GraphWrit3R: End-to-End 3D Scene Graph Writing

Luka Milivojevic[1] **Nikola Popovic**[1†] **Sayan Deb Sarkar**[2] **Sebastian Koch**[3]
Iro Armeni[2] **Luc Van Gool**[1] **Danda Pani Paudel**[1]

[1]INSAIT, Sofia University "St. Kliment Ohridski"
[2]Stanford University [3]Ulm University

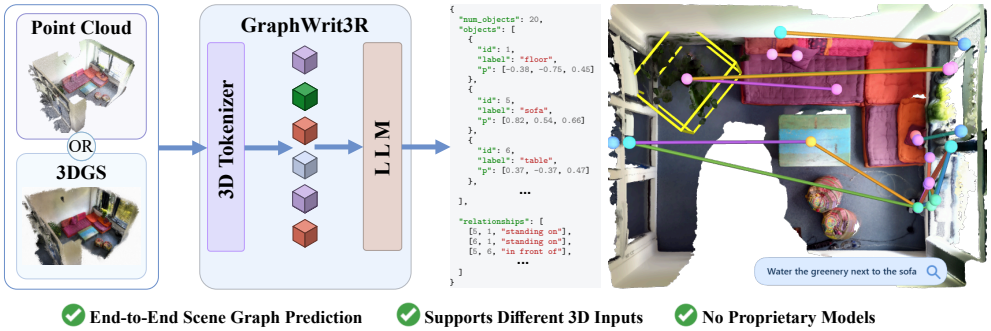

Figure 1: **GraphWrit3R: End-to-End 3D Scene Graph Writing.** GraphWrit3R takes a point cloud, 3D Gaussian Splats, or both as input and directly generates a scene graph as a structured JSON script. Unlike prior multi-stage pipelines with fragile intermediate representations, objects and their relationships are jointly predicted within a single end-to-end latent model, without reliance on proprietary models. The resulting graph additionally supports open-vocabulary querying.

Abstract

3D scene graphs provide a structured representation of complex environments by encoding objects, their semantic attributes, and the spatial and functional relationships between them. Current approaches for 3D scene graph generation suffer from several fundamental limitations. They rely on complex multi-stage pipelines with explicit intermediate representations, making systems fragile and prone to error propagation. They assume access to ground-truth object annotations during inference, which deviates from real-world scenarios. They depend on proprietary models, hindering open-source deployment, or incur prohibitively slow inference. We present GraphWrit3R, a simple end-to-end method that takes a 3D point cloud, Gaussian Splats, or a combination of both as input, and directly outputs a complete scene graph as a structured JSON script. The graph lists all objects, their semantic attributes, and the relationships between them, while avoiding all of the above mentioned limitations. The choice of multiple input modalities is purely for versatility, allowing a single set of weights to handle diverse scenarios. Point cloud inputs are encoded via Sonata and Gaussian Splat inputs via Chorus, with both modalities projected onto a shared voxel grid and fused through a novel per-voxel contrastive alignment loss before being decoded by a large language model. As a natural consequence of the LLM, GraphWrit3R also supports open-vocabulary querying. On the 3DSSG benchmark, our method achieves state-of-the-art performance on object class, predicate, and triplet recall, outperforming methods that rely on ground-truth object annotations during inference. We further provide qualitative results and analyze different input modality configurations, contrastive loss formulations, and token fusion strategies. Our project page is available at graphwrit3r.insait.ai.

† Project lead

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

The community has made tremendous progress in 3D scene understanding through semantic segmentation [68, 79, 80], object detection [50, 47, 29], instance decomposition [49, 74, 69], and visual-language grounding [30, 31, 32]. However, these advances have largely focused on individual components rather than the relational structure that connects them. 3D scene graphs [2, 59, 19] directly address this gap, offering a structured representation that encodes objects, their semantic attributes, and the spatial and functional relationships that tie them into a coherent whole. In robotic manipulation, grasping a target object that is *under* another object first requires moving the object *on top of* it out of the way. The robot must resolve spatial relations between objects to determine the correct action sequence. In AR, anchoring virtual content in a physical room requires reasoning about which surfaces support objects, which items occlude others, and how a user query like "show me everything on the shelf" maps onto the actual 3D arrangement in a view-consistent way. Nevertheless, scene graphs remain significantly underexplored compared to their instance-level counterparts.

