Guided Scene Graph generation method that enables users to interactively refine 3D semantic relationships and improve predictions in real time. We propose three types of strokes and a lightweight SGstrokes dataset tailored for this modality. Our model integrates stroke guidance representation and injection for spatio-temporal feature learning and reasoning correction, along with intervention losses that combine consistency-repulsive and geometry-sensitive constraints to enhance accuracy and generalization. Experiments and the user study show that SGSG outperforms state-of-the-art methods 3DSSG and SGFN in overall accuracy and precision, surpasses JointSSG in predicate-level metrics, and reduces task load across all control conditions, establishing SGSG as a new benchmark for interactive 3D scene graph generation and semantic understanding in XR. Implementation resources are available at: https://github.com/Sycamore-Ma/SGSG-runtime.

Index Terms—Extended Reality, Scene Graph Generation, Spatial Computing, User Interaction.

1 Introduction

Semantic scene graphs play a vital role in spatial computing and task planning within Extended Reality (XR) environments. By structurally representing the semantics of objects and their relationships, they support diverse applications such as embodied intelligence [20, 37], virtual-physical interaction [24, 33], and scene-based content generation [12, 44]. With advances in physical scene acquisition and processing, research has shifted from 2D to 3D scene graph generation to enable deeper semantic reasoning in real-world environments. This series of studies bridge computer graphics and vision, contributing significantly to high-level scene understanding. Most methods use point cloud encoders and message-passing backbones, typically requiring high-quality instance-segmented point clouds as input.

Due to sensing and resolution limitations of edge devices in practical scenarios, the acquired point clouds are often coarse, noisy, or incomplete. These deficiencies hinder the reliability and accuracy of scene graph generation, as the encoder is sensitive to object shape and neighborhood message-passing struggles with ambiguous, disconnected, or fragmented points that lack relational consistency. Some methods attempt to compensate by incorporating external knowledge or multimodal signals, but such priors are tailored to instance-segmented inputs or lack interactive strategies for refining predictions on degraded geometry. In contrast, XR naturally supports intuitive spatial interaction, enabling users to refine scene graph generation through direct, structured in-scene feedback with minimal cognitive overhead, and to express subjective or non-verbal semantics aligned with the 3D context. To incorporate high-level user guidance, three challenges remain: (1) designing controllable guidance that effectively influences prediction, (2) injecting it smoothly into inference without disrupting learning dynamics, and (3) generalizing local corrections to produce holistic improvements across the scene graph.

In this paper, we propose SGSG, an interactive Stroke-Guided Scene Graph generation approach addressing the above challenges. We design

- *Qixiang Ma, Runze Fan, Lizhi Zhao, Jian Wu, and Lili Wang are with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing, China, 100191. E-mail: {sycamore_ma,by2106131,lizhizhao,lanayawj,wanglily}@buaa.edu.cn.*
- *Sio-Kei Im is with Faculty of Applied Sciences, Macao Polytechnic University, Macao SAR, China, 999078. E-mail: marcusim@mpu.edu.mo*
- ✉ *Lili Wang is the corresponding author.*

an interaction paradigm with three distinct types of stroke guidance and a lightweight algorithm to generate the SGstrokes dataset, where each input is controllable and spatially grounded. Our model incorporates a guidance representation component to capture spatio-temporal intent and a guidance injection network to integrate user feedback into reasoning without disrupting inference. Alongside core loss terms, we introduce an intervention loss combining consistency-repulsive and geometry-sensitive constraints. In comparative experiments with state-of-the-art (SOTA) methods, SGSG outperforms 3DSSG [35] and SGFN [39] in overall accuracy and precision, including a 14.3% improvement in relationship tuple accuracy, and surpasses the multimodal method JointSSG [38] in predicate-level metrics. The user study shows that SGSG reduces task load as well as post-correction error. As shown in Fig. 1, it enables users to adjust object and predicate semantics via strokes at both local and global levels during spatial navigation over coarse-segmented point clouds. The immersive semantic scene graph can serve as augmented contextual information, supporting future XR applications such as intent-driven interaction and embodied intelligence. Our main contributions are summarized as follows:

- We propose an interactive 3D scene graph generation pipeline and construct a SGstrokes dataset.
- We propose a novel representation for stroke guidance combining temporal and geometric features.
- We propose a guidance injection network to improve semantic prediction via both local and global correction.
- We design a set of intervention loss functions integrating consistency-repulsive and geometric constraints.

2 Related Work

We review prior work on semantic scene graphs, covering foundational generation methods, prediction with structural cues, and applications in spatial computing.

2.1 Semantic Scene Graph Generation

Semantic scene graph generation, also known as scene graph prediction, aims to extract structured representations of objects and their relationships from visual scenes [2]. Significant progress has been made in 2D settings, focusing on network architecture optimization [26], external knowledge integration [47], and incorporating spatio-temporal features from video sequences [6]. With the availability of 3D datasets such as ScanNet [7] and 3RScan [34], scene graph generation has expanded beyond 2D. 3RScan provides structurally detailed instance and relationship annotations, establishing a benchmark for 3D scene graph generation tasks. Early work by Armeni et al. [1] proposed a hierarchical semantic scene representation, followed by the representative 3DSSG (also known as SGPN) [35], which combined PointNet [16] and graph neural networks (GNNs) [42] for joint learning of object and relationship features. Later improvements introduced cross-attention mechanisms between nodes and edges [45], and enhanced contextual relation sampling [27]. Koch et al. [22] adapted autoencoder-based scene reconstruction for scene graph generation, enhancing structural regression. However, these 3D methods rely on instance-level inputs and overlook external guidance, limiting their robustness on coarse-segmented or noisy point clouds, as seen in our target scenarios.

2.2 Scene Graph Prediction with Structural Cues

Beyond foundational scene graph generation, recent approaches incorporate higher-level structural cues, such as linguistic or multi-modal signals, to enhance robustness and semantic richness. Language embedding methods [36], such as CLIP [32], have inspired the use of vision-language models in scene graph prediction. These models introduce structural cues via meta-word embeddings [46], improving latent space interpretability, and language model tuning [21], which further enhances semantic accuracy. Meanwhile, multi-modal alignment and model distillation unify semantic representations across language, image, and 3D inputs into shared spaces [40], enabling more generalizable scene understanding. The recent method fuse pre-segmented image features [15], relational text queries [25], and 3D geometry for open-vocabulary scene graph prediction [23], expanding semantic label diversity. However, these methods still rely on instance-level point clouds inputs, limiting applicability in scenarios without reliable segmentation. To address this, some works explore scene graph prediction from coarse geometric segmentation. SGFN [39] handles non-instance inputs by inferring same-part semantics from over-segmented points patches reasoning using feature fusion. JointSSG [38] extends this by aggregating image features with 2D structural cues from video keyframes. Still, current methods lack an intuitive 3D interaction mechanism for actively guiding prediction and improving accuracy, which is the gap our approach aims to fill.

2.3 Scene Graph for Spatial Computing

Spatial computing has evolved alongside XR [5] technologies and is now closely linked to immersive human-computer interaction [30]. Recent research identifies its two key components: spatial content perception or awareness [8, 9, 14], and scene-level semantic understanding [11, 31], which support various intelligent applications, including context-aware interaction [43, 49] and haptic intent modeling [28, 48]. Semantic scene graphs bridge perception and reasoning by providing structured spatial representations. The work [41] explores scene understanding and editing, treating scene graph generation as a byproduct. By capturing contextual layouts, scene graphs have been applied to virtual–physical scenario alignment [10, 29], autonomous driving [3], task planning [20], generative scene design [13], and 3D-aware segmentation [18]. These applications demonstrate the potential of scene graphs in advancing embodied intelligence through spatial computing.

3 Method

Our SGSG aims to produce a refined 3D scene graph by interactively incorporating user corrections through XR stroke guidance. The overall SGSG pipeline is illustrated in Fig. 2. In the first step, we represent the temporal-spatial guidance features based on the input strokes (Sec. 3.2). In the second step, we introduce a guidance injection network to align and integrate these guidance features into the semantic prediction, thereby correcting it both locally and globally (Sec. 3.3). In the third step, we adapt a GNN-based inference backbone [39] to combine traditional 3D scene features with our guidance features and produce the final semantic prediction. Throughout these three steps, we employ a set of losses, including an intervention loss, to train the entire model (Sec. 3.4).