Current approaches for 3D scene graph generation suffer from several fundamental limitations. First, most methods rely on complex multi-stage pipelines [26, 27, 71, 34, 78, 13, 12, 46], where objects are detected or segmented from 3D and/or 2D inputs, per-object features are extracted, and graph nodes and edges are initialized via multi-view feature aggregation before being classified by a graph neural network, frozen LLM, or VLM. Each stage introduces its own failure modes, making the overall system fragile, prone to error propagation, and computationally inefficient. Second, many existing methods assume access to ground-truth object bounding boxes or instance masks during inference [59, 77, 24, 25, 26, 27, 71, 34] an assumption that is rarely satisfied in real-world scenarios where such annotations are unavailable. Any object or mask estimator used as a substitute inevitably introduces additional errors that cascade through the remainder of the pipeline. Third, many recent methods rely heavily on proprietary models via API calls [34, 78, 13, 12, 46], making them unsuitable for full open-source deployment. Finally, methods that avoid proprietary dependencies often suffer from prohibitively slow inference, requiring minutes or hours to process a single scene [26, 71, 34, 27].

We introduce GraphWrit3R to address these limitations. Given a 3D scene as a point cloud, or a set of Gaussian Splats [22], or a multi-modal combination of both, GraphWrit3R directly generates a scene graph as a structured JSON script, listing all objects, their semantic attributes, and the relationships between them (see Fig. 1). As a natural consequence of the LLM backbone, it also supports open-vocabulary scene graph querying. GraphWrit3R operates entirely in latent space, requiring no fragile intermediate explicit representations, no proprietary model calls, and no ground-truth object annotations as input during inference (see Fig. 2). *The choice to support multiple input modalities is motivated purely by versatility*, rather than performance. We train with multi-modal inputs and evaluate with a single modality during inference, reflecting real-world settings where different 3D representations exist across different environments. Point cloud inputs are encoded via Sonata [68] into spatially grounded tokens, while Gaussian Splat inputs are encoded via Chorus [32], both producing representations on a shared voxel grid. When both modalities are present, tokens at co-located voxels are fused and passed to the Qwen2.5-0.5B [55] LLM, which decodes the full scene graph. To enable coherent cross-modal fusion, we introduce a per-voxel contrastive alignment loss that brings Sonata and Chorus features into a common representational space.

Our method is inspired by SpatialLM [39], which encodes point clouds with Sonata and uses an LLM to produce a structured JSON list of objects from indoor scenes. We leverage this foundation model by initializing both our Sonata encoder and LLM from the SpatialLM indoor fine-tuning checkpoint, providing a strong 3D prior. However, GraphWrit3R departs from SpatialLM, in both scope and design. Rather than predicting a flat list of objects, we extend the output representation to a full relational scene graph. Also, we introduce a complementary Gaussian Splat encoding branch via Chorus. Combined with a voxel-level fusion mechanism trained with a contrastive alignment objective, this yields a system with one set of weights that handles multiple input modalities. Finally, we fine-tune the resulting architecture end-to-end for scene graph generation.

We evaluate GraphWrit3R on the 3DSSG benchmark [59] across standard recall-based metrics for object class, predicate, and relationship triplet prediction, achieving state-of-the-art results against established baselines. Notably, most competing methods operate under the privileged assumption of ground-truth object annotations at inference time and incur substantially higher inference latency. We further analyze different input modality configurations, contrastive loss formulations, and token fusion strategies. Finally, we demonstrate that jointly supervising object detection and relationship

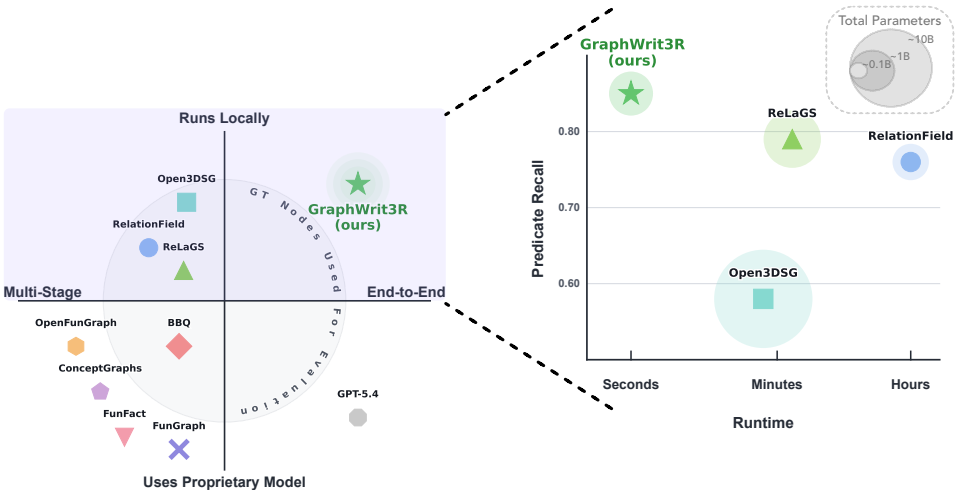

Figure 2: **Comparison to Relevant Methods.** On the left, we categorize relevant scene graph generation methods along several key dimensions. GraphWrit3R is the only method operating in a fully end-to-end latent fashion, requiring no fragile intermediate explicit representations, no proprietary models calls, and no ground-truth object annotations at inference time. On the right, we zoom in on methods that do not rely on proprietary models, showing that GraphWrit3R simultaneously achieves the highest performance and orders of magnitude faster inference.

prediction yields stronger detection performance than object supervision alone, suggesting that relational context provides a useful inductive signal for grounding individual objects.

Our contributions can be summarized as follows:

- We propose an end-to-end scene graph generation method outputting structured JSON, without intermediate explicit representations, ground-truth annotations, or proprietary models.
- We introduce a unified multi-modal 3D encoder accepting point clouds, or 3DGS, or both, with a novel per-voxel contrastive alignment loss for cross-modal fusion.
- We demonstrate state-of-the-art scene graph results on 3DSSG, analyze key design choices, and show that relational supervision benefits object detection.

2 Related work

3D Perception. Early 3D scene understanding relied on complex multi-stage pipelines involving explicit reconstruction, proposal generation, and feature aggregation. Driven by advancements in 3D backbones [43, 44, 7, 73, 81, 70, 69], the field has largely shifted toward end-to-end latent-space architectures. This transition is evident in object detection, where early pipeline-heavy methods [42, 62, 61] have been superseded by fully end-to-end detectors [50, 47, 29]. Similarly, in segmentation, multi-stage clustering and per-scene optimization approaches [21, 40, 36, 54, 41, 23, 28] are rapidly being replaced by unified latent architectures for both closed- [73, 81, 70, 69, 68, 79, 80] and open-vocabulary [30, 31, 37, 32] settings. Despite this broader trend, 3D scene graph generation remains stuck in the multi-stage paradigm. Existing methods [59, 25, 67, 24, 26, 64, 13, 34] first extract explicit instances to build an initial graph, then apply GNNs [59, 25, 67, 24, 64] or LLMs [13, 34, 26] to predict node and edge properties. This cascaded design introduces complexity and propagates errors. To our knowledge, GraphWrit3R is the first method to break this pattern. By utilizing a fully end-to-end latent pipeline, our approach directly generates a structured scene graph in JSON format without relying on explicit intermediate predictions.

3D Scene Graphs. Beyond object-centric perception, 3D scene graphs model inter-object relationships for scene understanding. Early works introduced hierarchical representations [2] and formalized semantic relationships [59]. Subsequent methods enabled real-time estimation [19] and refined predictions via incremental estimation, scene priors, and pre-training [67, 66, 77, 25, 24, 64]. More recent approaches further improve closed-vocabulary prediction through edge-centric relational reasoning [38] and volumetric multi-granularity feature modeling for objects and their relations [18].

However, these approaches remain confined to fixed, training-time taxonomies. Recent methods achieve open-vocabulary capabilities by integrating foundation models. One line of work distills features into language-queryable 3D representations using GNNs [26], radiance fields [27, 71], or hierarchical graphs [65]. Recent work also explores geometry-guided discrete diffusion to jointly reason over object geometry and semantic relationships [10]. Another line of work extracts relationships by aggregating natural-language outputs from frozen LLMs [13, 34], often using advanced prompting for fine-grained affordances [78, 46]. Furthermore, recent work extends open-vocabulary scene graphs toward language-guided interaction and downstream tasks by combining dynamic graph construction with retrieval-augmented reasoning [75]. Despite their flexibility, these methods rely on multi-stage pipelines and stacked, costly API calls to closed-source models [1, 8], significantly increasing latency. In contrast, we formulate 3D scene graph generation as an efficient, end-to-end learning objective. By removing dependencies on external APIs, our approach enables efficient and local predictions that generalize across diverse environments.

3D Vision-Language Models. 3D vision-language models (3D VLMs) enable natural language interaction with 3D scenes. Following early 3D captioning pipelines [6], modern architectures generally encode 2D and 3D modalities into tokens for LLM-driven visual question answering (VQA). These models employ diverse encoding strategies, including joint fine-tuning of 3D object tokens [63], cross-attention with frozen encoders [5], and multi-modal fusion of 2D, 3D, or BEV features [16, 17, 11, 85]. Recent advances further refine this via hierarchical tokenization [33, 76], adaptive resolution [84], or Gaussian Splat inputs [15]. Despite these architectural advances, most 3D VLMs produce plain-text VQA answers lacking spatial grounding, limiting their utility in embodied applications. While recent models like SceneScript [3] and SpatialLM [39] address this by outputting flat JSON lists of detected 3D objects, they ignore inter-object relationships. Our GraphWrit3R goes a step further by extending structured language outputs to the relational structure of the scene, moving beyond flat object lists toward a richer, relationally grounded representation of 3D environments.