3.1 Main Methodology

We formulate the problem using the following function:

$$\{\hat{\mathbf{t}}, \hat{\mathbf{y}}\} = \mathcal{F}(\mathcal{S}, \mathbf{P}). \tag{1}$$

Here, the input to our pipeline consists of a stroke sequence $\mathcal{S} = \{\mathbf{S}_k\}_{k=0}^{N-1}$, representing N user strokes that interactively guide the generation process, and a scene point cloud $\mathbf{P}$. The model $\mathcal{F}$ outputs $\hat{\mathbf{t}}$, the predicted interaction types for each stroke (e.g., binding nodes, modifying edges, or hinting nodes), and structured labels $\hat{\mathbf{y}}$ representing refined scene semantics, including object identities and their predicates (e.g., "*table – standing on – ground*"). Following the three steps of our pipeline, we briefly describe the input-output flow of each component and their functionalities:

- **Guidance Representation:** It learns guidance representations from user strokes, capturing both temporal intent and geometric spatial contexts. Given an input stroke $\mathbf{S}_k$, it predicts the guidance type $\hat{t_k}$ and generates learned stroke guidance features $\mathbf{z}_k^{(\hat{t_k})}$, which encode the temporal meaning and spatial information necessary for refining the scene graph. (Blue background in Fig. 2)
- **Guidance Injection:** It receives the stroke guidance features $\mathbf{z}_k^{(\hat{t_k})}$ and the predicted guidance type $\hat{t_k}$ as inputs. Then it processes these to generate aligned features $\mathbf{f}^{(\hat{t_k})}$, which are then

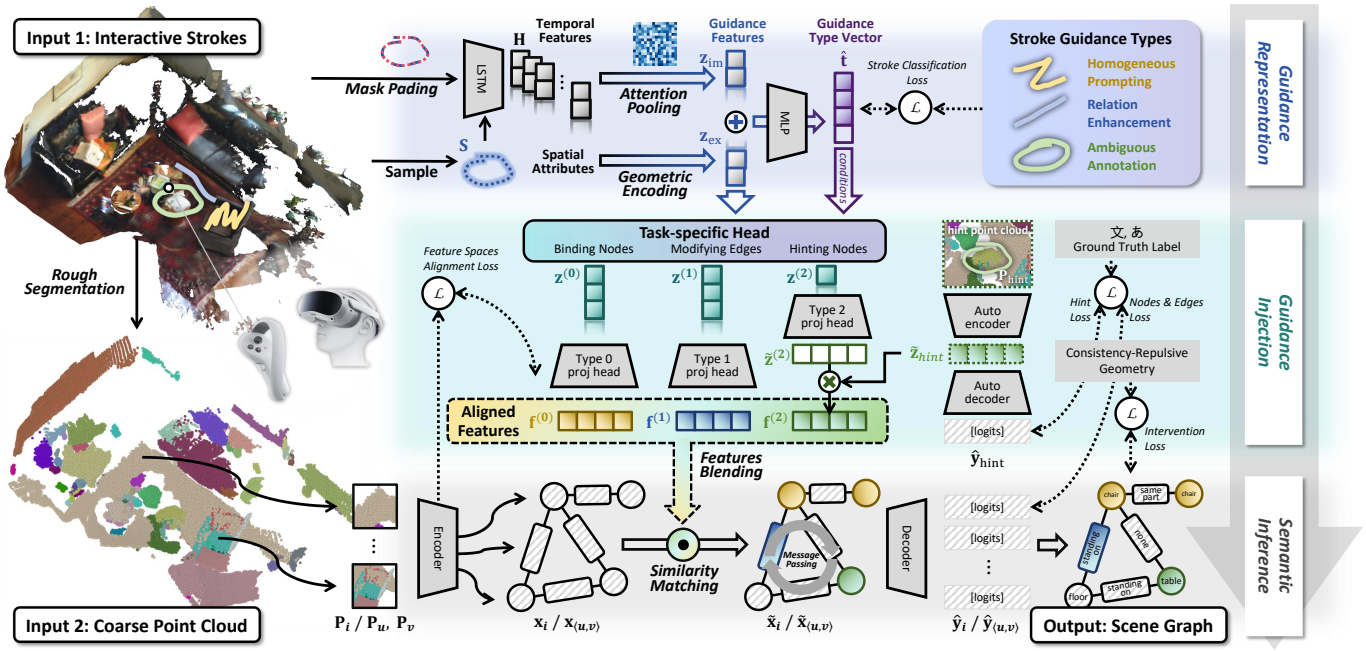


Fig. 2: Pipeline of our SGSG method. In this interactive model, the top two rows process user strokes to identify guidance types and generate guidance information. The *Guidance Representation* component (blue background) samples the strokes into $\mathbf{S}$, extracts their guidance features $[\mathbf{z}_{\text{im}}, \mathbf{z}_{\text{ex}}]$, and predicts the intended guidance type $\hat{\mathbf{t}}$. The *Guidance Injection* network (cyan background) uses a multi-head task specification structure to project these features into an aligned guidance space, which also encodes ambiguous hints (e.g., the green $\tilde{\mathbf{z}}_{\text{hint}}$). The resulting $\mathbf{f}$ is matched by similarity and injected into the *Inference* backbone (bottom row, gray background) to produce the refined $\tilde{\mathbf{x}}$.

injected into the backbone by mapping them to the corresponding semantic intermediate features $\mathbf{x}_i$ or $\mathbf{x}_{\langle u,v\rangle}$. Mixed updates are applied to these to produce $\tilde{\mathbf{x}}$, refining target features both locally and globally, and influencing the structure of the scene graph. (Cyan background in Fig. 2)

- **Semantic Inference:** It encodes the coarse point clouds input $\mathbf{P}_i$ to produce node representations [39] $\mathbf{x}_i$. For predicates between point clouds patches $\mathbf{P}_u$ and $\mathbf{P}_v$, edge features $\mathbf{x}_{\langle u,v\rangle}$ are computed. After being refined through **Guidance Injection** to $\tilde{\mathbf{x}}$, they are then passed through GNN, producing updated node features $\mathbf{x}'_i$ and edge features $\mathbf{x}'_{\langle u,v\rangle}$. The decoder generates objects and predicates semantic logits $\hat{\mathbf{y}}$, which are used to predict node and edge labels respectively for the scene graph. (Gray background in Fig. 2)

3.2 Stroke Guidance Representation

3.2.1 Stroke Definition

As different forms of interactive intention, we define three types of strokes to address specific prediction errors:

- **Homogeneous Prompting (Type 0):** *Binding nodes.* Brushing over large objects combines multiple semantic parts into a single entity, preventing them from being mistakenly interpreted as different separate semantics objects.
- **Relation Enhancement (Type 1):** *Modifying edges.* Linking two parts in the scene and correcting or enhancing the edges of the scene graph improves the accuracy of predicate predictions.
- **Ambiguous Annotation (Type 2):** *Hinting nodes.* Circling mismatched or unrecognized objects, such as small ones or those merging with the background, optimizes the precision and completeness of the scene graph.

These three stroke types are the result of an author-informed heuristic design, with exploratory validation conducted through a small set of collected real user strokes. Preliminary study showed meaningful clustering results, supporting the feasibility of the proposed pattern. See supplementary material for details. An input stroke, captured in the 3D scene via XR controllers, is first sampled at a certain frequency to produce an ordered sequence $\mathbf{S}_k = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{n-1})$, where each point $\mathbf{s}_i \in \mathbb{R}^3$. The sequence is then processed through two encoding pathways: implicit and explicit, deriving latent feature $\mathbf{z}_{\text{im}}$ and explicit feature $\mathbf{z}_{\text{ex}}$. These features capture the user's interactive intent incorporating spatial context and facilitate subsequent targeted corrections.

3.2.2 Implicit Temporal Feature Representation

The stroke sequence $\mathbf{S}$ is fed into an LSTM network to encode temporal dependencies. For each point $\mathbf{s}_i$, it generates a hidden state $\mathbf{h}_i \in \mathbb{R}^h$, forming the hidden states sequence $\mathbf{H} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{n-1})$ with the initial state $\mathbf{h}_0 = \mathbf{0}$. The LSTM processes the sequence as: $\{\mathbf{h}_i, \omega_i\} = \text{LSTM}(\mathbf{s}_{i-1}, \mathbf{h}_{i-1}, \omega_{i-1})$, where $1 \le i \le n-1$, and ω_i is the cell state with $\omega_0 = \mathbf{0}$. To highlight key temporal patterns, an attention mechanism is applied to the hidden states. We obtain an intermediate attention representation $\mathbf{U} \in \mathbb{R}^{n\times a}$ by applying a linear transformation and a non-linear activation to the hidden states: $\mathbf{U} = \tanh(\mathbf{H}\mathbf{W}_{\text{att}} + \mathbf{b}_{\text{att}})$, where $\mathbf{W}_{\text{att}} \in \mathbb{R}^{h\times a}$ and $\mathbf{b}_{\text{att}}$ are learnable parameters. The latent feature $\mathbf{z}_{\text{im}}$ is then computed as:

$$\mathbf{z}_{\text{im}} = \text{softmax}(\mathbf{U}\mathbf{w}_{\text{ctx}})^\top \mathbf{H}, \tag{2}$$

with context vector $\mathbf{w}_{\text{ctx}} \in \mathbb{R}^{a\times 1}$. This implicit feature captures temporal dynamics meaning such as trend, timing, points of interest, and key segments, providing permutation-variant interactive guidance.