3 Method

3.1 Overview

GraphWrit3R predicts a 3D scene graph $\mathcal{G} = (\mathcal{O}, \mathcal{R})$ as a structured JSON, given a 3D scene represented as a point cloud $\mathcal{P}$, or Gaussian Splats $\mathcal{S}$, or both. Here, $\mathcal{O} = \{o_i\}_{i=1}^{N}$ denotes the set of detected objects, where each object $o_i = (\ell_i, \mathbf{p}_i, \mathbf{s}_i, \theta_i)$ contains a natural-language semantic label ℓ_i, 3D centroid $\mathbf{p}_i \in \mathbb{R}^3$, bounding box extents $\mathbf{s}_i \in \mathbb{R}^3$, and yaw angle $\theta_i \in \mathbb{R}$. The relation set $\mathcal{R} = \{\{(o_i, r_{ij}^{(m)}, o_j)\}_{m=1}^{M_{ij}}\}_{i,j=1}^{N}$ contains directed subject-predicate-object triplets, where $r_{ij}^{(m)}$ is a natural-language predicate and M_{ij} denotes the number of predicates between objects o_i and o_j. Multiple relations may exist between the same pair of objects. Fig. 3 shows the full pipeline. Sonata [68] encodes $\mathcal{P}$ and Chorus [32] encodes $\mathcal{S}$ into tokens on a shared voxel grid (Sec. 3.2). Co-located voxel tokens are averaged when both modalities are present, or passed through otherwise. A two-layer MLP projects the fused tokens into the embedding space of the LLM, which decodes $\mathcal{G}$ autoregressively as a JSON script (Sec. 3.2). Training combines token-level cross-entropy loss at the LLM output with a contrastive per-voxel alignment loss between Sonata and Chorus (Sec. 3.3). Furthermore, a single set of learned weights for the whole architecture handles all three input configurations at inference, namely $\mathcal{P}$, $\mathcal{S}$, or both. Therefore, Sonata and Chorus must produce tokens in a shared latent space, which is enforced by our alignment loss as explained in Sec. 3.3. Moreover, we build on SpatialLM [39] as a foundation model, rather than training a 3D LLM from scratch, and reuse its Sonata encoder, projector, and LLM weights at the beginning of our training.

3.2 Scene Graph Generation

Point Cloud Encoding. Given a point cloud $\mathbf{P} \in \mathbb{R}^{N_p \times 6}$ of $\{x, y, z\}$ coordinates and RGB color, Sonata [68] returns voxelized tokens $\mathbf{Z}_\mathrm{P} = \{(\mathbf{v}_i, \mathbf{f}_i^\mathrm{P})\}_{i=1}^{M_p}$, where $\mathbf{v}_i \in \mathbb{Z}^3$ is a sparse voxel coordinate, $\mathbf{f}_i^\mathrm{P} \in \mathbb{R}^d$ is its feature, and M_p is the number of occupied voxels. Sonata is a sparse voxel transformer built on the PTv3-M2 backbone [69], pretrained by self-distillation on 140k indoor and outdoor scenes. We initialize Sonata from the fine-tuned checkpoint of SpatialLM [39], which was fine-tuned jointly with the downstream MLP and LLM for indoor scene parsing.

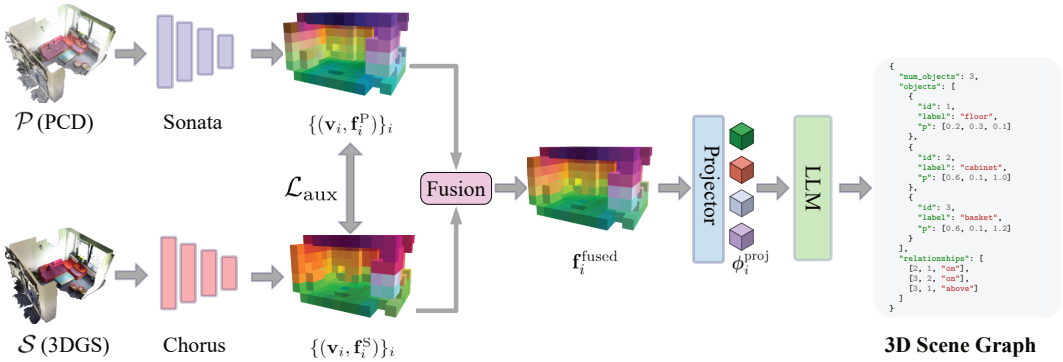

Figure 3: **GraphWrit3R.** Each 3D modality is processed by a designated encoder. Sonata [68] encodes point clouds and Chorus [32] encodes Gaussian splats into voxel tokens on a shared grid. Tokens at co-located voxels are averaged, projected by an MLP into the LLM's embedding space, and decoded autoregressively into a scene graph in structured JSON format.