3.2.3 Explicit Spatial Feature Representation

This captures the geometric and structural attributes of the stroke $\mathbf{S}$. It consists of point count n, cumulative distance $d = \sum_{i=1}^{n-1} \|\mathbf{s}_i - \mathbf{s}_{i-1}\|$, average curvature $\overline{\kappa} = \sum_{i=1}^{n-2} \frac{2\sin(\theta\langle \overrightarrow{\mathbf{s}_{i-1}\mathbf{s}_i}, \overrightarrow{\mathbf{s}_i\mathbf{s}_{i+1}}\rangle)}{(n-2)\cdot\|\mathbf{s}_{i+1}-\mathbf{s}_{i-1}\|}$, voxel volume ν, centroid $\mathbf{c}$, start point $\mathbf{s}_0$, end point $\mathbf{s}_{n-1}$, and cumulative cross product vector $\mathbf{v} = \sum_{i=1}^{n-2} \overrightarrow{\mathbf{s}_{i-1}\mathbf{s}_i} \times \overrightarrow{\mathbf{s}_i\mathbf{s}_{i+1}}$. These attributes are concatenated to form the spatial feature descriptor:

$$\mathbf{z}_{\text{ex}} = [n, d, \overline{\kappa}, \nu, \mathbf{c}, \mathbf{s}_0, \mathbf{s}_{n-1}, \mathbf{v}]. \tag{3}$$

This representation of spatial feature $\mathbf{z}_{\text{ex}}$ enables the network to more efficiently understand high-dimensional semantic guidance and intent from human inputs. Specifically, n and d together indicate stroke length and drawing speed, aiding in clustering long type 0 strokes. The $\overline{\kappa}$ separates straight and curved strokes, which is useful for identifying linear type 1 strokes as well as regions with high geometric variation. The ν, reflecting the voxel occupancy ratio, mitigates feature contamination from repetitive overpainting. The $\mathbf{c}$, $\mathbf{s}_0$, and $\mathbf{s}_{n-1}$ help infer spatial intent, such as the target area (type 0, 2) or connection endpoints (type 1). The $\mathbf{v}$ facilitates clustering of looped type 2 strokes with strong normal components and implicitly encodes the geometric surface orientation of the corresponding scene region.

3.2.4 Stroke Classification

This component enables our model to infer stroke guidance types based on the user's interaction intent. The combined feature vector $\mathbf{z} = [\mathbf{z}_{\text{im}}, \mathbf{z}_{\text{ex}}]$ is used as input for stroke classification, applied as:

$$\hat{t} = \arg\max(\text{softmax}(g_t(\mathbf{z}))) \tag{4}$$

where $g_t(\cdot)$ is an MLP, and $\hat{t} \in \hat{\mathbf{t}}$ represents the predicted interaction type of stroke $\mathbf{S}$, taking values in $\{0, 1, 2\}$ to satisfy task-specific correction requirements as in stroke definition. The $\mathbf{z}_{\text{ex}}$ improves the distinction of interaction types, for example, ν ensures that users can consistently select or revisit the same area within a stroke without misinterpreting the interaction type. The output $\mathbf{Z} = (\mathbf{z}_k)_{k=0}^{N-1}$ and $\hat{\mathbf{t}}$ are then passed to the next guidance injection network.

3.3 Guidance Injection Network

We propose a network that modifies the features of objects and predicates based on the specified predicted interaction types $\hat{t}$. Once extracted, the stroke features are first projected and aligned with the backbone's geometric space, and then blended into the inference backbone through strong or weak guidance injection mechanisms to refine scene graph generation locally or globally.

3.3.1 Strokes Guidance Feature Alignment

We align each user stroke guidance feature $\mathbf{z} \in \mathbf{Z}$ from the stroke feature space $\mathbb{R}^{\dim_{\mathbf{z}}}$ to the backbone's geometric space $\mathbb{R}^{\dim_{\mathbf{f}}}$, preparing for subsequent point-wise or global modifications and facilitating joint training. We design a task-specific projection head to adaptively transform features by p_i or $\mathcal{P}_i$ based on the task t:

$$\mathbf{f}^{(t)} = \delta_0 \cdot p_0(\mathbf{z}) + \delta_1 \cdot p_1(\mathbf{z}) + \delta_2 \cdot \mathcal{P}_2(\mathbf{z}, \mathbf{P}_{\text{hint}}), \tag{5}$$

where Dirac function $\delta_i = \delta(t-i)$ is 1 if $t = i$ and 0 otherwise. Projector $p_i(\cdot)$ is an MLP. Crucial for Type 2 interactions, the projector $\mathcal{P}_2(\cdot)$ combines stroke features with spatial context hint features to jointly compensate for the insufficient representations in the backbone inference: $\mathcal{P}_2(\mathbf{z}, \mathbf{P}_{\text{hint}}) = (1-\gamma_{\text{hint}}) \cdot p_2(\mathbf{z}) + \gamma_{\text{hint}} \cdot f_{\text{h}} \cdot (\mathbf{P}_{\text{hint}})$. Here, the hint input $\mathbf{P}_{\text{hint}}$ is the point cloud enclosed by the stroke, contributed by those inside a sphere with the center of stroke and its fitted radius. The function $f_{\text{h}}(\cdot) = [f_p(\cdot), f_{\text{des}}(\cdot)]$ is concatenated from the PointNet, and the descriptor as in [39], and γ_{hint} is the weighting factor. Particularly, we introduce an autoencoder afterward, where the resulting latent feature is decoded to produce hint logits $\hat{\mathbf{y}}_{\text{hint}} = \phi_{\text{hint}}(f_{\text{h}}(\mathbf{P}_{\text{hint}}))$. This logits will be incorporated into the loss constraints discussed in Sec. 3.4.

3.3.2 Strong Guidance Injection

The purpose of the strong guidance injection mechanism is to apply local modifications to the node or edge middleware features of the scene graph, directly adjusting the semantics of the objects or predicates involved in stroke interactions. The modification method varies depending on the specific stroke type:

- **Type 0:** If stroke $\mathbf{S}_k$ intersects the bounding box $b(\mathbf{P}_i)$ of a point cloud patch $\mathbf{P}_i$, the corresponding node feature $\mathbf{x}_i$ is blended with $\mathbf{f}_k^{(0)}$ to produce $\tilde{\mathbf{x}}_i$.

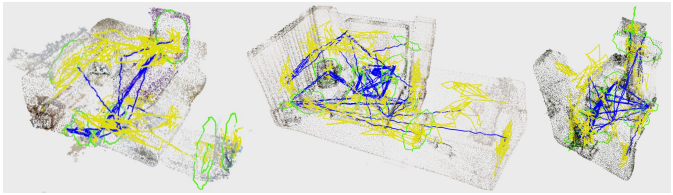


Fig. 3: Illustration of the SGstrokes dataset, showing stroke guidance annotations for different objects and relationships in three example scenarios. The interactive guidance types 0, 1, and 2 of strokes are marked in yellow, blue, and green, respectively.

- **Type 1:** The stroke $\mathbf{S}_k$ influences the edge feature $\mathbf{x}_{\langle u,v \rangle}$ between the patches nearest to the stroke's endpoints. It is blended with $\mathbf{f}_k^{(1)}$ to yield $\tilde{\mathbf{x}}_{\langle u,v \rangle}$.
- **Type 2:** If the patch is within the region of stroke $\mathbf{S}_k$, which is a spherical area around the centroid $\mathbf{c}$ with radius r, the patch feature $\mathbf{x}_i$ is blended with $\mathbf{f}_k^{(2)}$.

In all cases, the updated $\tilde{\mathbf{x}}$ is computed by re-weighting the original feature $\mathbf{x}$ with the aligned stroke feature $\mathbf{f}_k$: $\tilde{\mathbf{x}} = \frac{\mathbf{x}+\alpha \cdot \mathbf{f}_k}{1+\alpha}$, where α is the strong injection rate. The $\tilde{\mathbf{x}}$ is then used for subsequent message passing, thereby correcting the prediction error locally. For Type 1 strokes, additional node features of patches within each stroke endpoint's control region are also updated, which will be discussed in Sec. 3.4.

3.3.3 Weak Guidance Injection

Similar scene graph errors may recur in predictions of the same type. Weak guidance injection aims to correct these errors in a batch using fewer interactions than the number of errors. Since features have been well aligned in prior projection steps, objects or predicates with similar semantics or error patterns can be adjusted jointly. Taking the example of modifying multiple errors through a single $\mathbf{S}_k$, we first compute the mask M_k to include middleware features of the scene graph that are similar to the stroke guidance intent:

$$M_k = \{(\mathbf{x}, \rho_{\mathbf{x}}) \mid \rho_{\mathbf{x}} = \cos(\mathbf{f}_k, \mathbf{x}), \rho_{\mathbf{x}} > \tau\}, \tag{6}$$

where $\rho_{\mathbf{x}}$ is the cosine similarity between $\mathbf{f}_k$ and each node or edge feature $\mathbf{x}$, and τ is the similarity threshold. All features in M_k are re-weighted by $\mathbf{f}_k$: $\tilde{\mathbf{x}} = \frac{\mathbf{x}+\beta\, \rho_{\mathbf{x}}\, \mathbf{f}_k}{1+\beta\, \rho_{\mathbf{x}}}, \forall \mathbf{x} \in M_k$, where β is the weak injection rate. This can gently assign single stroke corrections across globally similar or semantically related parts of the scene graph.

3.4 Training Setup

Our training process used the 3RScan dataset [34] and our constructed SGstrokes dataset (Fig. 3). The training objective is to improve feature alignment, enhance inference quality, and address reasoning limitations on coarse point cloud patches and their relations. This is achieved using core and intervention losses, with the total loss defined as $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{core}} + \lambda_{\text{itv}} \mathcal{L}_{\text{itv}}$.

3.4.1 SGstrokes Dataset

Our SGstrokes dataset augments 3D scenes with stroke-based guidance and annotations for objects and predicates, which contains 291k guiding strokes across three interaction types, paired with 1335 scanned scenes based on 3RScan [34]. Each stroke $\mathcal{S}_{\text{sg}}$ is represented by $(\mathcal{S}, t, o, l, n, d)$, denoting stroke points, type, guided counter-object (or edge) index, desired label, number of sampled points, and stroke trajectory length. The dataset was generated using geometric algorithms tailored to each type, simulating realistic user interactions. As shown in Algorithm 1, noise augmentation aligns the dataset more intuitively with human interaction patterns, improving guidance for 3D scene graph generation tasks. The noise model used here is formulated based on a coarse fit to preliminary user data described in Sec. 3.2.1. Further details are provided in the supplementary material.