Gaussian Splats Encoding. Given Gaussian Splat $\mathcal{S} = \{(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \mathbf{c}_j, \alpha_j)\}_{j=1}^{N_s}$, where for each splat j, $\boldsymbol{\mu}_j \in \mathbb{R}^3$ denotes the center, $\boldsymbol{\Sigma}_j \in \mathbb{R}^{3\times3}$ the covariance matrix parameterized via scale and rotation [22], $\mathbf{c}_j \in \mathbb{R}^3$ the color, and $\alpha_j \in \mathbb{R}$ the opacity, Chorus [32] outputs $\mathbf{Z}_{\mathrm{S}} = \{(\mathbf{v}_i, \mathbf{f}_i^{\mathrm{S}})\}_{i=1}^{M_s}$, where $\mathbf{v}_i \in \mathbb{Z}^3$ is a sparse voxel coordinate, $\mathbf{f}_i^{\mathrm{S}} \in \mathbb{R}^d$ is its feature, and M_s is the number of occupied voxels that are on the same voxel grid as Sonata. Chorus also uses the PTv3-M2 backbone, making it a feed-forward sparse voxel transformer with the same structure as Sonata. The key difference lies in pretraining. Rather than self-supervision, Chorus distills three 2D foundation models into a shared backbone via teacher-specific projection heads, namely SigLIP2 [57] for language alignment, DINOv3 [52] for visual features, and PE-Spatial [4] for object awareness. The shared PTv3 backbone between the two encoders is what makes feature-level alignment between them tractable.

Fusion. When both 3D modalities are present, we average features at every voxel location $\mathbf{v}_i$ from both encoders.

$$\mathbf{f}_i^{\mathrm{fused}} = \tfrac{1}{2}\left(\mathbf{f}_i^{\mathrm{P}} + \mathbf{f}_i^{\mathrm{S}}\right), \qquad i \in \mathcal{V}_{\mathrm{P}} \cap \mathcal{V}_{\mathrm{S}}. \tag{1}$$

If a specific voxel of one modality does not have a counterpart feature in the other, which happens infrequently due to the same nature of $\mathcal{P}$ and $\mathcal{S}$, that voxel is discarded. If only one modality is used during inference, $\mathbf{Z}_{\mathrm{fused}}$ reduces to either $\mathbf{Z}_{\mathrm{P}}$ or $\mathbf{Z}_{\mathrm{S}}$.

Projection. A two-layer MLP ϕ maps each fused token $\mathbf{f}_i^{\mathrm{fused}} \in \mathbb{R}^d$ into the LLM's input embedding space. We initialize ϕ from the projector of SpatialLM and fine-tune it end-to-end.

LLM. The projected tokens $\{\boldsymbol{\phi}_i^{\mathrm{proj}} = \phi(\mathbf{f}_i^{\mathrm{fused}})\}_{i=1}^{M}$, $\boldsymbol{\phi}_i^{\mathrm{proj}} \in \mathbb{R}^h$ enter the LLM after a fixed text prompt that specifies the desired output structure. Following SpatialLM [39], the LLM autoregressively decodes a structured JSON script representing the scene graph $\mathcal{G}$ defined in Sec. 3.1. The sequence contains object entries with semantic labels, 3D centroids, box extents, and yaw angles, followed by relationship entries represented as subject-predicate-object triplets.

3.3 Training Objective

Cross-Modal Alignment. Since Sonata and Chorus produce tokens on the same voxel grid (Sec. 3.2), we can pair their features by voxel coordinate. For every dual-modality scene in a batch, let

$$\mathcal{B} = \{(\mathbf{f}_i^{\mathrm{P}}, \mathbf{f}_i^{\mathrm{S}}) : \mathbf{v}_i \in \mathcal{V}_{\mathrm{P}} \cap \mathcal{V}_{\mathrm{S}}\} \tag{2}$$

be the set of co-located token pairs, and let $\tilde{\mathbf{f}} = \mathbf{f}/\|\mathbf{f}\|_2$ denote ℓ_2-normalization. The cross-modal alignment loss has three terms, each acting on these pairs. The first is symmetric InfoNCE [48], where the positive for a Sonata token at voxel $\mathbf{v}_i$ is the Chorus token at the same voxel, and the negatives are Chorus tokens at all other voxels within the same scene. Denoting the set of co-located voxel locations in that scene by $\mathcal{V}$:

$$\mathcal{L}_{\mathrm{InfoNCE}} = -\frac{1}{2|\mathcal{V}|}\sum_{i\in\mathcal{V}}\left[\log\frac{\exp(s_{ii}/\tau)}{\sum_{j\in\mathcal{V}}\exp(s_{ij}/\tau)} + \log\frac{\exp(s_{ii}/\tau)}{\sum_{j\in\mathcal{V}}\exp(s_{ji}/\tau)}\right], \tag{3}$$