Algorithm 1 Stroke Generation Algorithm

Require: Stroke Type t, Counter-Object(s) Bounding Box b
Ensure: Generated Stroke Points $\mathcal{S}$

Initialize $\mathcal{S} = \{\}$, and compute center c from b
if $t = 0$ and $\max(b_x, b_y, b_z) > 0.2$ **then**
  Compute $volume \leftarrow b_x \cdot b_y \cdot b_z$
  $num\_points \leftarrow \text{clip}\left(30 + 40 \cdot \frac{volume - 0.008}{36 - 0.008}, 30, 70\right)$
  for $i = 1$ to num_points **do**
    Generate random point $\mathbf{S}_i$ inside $c \pm 0.4b$
    $\mathcal{S}$.append($\mathbf{S}_i + \Delta\mathbf{S}_i$), noise $\Delta\mathbf{S}_i \sim U(-0.1b, 0.1b)$
else if $t = 1$ **then**
  Select $start, end \sim U(c_{st,ed} \pm 0.5b_{st,ed})$
  Compute $length \leftarrow \|end - start\|$
  $num\_points \leftarrow \text{clip}\left(10 + 20 \cdot \frac{length - 0.1}{2.5 - 0.1}, 10, 30\right)$
  for $i = 1$ to num_points **do**
    Interpolate $\mathbf{S}_i \leftarrow start + i \cdot \frac{end - start}{(num\_points - 1)}$
    $\mathcal{S}$.append($\mathbf{S}_i + \Delta\mathbf{S}_i$), noise $\Delta\mathbf{S}_i \sim U(-0.05, 0.05)$
else if $t = 2$ and $\max(b_x, b_y, b_z) < 1.5$ **then**
  Find major axes $\mathbf{a}_1, \mathbf{a}_2$ from b
  Compute radius $r_1 \leftarrow b_{\mathbf{a}_1}/2, r_2 \leftarrow b_{\mathbf{a}_2}/2$
  Compute $perimeter \leftarrow \pi \cdot (r_1 + r_2)$
  $num\_points \leftarrow \text{clip}\left(20 + 30 \cdot \frac{perimeter - 0.2\pi}{3.0\pi - 0.2\pi}, 20, 50\right)$
  for θ from 0 to 2π with step $\frac{2\pi}{num\_points}$ **do**
    Compute $\mathbf{S}_{local} \leftarrow r_1 \cos(\theta)\mathbf{a}_1 + r_2 \sin(\theta)\mathbf{a}_2$
    Apply augmented noise $\Delta\mathbf{S}_i \sim N(0, 0.025)$
    $\mathcal{S}$.append($\mathbf{S}_i \leftarrow c + \mathbf{S}_{local} + \Delta\mathbf{S}_i$)
  return $\mathcal{S}$

3.4.2 Core Losses

Core losses $\mathcal{L}_{\text{core}}$ ensure classification accuracy and feature alignment consistency, defined as $\mathcal{L}_{\text{core}} = \mathcal{L}_{\text{class}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}$.

Classification Constraints Represented as $\mathcal{L}_{\text{class}}$, it minimizes discrepancies between process or result predictions and ground truth semantic labels by: $\mathcal{L}_{\text{class}} = \mathcal{L}_{\text{stroke}} + \mathcal{L}_{\text{hint}} + \mathcal{L}_{\text{node}} + \mathcal{L}_{\text{edge}}$. Here, $\mathcal{L}_{\text{stroke}} = \text{CE}(\hat{\mathbf{t}}, \mathbf{t})$ and $\mathcal{L}_{\text{hint}} = \text{CE}(\hat{\mathbf{y}}_{\text{hint}}, \mathbf{y}_{\text{hint}})$ are cross-entropy losses $\text{CE}(\cdot)$ for stroke guidance type classification and autoencoder hint prediction (Sec. 3.3) in the Type 2 projection head. Ground truth labels $\mathbf{t}$ and $\mathbf{y}_{\text{hint}}$ come from the SGstrokes dataset. $\mathcal{L}_{\text{node}}$ and $\mathcal{L}_{\text{edge}}$ use focal loss [39] to address class imbalance in the final semantic predictions.

Alignment Constraints Represented as $\mathcal{L}_{\text{align}}$, it ensures consistency between projected stroke features $\mathbf{f}$ and corresponding backbone features $\mathbf{x}$, allowing projectors to better transform features from stroke guidance to geometric space:

$$\mathcal{L}_{\text{align}} = \frac{1}{2n}\sum_{k=0}^{n-1}\left[\text{MSE}(\mathbf{f}_i^{(0)}, \mathbf{x}_k) + \text{MSE}(\mathbf{f}_i^{(2)}, \mathbf{x}_k)\right] + \frac{1}{m}\sum_{\langle u,v\rangle=0}^{m-1}\text{MSE}(\mathbf{f}_j^{(1)}, \mathbf{x}_{\langle u,v\rangle}); \quad i = \psi(k), j = \langle\psi(u), \psi(v)\rangle, \tag{7}$$

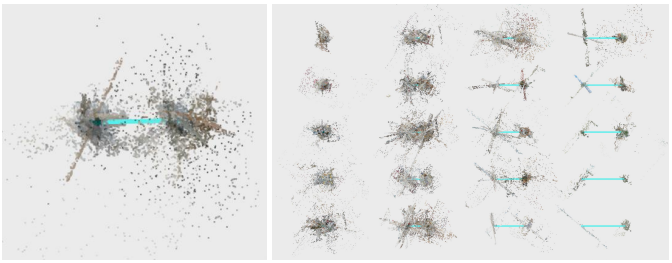
where n and m denote the node and edge counts of the scene graph, and $\psi(\cdot)$ maps point cloud patch node indices to object instance IDs. The $\langle\cdot,\cdot\rangle$ uses a prime hash function.

3.4.3 Intervention Losses

Intervention losses $\mathcal{L}_{\text{itv}}$ use additional information from stroke guidance and geometric priors to jointly address common prediction errors and enhance model robustness and generalization, defined as $\mathcal{L}_{\text{itv}} = \mathcal{L}_{\text{con}} + \mathcal{L}_{\text{rep}} + \lambda_{\text{geo}}\mathcal{L}_{\text{geo}}$.

Consistency Constraints Represented as $\mathcal{L}_{\text{con}}$, it ensures consistent semantic predictions within patches groups influenced by the same specific stroke:

- **Type 0:** Patches traversed by the same Type 0 stroke form group $\mathcal{G}_0$ and should have consistent semantic labels. Additionally, the connecting edges between them should follow the semantic label "*same part (sp)*".



(a) Point cloud distribution. (b) Distribution density by edge length.

Fig. 4: (a) Distribution of point cloud patches around edge endpoints, motivating the design of guidance intervention fitting; (b) Distribution of point-to-endpoint distances under different edge lengths (step size 0.2 m). These distributions are approximately modeled by a gamma function ($a = 2.0$, scale 0.125) used in training.

- **Type 1:** Patches near the endpoints of a Type 1 stroke form groups $\mathcal{G}_{1u}$ and $\mathcal{G}_{1v}$, with patches within each group having consistent labels. The fitting details for the endpoint neighborhoods are shown in Fig. 4.

Based on the above, the consistency loss is formulated as:

$$\mathcal{L}_{\text{con}} = C_{\mathcal{G}_0}\left[\sum_{i,j\in\mathcal{G}_0}\text{MSE}(\hat{y}_i, \hat{y}_j) + \sum_{u,v\in\mathcal{G}_0}\text{CE}(\hat{y}_{\langle u,v\rangle}, sp)\right] + \sum_{\mathcal{G}\in\{\mathcal{G}_{1u},\mathcal{G}_{1v}\}} C_{\mathcal{G}}\sum_{i,j\in\mathcal{G}}\Phi(\phi_i, \phi_j)\text{MSE}(\hat{y}_i, \hat{y}_j), \tag{8}$$

where $C_{\mathcal{G}} = \frac{2}{|\mathcal{G}|(|\mathcal{G}|-1)}$ and the probability harmonization function is $\Phi(\phi_i, \phi_j) = \frac{2\phi_i\phi_j}{\phi_i + \phi_j}$, with $\phi_i = \phi(\|\mathbf{c}_i - \mathbf{p}_e\|)$ representing gamma probabilities based on the distance from patch centers $\mathbf{c}_i$ to endpoints $\mathbf{p}_e$ as fitted in Fig. 4.

Repulsive Constraints Represented as $\mathcal{L}_{\text{rep}}$, it leverages additional information from stroke guidance and applies contrastive learning to push predictions away from clearly incorrect labels, repulsing irrelevant semantic features:

- **Type 1:** Edges predictions with Type 1 stroke should be repulsed from "*sp*" to reduce false positives. These are edges between patches groups $\mathcal{G}_{1u}, \mathcal{G}_{1v}$ in the stroke endpoints' neighborhood, with probability harmonization.
- **Type 2:** Foreground patch group $\mathcal{G}_{2f}$ semantics in the Type 2 stroke disambiguation hint region should be repulsed from background group $\mathcal{G}_{2b}$, and their predicates distanced from *sp*, enforcing exclusive classification.

Based on the above, the repulsive loss is formulated as:

$$\mathcal{L}_{\text{rep}} = -\frac{1}{|\mathcal{G}_{1u}||\mathcal{G}_{1v}|}\sum_{u\in\mathcal{G}_{1u}}\sum_{v\in\mathcal{G}_{1v}}\Phi(\phi_u, \phi_v)\text{FP}_{sp}(\hat{y}_{\langle u,v\rangle}) + \frac{1}{|\mathcal{G}_{2f}||\mathcal{G}_{2b}|}\sum_{i\in\mathcal{G}_{2f}}\sum_{j\in\mathcal{G}_{2b}}\mathcal{N}_j\left[\text{MSE}(\mathbf{x}_i, \mathbf{x}_j) - \text{FP}_{sp}(\hat{y}_{\langle i,j\rangle})\right], \tag{9}$$

where $\text{FP}_{sp}(\hat{y}) = \log\left(1 - \hat{y}\cdot\mathbb{1}_{\{sp\}} + \epsilon\right)$, with the label *sp* one-hot encoded as $\mathbb{1}_{\{sp\}}$, and a small constant ϵ. $\mathcal{N}_j = \mathcal{N}(\|\mathbf{c}_j - \mathbf{p}_\mathbf{c}\|; \mu(r), \sigma(r))$ is a normal distribution based on the distance from background patch $\mathbf{c}_j$ to the hint region center $\mathbf{p}_\mathbf{c}$, where $\mu(r) = 2r$, $\sigma(r)$ are the mean and standard deviation, with the 2-sigma rule keeping the background distribution within range $[r, 3r]$. r is the hint region's radius.

Geometric-Sensitive Constraints Represented as $\mathcal{L}_{\text{geo}}$, it leverages geometric priors to enhance classification:

- **Large Flat Objects:** Distinguish objects such as "*curtain*" or "*table*" by their principal normal vectors. Inspired by [29], we categorize them vertically or horizontally into groups V and H, penalizing mismatched orientations.

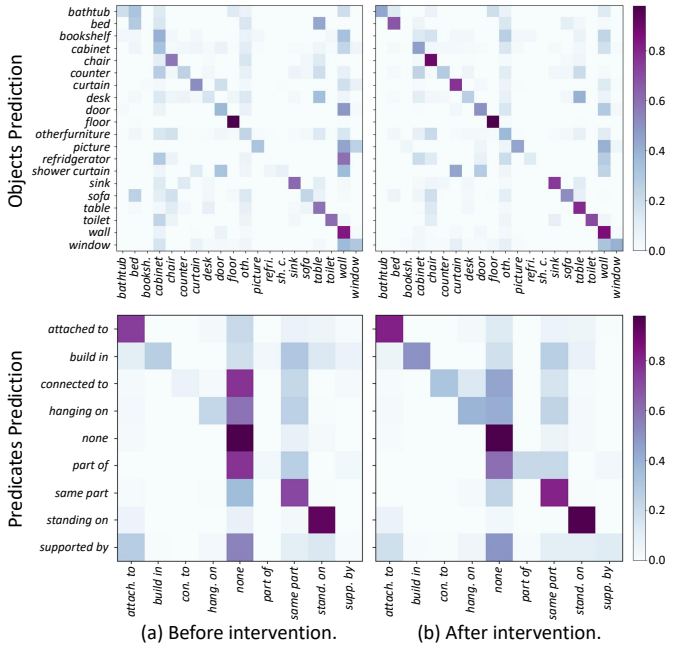


Fig. 5: Comparison of confusion matrices (a) before and (b) after intervention. As shown in the top row matrices, nodes prediction becomes more concentrated along the diagonal, especially for planar objects such as "*bed*", "*chair*", "*curtain*", "*desk*", "*refrigerator*", "*table*", or "*walls*". The bottom row matrices show improved edges prediction related to spatial hierarchy, including categories like "*built in*" or "*part of*".

- **Hierarchical Relationships:** Overlap in the z-projection boxes indicates predicates like "*built in*" or "*standing on*". An IoU-based loss refines these hierarchical predictions.

Based on the above, the geometric loss is formulated as:

$$\mathcal{L}_{\text{geo}} = \sum_{i \in V} (\mathbf{n}_i \cdot \mathbf{k})^2 + \sum_{i \in H} (1 - (\mathbf{n}_i \cdot \mathbf{k}))^2 + \sum_{\langle u,v \rangle = 0}^{m-1} \hat{y}_{\langle u,v \rangle} \, \mathbb{1}_{\{\text{hier}\}} \left[1 - \text{IoU}_{xy}(b(\mathbf{P}_u), b(\mathbf{P}_v))\right], \tag{10}$$

where $\mathbf{k}$ is the upward unit vector, and IoU_{xy} denotes the intersection over union in the vertical projection, with the denominator set to the smaller bounding box area to increase sensitivity for small objects in hierarchical structures. The improvements introduced by $\mathcal{L}_{\text{itv}}$ are shown in Fig. 5.

4 Experiments

4.1 Implementation Details

The models used in our experiments were trained and evaluated on a device with an i7-10700KF CPU, 32 GB RAM, and an RTX 3080Ti GPU. The user host used a PICO 4 XR device for spatial navigation and stroke input, communicating with the model server via a Flask-based protocol [17]. The average runtime latency of ours for scene graph feedback after each interaction was 261.5 ms, including 55.7 ms for communication, 18.7 ms for inference, 179.1 ms for 3D graph calibration, and 7.9 ms for rendering. A detailed analysis of the system implementation performance is provided in the supplementary material. During training, the batch size was 1, with a maximum of 300 epochs, early stopping patience set to 30, and an initial learning rate of 1×10^{-4}. We randomly sampled $k \in [0, 10]$ strokes per scene. When $k \to 0$, the inference process reduces relying on stroke guidance to prevent joint learning degradation. The similarity threshold was set to $\tau = 0.7$. Loss weights were $\lambda_{\text{itv}} = 0.1$, $\lambda_{\text{align}} = 1.0$, and $\lambda_{\text{geo}} = 2.0$. Stroke sampling frequency was 100 Hz, voxel density 0.1 m^3, and input stroke size padded to 100 points ($h = 16$, $a = 16$). The type classifier $g_t(\cdot)$ has two layers (32 and 16 units), projecting to size 4. Projectors $p_i(\cdot)$ have two layers (32 and 64 units), taking and yielding features with $\dim_{\mathbf{z}} = 32$ and $\dim_{\mathbf{f}} = 256$. Backbone settings follow those in [39]. To explore the effect of injection rates, we set up two configurations: Config. A with a strong injection rate $\alpha = 0.7$ and weak injection rate $\beta = 0.3$, and Config. B with $\alpha = 0.35$ and $\beta = 0.15$. Ablation studies for different settings or toggles are presented in Sec. 4.3.

4.2 Results and Analysis

We compared our SGSG method with several SOTA methods 3DSSG [35], SGFN [39], and JointSSG [38], on the 3RScan dataset, which features coarse-segmented geometry with 20 object and 9 predicate classes label annotations (Segm-l20).

4.2.1 Qualitative Comparison

Fig. 6 presents a qualitative comparison between the non-guidance SGFN method with coarse-segmented point cloud as input [39] and our SGSG. As shown in four example scenes, SGSG, with few strokes interaction, generates more accurate scene graphs compared to traditional methods. In the *hall* scene, some object labels were mispredicted, and the predicates to the ground were mismatched. After one ／ and one ⬭ stroke guidance, the label "*sofa*" was corrected to "*chair*", and several other mispredicted predicates with same error pattern were corrected to "*standing on*". Similarly, in the *study room*, ∿ strokes refined the armchair's structure by preventing over-segmented patches of the same object from being predicted as different semantic instances, and preserving their internal relations. After that, the ⬭ interaction successfully hinted at the "*cabinet*" from the "*wall*" background, correcting the object label while also decoupling their "*same part*" association. In the *park* scene, the "*wall*" of the hut was corrected with a ∿ stroke, and "*table*" was identified using a ⬭, which also helped separate it from distant objects previously grouped under the "*same part*" relation; ／ corrected the "*standing on*" predicate. In the *bedroom*, "*bed*" and "*cabinet*" were corrected through ∿ and ⬭ interactions; meanwhile, certain relational structures and their associated objects, such as "*wall - attached to - floor*", were generalized and corrected by the combined effect of multiple strokes. Fig. 7 shows the latent feature distributions of node semantic labels for the SGFN baseline, as well as for SGSG before and after incorporating interaction guidance. Our method enhances semantic embeddings and discrimination. It clearly distinguishes large objects like "*floor*", differentiates similar objects such as "*window*" and "*picture*", and closely clusters semantically similar items like "*curtain*" and "*shower curtain*". Although our encoder learns high-dimensional features before stroke integration, adding stroke guidance further improves precision. For instance, embedding distances between "*sink*" and "*toilet*" in similar scenes become more distinct. Extended qualitative comparisons and temporal evolution visualizations, along with corresponding analyses, are provided in the supplementary material.