with $s_{ij} = \langle \tilde{\mathbf{f}}_i^{\mathrm{P}}, \tilde{\mathbf{f}}_j^{\mathrm{S}} \rangle$ and temperature τ. The loss is computed independently for each scene and averaged across the batch. The second is a per-pair cosine term $\mathcal{L}_{\cos} = 1 - \langle \tilde{\mathbf{f}}_i^{\mathrm{P}}, \tilde{\mathbf{f}}_i^{\mathrm{S}} \rangle$, and the third is a feature-level MSE term $\mathcal{L}_{\mathrm{mse}} = \|\mathbf{f}_i^{\mathrm{P}} - \mathbf{f}_i^{\mathrm{S}}\|_2^2$. The three terms do different things: cosine pulls paired tokens onto a shared direction, MSE pulls them onto a shared magnitude, and InfoNCE keeps tokens at different voxels apart, so that alignment does not collapse the feature space. The combined alignment loss is: $\mathcal{L}_{\mathrm{aux}} = \lambda_{\mathrm{nce}}\mathcal{L}_{\mathrm{InfoNCE}} + \lambda_{\cos}\mathcal{L}_{\cos} + \lambda_{\mathrm{mse}}\mathcal{L}_{\mathrm{mse}}$. The features of Sonata are already aligned with the LLM in SpatialLM, since they were jointly trained. By contrast, Chorus, was pretrained to match 2D foundation model features. Therefore, we stop the propagation of $\mathcal{L}_{\mathrm{aux}}$ gradients through Sonata, such that they propagate exclusively through Chorus. This pulls Chorus's feature space closer to Sonata's, which keeps the LLM's input distribution stable and lets the cross-entropy signal continue to land where SpatialLM expects it. The shared PTv3 backbone between the two encoders is what makes the asymmetric alignment well-posed. If the architectures differed, matching feature statistics would not imply matching feature semantics.

Final Loss. We supervise the JSON output with the standard next-token cross-entropy loss $\mathcal{L}_{\mathrm{CE}}$, and add the auxiliary alignment loss with scalar weight α to form the full objective $\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \alpha\,\mathcal{L}_{\mathrm{aux}}$. Each training batch contains a mix of point-cloud-only, Gaussian-only, and dual-modality scenes, while $\mathcal{L}_{\mathrm{aux}}$ is computed only on the dual-modality subset. This yields a single set of weights that handles all three input configurations at inference, namely $\mathcal{P}$, $\mathcal{S}$, or both.

4 Experiments

4.1 Experimental Setup

Datasets and tasks. We evaluate scene graph prediction on 3DSSG [59], using its evaluation vocabulary of 160 object classes and 27 predicate classes. For training, we use $1,170$ point-cloud scenes from SceneVerse [20], of which 620 have Gaussian-splat representations from SceneSplat++ [37], with scene graph annotations from 3DSSG. For out-of-domain evaluation, we additionally use the official ScanNet validation set comprising of 312 scenes. We map the SceneVerse annotations to the 3DSSG vocabulary and construct $1,339$ sub-scenes. The model is trained exclusively on 3DSSG and evaluated on ScanNet without fine-tuning. Moreover, since GraphWrit3R predicts both objects and relations end-to-end, we also evaluate object detection separately on ScanNet [9], following the evaluation protocol of SpatialLM [39]. Further dataset construction details are provided in Sec. A.

Metrics. We evaluate scene graph prediction with standard top-K recall metrics [35]. Recall@K measures whether a ground-truth element is recovered among the top-K ranked predictions. Following prior scene graph protocols [59, 60, 72], we report node recall, predicate recall, and triplet recall. Node recall evaluates object labels, predicate recall evaluates relation labels for matched object pairs, and triplet recall evaluates the full subject–predicate–object prediction. For object detection, we report the $\mathrm{IoU}_{0.25}$ F1 metric used by SpatialLM [39], with more details discussed in Sec. A.

Implementation details. Most prior 3D scene graph methods assume ground-truth object segments or nodes as input during evaluation. In contrast, GraphWrit3R jointly detects all objects, classifies them, and predicts their relationships during scene graph generation. We therefore establish a one-to-one correspondence between predicted and ground-truth objects via Hungarian matching based on 3D IoU, in order to compute the metrics and evaluate our method. The full details of this matching procedure, as well as the metric computations, are thoroughly discussed in Sec. A. In addition, all architectural choices, hyperparameters, and training schedules are provided in Sec. A.