4.2.2 Quantitative Comparison

We also conducted quantitative comparisons using the same benchmark codebase and parameter settings, and reporting the best results from three independent training runs. Tab. 1 presents the results on Segm-l20, comparing our method with SOTA methods, all of which can be adapted to take coarse-segmented point clouds as input. The three main columns represent metrics for object label semantics, predicate label semantics, and triplet relationship semantics, respectively. As shown, SGSG without stroke interaction input (Ours w/o $\mathcal{S}$) already slightly outperforms the SGFN baseline, indicating that joint training with our stroke representation and guidance injection network improves global representation and inference. With an average of 10 random strokes input from SGstrokes, our method surpasses SGFN across all accuracy (Acc.) metrics, with improvements of 7.5% and 2.2% in object and predicate accuracy respectively, and a 14.3% increase in relationship tuple accuracy, reflecting an increase in correct label predictions. For top-K recall (R@K), our method shows limited improvement at higher K values but excels in top-1 prediction, indicating that our model is more confident in the first prediction. Additionally, our method improves macro-precision (Prec.), demonstrating better prediction not only for major categories but also for those minor imbalanced classes through guidance corrections, surpassing the baseline by 17.2% and

Fig. 6: Comparison of SGFN [39] and our SGSG on Segm-I20. The first column shows the input coarse-segmented point cloud, and the fourth column adds the interactive strokes guidance for SGSG. The second and third columns display SGFN and SGSG results, with correctly predicted nodes circled in ⬭ and incorrectly predicted nodes framed in ▭. Node background color corresponds to coarse-segmentation pseudocolor. For predicates, correctly predicted edges are marked with ↗, and "*same part*" edges are highlighted with ↗. Incorrectly predicted edges are marked with ↗, and those falsely predicted as "*none*" are marked with ↗. Generically predicted edges not annotated in the dataset are marked with ↗.

Table 1: Quantitative performance comparison across methods and metrics.

	Method	Objects					Predicates					Relationships			
		R@1(Acc.)↑	R@3↑	R@5↑	R@10↑	Prec.↑	R@1(Acc.)↑	R@3↑	R@5↑	R@10↑	Prec.↑	R@1(Acc.)↑	R@3↑	R@5↑	R@10↑
Segm-I20	3DSSG [35]	0.4674	0.7477	0.8663	0.9657	0.2823	0.8322	0.9839	0.9965	**1.0**	0.4736	0.2248	0.3332	0.4126	0.5187
	SGFN [39]	0.6354	0.8489	0.9334	**0.9858**	0.4544	0.8351	**0.9846**	**0.9969**	**1.0**	0.4483	0.3988	0.4831	0.5856	0.6933
	Ours (w/o $\mathcal{S}$)	0.6617	0.8565	0.9325	0.9804	0.4964	0.8458	0.9809	0.9953	**1.0**	0.5054	0.4341	0.5399	0.6205	0.7254
	Ours	**0.6830**	**0.8633**	**0.9393**	0.9843	**0.5330**	**0.8538**	0.9801	0.9951	**1.0**	**0.5381**	**0.4561**	**0.5643**	**0.6446**	**0.7437**
	JointSSG [38]	*0.6880*	*0.9041*	*0.9630*	*0.9931*	*0.5786*	0.8454	*0.9887*	*0.9970*	1.0	0.5169	*0.4601*	*0.5848*	*0.6733*	*0.7771*

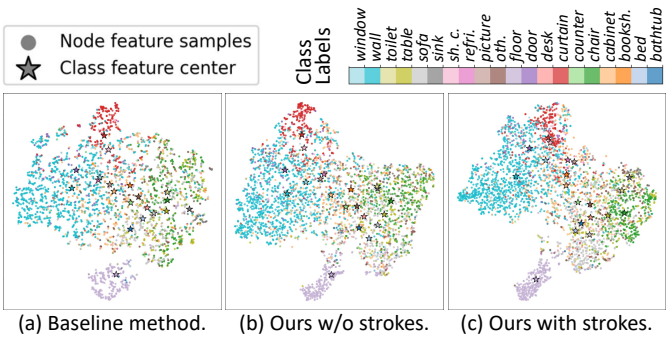
20.0% in object and predicate macro-precision, respectively. The row below the rule lists the multimodal input method JointSSG, which incorporates 2D serial visual information input; our method's performance is comparable to it. Notably, we surpass JointSSG in terms of predicate results, with accuracy and precision improved by 1.0% and 4.1%, respectively. This likely benefits from our type 1 interaction, a direct and simple geometric approach that explicitly establishes semantic connections between objects, which is challenging to achieve by relying solely on 2D images. We note that the average inference time of JointSSG during implementation reaches 472.1ms, over 25 times slower than that of ours, suggesting a trade-off between performance and efficiency.

4.3 Ablation Studies

We performed ablation studies to assess the impact of various design choices on our method. First, as shown in the first ablation column of Tab. 2, we evaluate the effect of applying point cloud normalization (nor.) before encoding. Although the baseline PointNet scales and normalizes input clusters, normalization in our method, where specific point cloud patches might be concatenated by strokes, could lose relative size and positional information, potentially leading to mismatches in large or relational objects. Surprisingly, comparisons between configurations M1–M3 and M4–M6 reveal that normalization consistently improves overall metrics instead. In M5, mRecall for objects and predicates increased by 8.4% and 4.5% respectively when compared with M2, suggesting that bypassing normalization forces the encoder to rely on rigid size and position features, which harms performance on objects with diverse morphologies under the same label or on predicting relationships across different scene scales. Next, in the second ablation column of Tab. 2, we assess the inclusion of explicit spatial encoding

Table 2: Ablation of model configurations on performance metrics.

Method	nor.	spt.	itv.	fit.	mRecall↑		Accuracy↑		Precision↑	
					obj.	pred.	obj.	pred.	obj.	pred.
M0					0.4176	0.4431	0.5549	0.8138	0.3816	0.4275
M1		✓			0.4144	0.4208	0.5830	0.8347	0.5045	0.4553
M2		✓	✓		0.4162	0.4185	0.5934	0.8299	0.4817	0.4547
M3		✓	✓	✓	0.4248	0.4127	0.6008	0.8303	0.4655	0.4269
M4	✓	✓			0.4965	0.4679	0.6567	0.8495	0.5111	0.4930
M5	✓	✓	✓		**0.5012**	0.4639	0.6691	0.8517	0.4969	0.4745
M6	✓	✓	✓	✓	0.4723	**0.4888**	**0.6830**	**0.8538**	**0.5330**	**0.5381**



(a) Baseline method. (b) Ours w/o strokes. (c) Ours with strokes.

Fig. 7: The object latent feature distribution under t-SNE of the (a) SGFN baseline method, (b) SGSG without strokes guidance injection, and (c) SGSG with strokes.

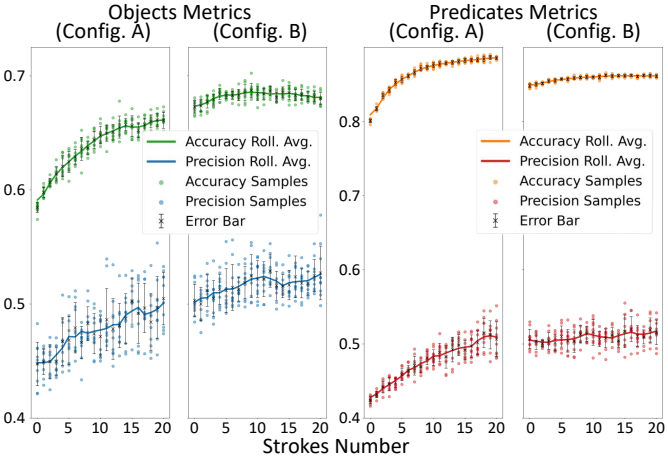


Fig. 8: Ablation of model's accuracy and precision under two injection rate configurations and various numbers of input strokes.

(spa.) in the stroke representation. In our early implementation as M0, we relied solely on temporal implicit encoding of stroke sequences, which made it difficult for the model to capture basic stroke properties such as curvature or endpoints, or align them with geometric space. From M1 onward, adding an explicit spatial encoding segment $\mathbf{z}_{\text{ex}}$ led to improvements in overall accuracy and precision. The third ablation column of Tab. 2 examines the use of the $\mathcal{L}_{\text{itv}}$ loss term (itv.), while the fourth ablation column of Tab. 2 investigates the impact of incorporating additional geometric probability harmonization fitting (fit.) for contrastive learning instead of relying solely on ground truth labels. Comparing M5 with M4, the inclusion of intervention improved overall prediction accuracy especially for objects, although precision slightly decreased. This degradation likely stems from the ground truth lacking neighborhood context, causing the model to correct semantics by intervention for large objects at the expense of low-frequency ones. Adopting probability harmonization fitting, as in M6, further improved predicate predictions and overall precision, likely due to better consideration of contact objects and spatial context. Similar trends were observed in the comparison between M2 and M3.

Table 3: Ablation on the impact of the solo interactive stroke type.

Type	N	Objects			Predicates			Relationships
		Acc.↑	mRec.↑	Prec.↑	Acc.↑	mRec.↑	Prec.↑	Acc.↑
	0*	0.6617	0.4566	0.4964	0.8458	0.4627	0.5054	0.4341
0	3	0.6704	0.4644	0.5040	0.8497	0.4793	0.5019	0.4403
	5	**0.6782**	0.4709	**0.5204**	**0.8532**	**0.4836**	0.5055	**0.4506**
1	3	0.6649	0.4634	0.5069	0.8455	0.4764	0.5115	0.4367
	5	0.6665	0.4613	0.4887	0.8474	0.4769	**0.5253**	0.4379
2	3	0.6717	0.4669	0.5065	0.8488	0.4709	0.5146	0.4410
	5	0.6746	**0.4848**	0.5200	0.8491	0.4733	0.5060	0.4442

Additionally, to isolate the effects of our three stroke interaction paradigms, we evaluated models with only N strokes from a single interaction type. As shown in Tab. 3, we expected type 0 and type 2 interactions to optimize object predictions and type 1 to enhance predicate predictions (corresponding cells highlighted in gray). The results confirm our expectations. Notably, some strokes improved both their target predictions and others. For instance, type 0 significantly boosted relationship metrics by refining both object predictions and the internal predicates of large object patches. Similarly, type 1 not only corrected predicate edges but also enhanced the semantics of objects nearby stroke endpoints, while type 2 considered both foreground and background, simultaneously extracting objects and refining related predicate semantics. These improvements in correlation metrics may also stem from message passing effects. In summary, all three types enhanced overall performance.

In line with the discussion at the beginning of this section, we conducted experiments for both Config. A and Config. B with N strokes inputs from 0 to 20 to investigate the effects of injection rates and the number of interactive strokes on model performance. Each set of hyperparameters was run 10 times, with sample and rolling averages of model accuracy and precision shown in Fig. 8. The model's performance improves rapidly at the beginning of stroke injection; however, as the number of injections increases, performance gains slow and even plateau in predicate metrics after 10 strokes, especially in terms of accuracy. This may be because an excessive number of stroke features causes saturation or dilutes the original 3D scene information, thereby introducing noise. As N increases, Config. A slightly outpaces Config. B in terms of performance improvement velocity; however, under both no-interaction and mass-interaction conditions, Config. B achieve higher performance. It shows a high injection rate likely causes the model to overemphasize stroke guidance while overlooking 3D scene details, resulting in coarser predictions. In summary, under Config. B with N from 5 to 10, the M6 model achieves the expected performance.

5 User Study

To evaluate the practical effectiveness and efficiency of SGSG, we conducted a within-subject user study comparing ours with three SOTA methods, as well as a naive SGSG without stroke interaction.

Participants We recruited 16 participants (10 males and 6 females), aged between 20 and 47, all with normal or corrected-to-normal vision and prior experience with XR.

Conditions and Setup For the control conditions (CC), we selected 3DSSG [35], SGFN [39], and JointSSG [38], referred to as CC1, CC2, and CC3, respectively. In addition, a stroke-disabled SGSG was included as CC4. Our full SGSG method served as the experimental condition (EC). The hardware and software setup followed that described in Sec. 4. Before the study, all participants received training on XR device operation and task instructions.

Tasks Each participant completed tests under all five conditions in randomized order. For each condition, participants freely navigated the predicted scene graphs in space and corrected incorrect semantic labels. In CC1–CC4, corrections were performed by hard correction, clicking on incorrect labels and selecting candidates from a list using the controller joystick. In EC, participants were additionally allowed to use SGSG's three type stroke-guidance interaction to assist in making corrections. To avoid fatigue, each condition included 3 randomly selected scenes, with a mandatory break of at least 2 minutes between conditions. For each scene under every condition, we recorded user controller actions count, task completion time, post-error rate, and locomotion trajectory. The post-error rate was defined as the ratio of incorrect labels remaining in the scene after user completed the corrections. After completing each condition, participants reported their perceived task load using four dimensions of the NASA-TLX [19] questionnaire: mental demand, physical demand, effort, and frustration.

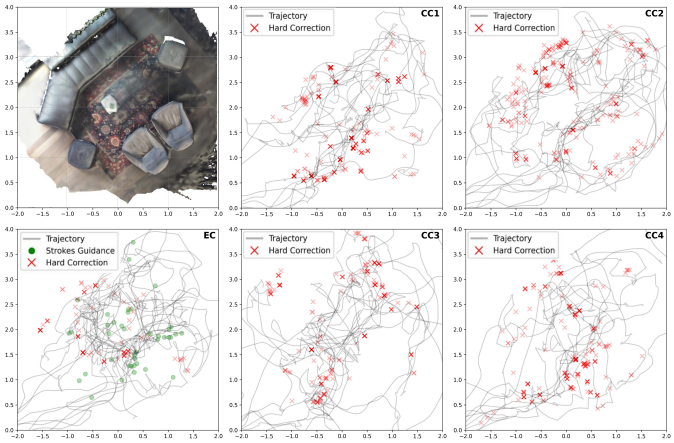
Results and Analysis The recorded metrics and Statistical analysis is presented in Tab. 4. We conducted t-tests and computed Cohen's d as a measure of effect size [4] for each comparison between each of CCs and EC. EC outperformed CC conditions across all metrics,

Table 4: User study task performance for each condition.

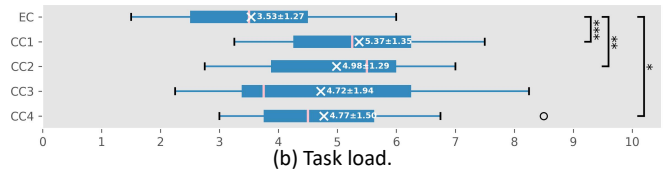
Cond.	**ActionsCnt**↓	p-value	Cohen's d	Effect Size	**TaskTime (s)**↓	p-value	Cohen's d	Effect Size	**PostErrRate**↓	p-value	Cohen's d	Effect Size
EC	**12.65 ± 7.07**				**48.81 ± 13.84**				**0.17 ± 0.11**			
CC1	21.87 ± 16.62	$< 0.001^*$	0.72	Medium	54.79 ± 20.78	0.085	0.34	Small	0.23 ± 0.11	0.014^*	0.49	Small
CC2	22.35 ± 14.96	$< 0.001^*$	0.83	Large	54.91 ± 16.78	0.044^*	0.40	Small	0.21 ± 0.09	0.043^*	0.40	Small
CC3	15.04 ± 12.94	0.241	0.23	Small	49.74 ± 20.00	0.782	0.05	Very small	0.27 ± 0.10	$< 0.001^*$	0.96	Large
CC4	21.09 ± 13.07	$< 0.001^*$	0.81	Large	50.57 ± 24.14	0.648	0.09	Very small	0.18 ± 0.10	0.660	0.09	Very small

Cohen's d effect sizes [4]: very small (<0.2), small (<0.5), medium (<0.8), large (<1.2), very large (<2.0), huge (≥2.0).

particularly in terms of actions count, where the effect size was large compared to both the CC2 baseline ($d = 0.83$) and CC4 ($d = 0.81$) methods. Regarding task completion time, EC achieved lower mean compared to CC2 ($d = 0.40$, small effect size); no statistical evidence supported significant differences when compared to other CCs. This may be due to the time gain associated with stroke drawing, as the EC group had a controller distance of 13.91, compared to 11.17 for CC2. For post-error rate, EC significantly outperformed SOTA methods, with a large effect size compared to the multimodal CC3 method ($d = 0.96$). This result is likely due to the 2D key frames strategy used for 3D label calibration, as users reported that some labels in CC3 were misaligned, extending beyond the scene boundaries or being occluded, which likely contributed to the higher post-error rate. User interaction behaviors and task load are shown in Fig. 9. The EC used stroke guidance to trade off hard corrections (visualized as green circles and red crosses in Fig. 9(a), respectively), with a reduced actions count. In Fig. 9(b), the task load box plot presents the mean results across four dimensions NASA-TLX. EC reported a lower cognitive load, with significant differences observed when compared to CC1 ($p < 0.001$), CC2 ($p = 0.004$), and CC4 ($p = 0.021$). These findings show that the stroke-guidance strategy in SGSG is both effective and practical.



(a) Accumulated trajectories and actions coordinate in the sample scene.



(b) Task load.

Fig. 9: User study results. Box plot showing the distribution of user ratings for each condition, with the means indicated by the white '×' symbols and the medians by the pink lines inside the boxes.

6 Conclusion, Limitations, and Future Work

We introduce SGSG, a Stroke-Guided Scene Graph generation approach for interactive semantic refinement in 3D space. By constructing the SGstrokes dataset and integrating three distinct XR stroke-guidance paradigms, our method captures high-level guidance knowledge while enabling both local and global corrections. We designed a model that incorporates stroke guidance representation, a guidance injection network, and intervention losses, including both consistency-repulsive and geometry-sensitive constraints, resulting in improved prediction accuracy and generalization. Experiments and the user study confirm that SGSG outperforms SOTA methods in key metrics, establishing a new benchmark for interactive 3D scene graph generation. This work contributes to the XR community by enabling interactive semantic modeling in immersive 3D environments, augmenting scene geometry with structured semantic information and enriching scene understanding for downstream XR applications with improved context and interaction accuracy. SGSG further supports real-time feedback and correction, opening up possibilities for immersive scene editing, context-aware placement, and embodied intelligence tasks.

Despite these promising results, our approach has several limitations. First, as an initial interactive prototype, it relies solely on human stroke input, excluding other modalities such as traditional 2D images or natural language, or even more direct forms like gesture or gaze intent, which could broaden its applicability. While our method is potentially extensible to other modalities supported by XR devices with richer sensory input, the current data collection lacks multi-modal annotations, limiting benchmarking and integration into applications that involve broader intent prediction. Second, the lightweight SGstrokes dataset may not fully represent the range of real-world interactions and scene configurations, especially in more complex or dynamic 3D environments. Its heuristic generation and parameter fitting lack adaptation to diverse user stroke styles, which may limit generalization across users. Lastly, handling evolving or open-ended scenes remains a challenge, where object delineation and relationships become increasingly intricate both spatially and temporally. Trained on a closed dataset, the model may not generalize well to incrementally expanding open scenes with new elements, temporal object motion, or spatial occlusion, highlighting the need for continual learning capabilities.