Baselines. We compare against current state-of-the-art open-vocabulary 3D scene graph methods. ConceptGraphs [13] constructs object-level 3D graphs from posed RGB-D sequences by detecting objects in 2D, associating them across views, and using large vision-language models to caption objects and infer inter-object relations. Open3DSG [26] operates on point-cloud object instances, distills 2D vision-language features into a 3D graph neural network, and uses an LLM to generate open-set relationship descriptions. RelationField [27] represents a scene as a radiance field and learns scene-specific open-vocabulary object and relationship feature fields by distilling relationship knowledge from multimodal LLMs. ReLaGS [71] operates on 3D Gaussian splats, constructing language-aligned Gaussian object representations and applying GNN-based relational reasoning to

Table 1: **Scene Graph Evaluation on RIO10.** Recall-based evaluation across object class, predicate, and relationship triplet prediction. Our GraphWrit3R achieves the best overall performance. Notably, RelationField, ReLaGS, and GEODE leverage *ground-truth object boxes* as input during inference, whereas our method must jointly detect all objects before predicting their classes and relationships. GraphWrit3R † denotes our variant supplied with ground-truth objects at inference. For a fair comparison, we report results using point cloud input only during inference, after multi-modal training. Experiments with different input modality configurations are provided in Table 3.

Model	Modality inference	GT-free inference	Runtime per scene	Object		Predicate		Triplet	
				R@5	R@10	R@3	R@5	R@50	R@100
Qwen3-VL-32B [56]	PCD	✓	~secs	0.26	0.31	0.58	0.59	0.42	0.45
GPT-5.4 [1]	PCD	✓	~secs	0.37	0.45	0.61	0.62	0.50	0.55
GPT-5.4 [1]	RGB	✓	~secs	0.34	0.42	0.74	0.75	0.61	0.65
GPT-5.4 [1]	PCD + RGB	✓	~secs	0.36	0.47	0.78	0.79	0.63	0.66
ConceptGraphs [13]	RGB-D	✓	~hours	0.37	0.46	0.74	0.79	0.69	0.71
Open3DSG [26]	RGB-D + PCD	✗	~mins	0.56	0.61	0.58	0.65	0.55	0.56
RelationField [27]	NeRF	✗	~hours	**0.69**	**0.80**	0.76	0.82	0.73	0.74
ReLaGS [71]	3DGS	✗	~mins	0.68	0.79	0.79	**0.87**	—	—
GEODE [10]	PCD	✗	~mins	—	—	0.79	0.83	**0.76**	0.80
GraphWrit3R †	PCD	✗	~secs	—	—	0.83	0.85	**0.76**	**0.92**
GraphWrit3R (ours)	PCD	✓	~secs	**0.69**	0.76	**0.85**	**0.87**	0.74	0.79

Table 2: **Scene Graph Evaluation on 3DSSG and ScanNet.** Recall across object class, predicate, and relationship triplet on the full 3DSSG and ScanNet evaluation sets. On ScanNet, both methods are evaluated zero-shot after training exclusively on 3DSSG under the same evaluation protocol. GraphWrit3R demonstrates stronger zero-shot predicate generalization. Our method achieves the best overall performance, confirming the findings from Tab. 1.

Model	Modality	Object		Predicate		Triplet	
		R@5	R@10	R@3	R@5	R@50	R@100
3DSSG (in-domain)							
Qwen3-VL-32B [56]	PCD	0.12	0.14	0.45	0.46	0.35	0.36
GPT-5.4	PCD	0.31	0.31	0.49	0.50	0.40	0.44
Open3DSG [26]	PCD+RGB-D	0.57	**0.68**	0.63	0.70	0.64	0.66
GraphWrit3R (ours)	PCD	**0.60**	0.64	**0.82**	**0.83**	**0.68**	**0.71**
ScanNet (zero-shot)							
Open3DSG [26]	PCD+RGB-D	0.70	**0.80**	0.51	0.53	0.71	**0.85**
GraphWrit3R (ours)	PCD	**0.74**	0.77	**0.90**	**0.92**	**0.73**	0.84

predict open-vocabulary relations. For each baseline, we follow its suggested evaluation protocol, which for most methods includes providing ground-truth object instances as input, a privilege not available to GraphWrit3R, which must first detect all objects. To isolate relational reasoning under the same setting, we additionally report GraphWrit3R †, a variant of our model in which the ground-truth object list is provided in the input prompt and the model predicts only the relationships between the given objects, without retraining. In addition, since our method relies on an LLM backbone, we also evaluate general-purpose vision-language models without task-specific fine-tuning, namely Qwen3-VL-32B [56] and GPT-5.4 [53]. Both are prompted to produce scene graphs in the same JSON format as GraphWrit3R, using input modality combinations common among recent methods. Further prompting details are provided in Sec. A.