In future work, we plan to address these issues and further expand SGSG. One direction involves developing more intuitive interactive strategies by incorporating the aforementioned or other natural interaction modalities for scene graph editing. Another area is semantic-driven content generation, enabling the model to produce contextually meaningful scene elements for applications such as indoor scene generation, editing, or style transfer. Moreover, improving failure handling is important for usability, such as when repeated interactions yield no response or the correct label cannot be hit. Adding fallback mechanisms like noisy feature clearing could further improve efficiency of XR-based interactive generation applications. Implementing feedback loops where scene graph drives adaptive downstream tasks also becomes feasible, supporting embodied intelligence scenarios such as task navigation, industrial assembly, and XR telepresence through real-time contextual updates from the scene graph.

Supplementary Material

Supplementary material is available as a downloadable item in the IEEE *Xplore* digital library associated with this paper. It briefly includes: (1) stroke design and modeling details, (2) system performance analysis, and (3) extended visual and quantitative analysis.

Acknowledgments

This work is supported by the National Natural Science Foundation of China through Project 61932003, 62372026, by Beijing Science and Technology Plan Project Z221100007722004, by National Key R&D plan 2019YFC1521102, and by the fundamental research funds for the central universities.

References

[1] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. 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. 2

[2] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann. A comprehensive survey of scene graphs: Generation and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1–26, 2021. 2

[3] L. Chen, X. Wang, J. Lu, S. Lin, C. Wang, and G. He. Clip-driven open-vocabulary 3d scene graph generation via cross-modality contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27863–27873, 2024. 2

[4] J. Cohen. *Statistical power analysis for the behavioral sciences*. routledge, 2013. 8, 9

[5] A. Çöltekin, I. Lochhead, M. Madden, S. Christophe, A. Devaux, C. Pettit, O. Lock, S. Shukla, L. Herman, Z. Stachoň, et al. Extended reality in spatial sciences: A review of research challenges and future directions. *ISPRS International Journal of Geo-Information*, 9(7):439, 2020. 2

[6] Y. Cong, W. Liao, H. Ackermann, B. Rosenhahn, and M. Y. Yang. Spatial-temporal transformer for dynamic scene graph generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 16372–16382, 2021. 2

[7] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839, 2017. 2

[8] Z.-C. Dong, W. Wu, Z. Xu, Q. Sun, G. Yuan, L. Liu, and X.-M. Fu. Tailored reality: Perception-aware scene restructuring for adaptive vr navigation. *ACM Transactions on Graphics (TOG)*, 40(5):1–15, 2021. 2

[9] R. Fan, X. Shi, K. Wang, Q. Ma, and L. Wang. Scene-aware foveated rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2

[10] R. Fan, L. Wang, X. Liu, S. K. Im, and C. T. Lam. Real-scene-constrained virtual scene layout synthesis for mixed reality. *The Visual Computer*, 40(9):6319–6339, 2024. 2

[11] T. Feng, W. Wang, X. Wang, Y. Yang, and Q. Zheng. Clustering based point cloud representation learning for 3d analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8283–8294, 2023. 2

[12] 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*, pp. 21295–21304, 2024. 1

[13] 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*, pp. 21295–21304, 2024. 2

[14] Y. Ghasemi, H. Jeong, S. H. Choi, K.-B. Park, and J. Y. Lee. Deep learning-based object detection in augmented reality: A systematic review. *Computers in Industry*, 139:103661, 2022. 2

[15] G. Ghiasi, X. Gu, Y. Cui, and T.-Y. Lin. Scaling open-vocabulary image segmentation with image-level labels. In *European conference on computer vision*, pp. 540–557. Springer, 2022. 2

[16] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017. 2

[17] M. Grinberg. *Flask web development*. " O'Reilly Media, Inc.", 2018. 6

[18] H. Guo, H. Zhu, S. Peng, Y. Wang, Y. Shen, R. Hu, and X. Zhou. Sam-guided graph cut for 3d instance segmentation. In *European Conference on Computer Vision*, pp. 234–251. Springer, 2025. 2

[19] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, vol. 52, pp. 139–183. Elsevier, 1988. 8

[20] D. Honerkamp, M. Büchner, F. Despinoy, T. Welschehold, and A. Valada. Language-grounded dynamic scene graphs for interactive object search with mobile manipulation. *IEEE Robotics and Automation Letters*, 2024. 1, 2

[21] S. Koch, P. Hermosilla, N. Vaskevicius, M. Colosi, and T. Ropinski. Lang3dsg: Language-based contrastive pre-training for 3d scene graph prediction. In *2024 International Conference on 3D Vision (3DV)*, pp. 1037–1047. IEEE, 2024. 2

[22] S. Koch, P. Hermosilla, N. Vaskevicius, M. Colosi, and T. Ropinski. Sgrec3d: Self-supervised 3d scene graph learning via object-level scene reconstruction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3404–3414, 2024. 2

[23] S. Koch, N. Vaskevicius, M. Colosi, P. Hermosilla, and T. Ropinski. Open3dsg: Open-vocabulary 3d scene graphs from point clouds with queryable objects and open-set relationships. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14183–14193, 2024. 2

[24] C. Li, W. Li, H. Huang, and L.-F. Yu. Interactive augmented reality storytelling guided by scene semantics. *ACM Transactions on Graphics (TOG)*, 41(4):1–15, 2022. 1

[25] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022. 2

[26] R. Li, S. Zhang, and X. He. Sgtr: End-to-end scene graph generation with transformer. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19486–19496, 2022. 2

[27] Y. Liu, C. Long, Z. Zhang, B. Liu, Q. Zhang, B. Yin, and X. Yang. Explore contextual information for 3d scene graph generation. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):5556–5568, 2022. 2

[28] Z. Liu, J. Wu, L. Wang, X. Li, and S. K. Im. Proxy importance based haptic retargeting with multiple props in vr. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2

[29] Q. Ma, L. Wang, W. Ke, and S.-K. Im. Smigraph: a perceptually retained method for passive haptics-based migration of mr indoor scenes. *The Visual Computer*, pp. 1–21, 2023. 2, 5

[30] E. Pangilinan, S. Lukas, and V. Mohan. *Creating augmented and virtual realities: theory and practice for next-generation spatial computing*. " O'Reilly Media, Inc.", 2019. 2

[31] S. Peng, K. Genova, C. Jiang, A. Tagliasacchi, M. Pollefeys, T. Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 815–824, 2023. 2

[32] 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*, pp. 8748–8763. PMLR, 2021. 2

[33] T. Tahara, T. Seno, G. Narita, and T. Ishikawa. Retargetable ar: Context-aware augmented reality in indoor scenes based on 3d scene graph. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 249–255. IEEE, 2020. 1

[34] J. Wald, A. Avetisyan, N. Navab, F. Tombari, and M. Nießner. Rio: 3d object instance re-localization in changing indoor environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7658–7667, 2019. 2, 4

[35] J. Wald, H. Dhamo, N. Navab, and F. Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3961–3970, 2020. 2, 6, 7, 8

[36] X. Wang, Q. He, J. Liang, and Y. Xiao. Language models as knowledge embeddings. *arXiv preprint arXiv:2206.12617*, 2022. 2

[37] A. Werby, C. Huang, M. Büchner, A. Valada, and W. Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024. 1

[38] S.-C. Wu, K. Tateno, N. Navab, and F. 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. 2, 6, 7, 8

[39] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. 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. 2, 3, 4, 5, 6, 7, 8

[40] P. Xu, X. Zhu, and D. A. Clifton. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12113–12132, 2023. 2

[41] X. Yan, Z. Yuan, Y. Du, Y. Liao, Y. Guo, S. Cui, and Z. Li. Comprehensive visual question answering on point clouds through compositional scene manipulation. *IEEE Transactions on Visualization and Computer Graphics*, 30(12):7473–7485, 2023. 2

[42] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 670–685, 2018. 2

[43] H. Ye, J. Leng, C. Xiao, L. Wang, and H. Fu. Proobjar: Prototyping spatially-aware interactions of smart objects with ar-hmd. In *Proceedings*

of the 2023 CHI Conference on Human Factors in Computing Systems, pp. 1–15, 2023. 2

[44] G. Zhai, E. P. Örnek, S.-C. Wu, Y. Di, F. Tombari, N. Navab, and B. Busam. Commonscenes: Generating commonsense 3d indoor scenes with scene graphs. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[45] C. Zhang, J. Yu, Y. Song, and W. Cai. Exploiting edge-oriented reasoning for 3d point-based scene graph analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9705–9715, 2021. 2

[46] S. Zhang, A. Hao, H. Qin, et al. Knowledge-inspired 3d scene graph prediction in point cloud. *Advances in Neural Information Processing Systems*, 34:18620–18632, 2021. 2

[47] Y. Zhang, Y. Pan, T. Yao, R. Huang, T. Mei, and C.-W. Chen. Learning to generate language-supervised and open-vocabulary scene graph using pre-trained visual-semantic space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2915–2924, 2023. 2

[48] Q. Zheng, L. Wang, W. Ke, and S. K. Im. Vvir-om: Efficient object manipulation in vr with variable virtual interaction region. *International Journal of Human–Computer Interaction*, pp. 1–14, 2023. 2

[49] K. Zhou, C. Chen, Y. Ma, Z. Leng, H. P. Shum, F. W. Li, and X. Liang. A mixed reality training system for hand-object interaction in simulated microgravity environments. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 167–176. IEEE, 2023. 2