4.2 Main results

The primary experimental results for 3D scene graph generation are presented in Tab. 1, evaluated on the RIO10 benchmark using the established recall-based evaluation across object class, predicate, and relationship triplet prediction. GraphWrit3R achieves the best overall performance, which is

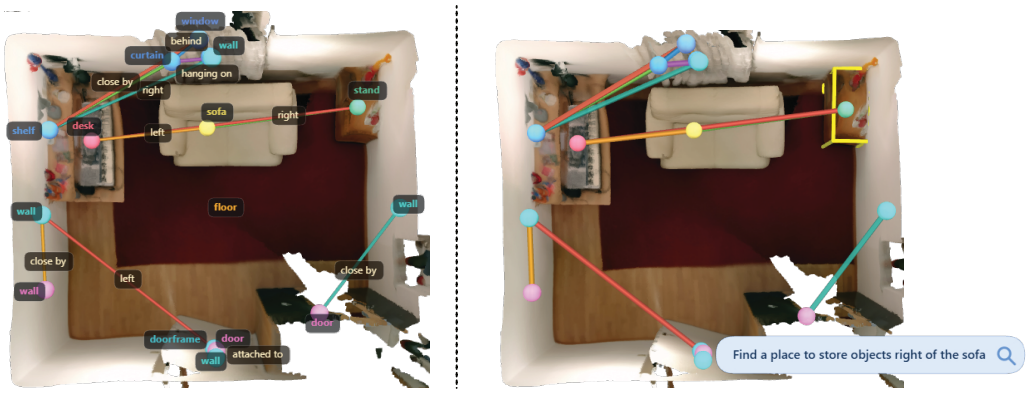

Figure 4: **Qualitative Scene Graph Generation and Open-Vocabulary Querying.** (a) Our GraphWrit3R generates the 3D scene graph directly from a raw point cloud input, with nodes representing detected objects and edges encoding their predicted semantic relationships. (b) The resulting graph supports open-vocabulary querying, where a natural-language query is matched against the predicted object and relationship labels to retrieve the corresponding elements within the 3D scene.

particularly notable given the more challenging operating conditions. Unlike the main competing approaches, GraphWrit3R does not leverage ground-truth objects and their boxes during inference. Therefore, it must jointly detect all objects as graph nodes before predicting their semantic classes and relationships. Object classification is therefore the greatest challenge for our method, since competing approaches only classify objects that are already given. Providing ground-truth objects to our model (GraphWrit3R †) leaves predicate recall roughly unchanged but substantially increases triplet recall, showing that object detection and classification are a major bottleneck for end-to-end triplet prediction. Every predicted relation is anchored to a predicted object, so a missed or misclassified object counts all of its relations as incorrect at the triplet level. For a fair comparison, we report results using point cloud input only during inference, after multi-modal training. Experiments with different input modality configurations are provided in Sec. 4.3. Beyond performance, our method operates as a fully end-to-end latent method that directly generates the scene graph as a structured JSON, resulting in substantially higher inference speed compared to other multi-stage pipeline alternatives. We also include GPT-5.4 in a frozen end-to-end fashion, as a reference point, evaluated by feeding it several combinations of popular modalities. Although it produces a valid scene graph, its performance falls well short of that of our specialized LLM, underscoring the value of task-specific adaptation.

Experiments on the full evaluation set of 157 3DSSG scenes, comprising 548 subsets, are reported in Tab. 2. Our method achieves the best overall performance, confirming the findings from Tab. 1. Due to reproducibility constraints and missing open-source implementations of baselines, this comparison is limited to baselines for which results were reported in the original publications or whose code is publicly available. Additionally, since SceneSplat++ [37] does not provide 3DGS reconstructions for a substantial portion of the 3DSSG dataset, this experiment uses only point cloud inputs during both training and evaluation. Moreover, Tab. 2 evaluates zero-shot domain shift, where both methods are trained exclusively on 3DSSG and evaluated on ScanNet. Our method retains strong performance under this domain shift, demonstrating that our framework generalizes well beyond its training dataset. Furthermore, Fig. 4 provides a qualitative illustration of a scene graph generated by GraphWrit3R on an evaluation scene, together with a demonstration of open-vocabulary querying over the resulting graph. Additional qualitative results with full JSON graph scripts can be found in Fig. 6.

4.3 Ablation and Analysis

Tab. 3 presents an evaluation of all inference-time input configurations of our GraphWrit3R following multi-modal training on both point clouds and 3DGS. Point-cloud-only inference achieves the strongest performance, which we attribute to the noticeably lower reconstruction quality of the 3DGS counterparts sourced from SceneSplat++ [37], introducing noise into the Gaussian Splat encoding branch and reinforcing modality bias [83, 82] toward the more informative point cloud signal. Further discussion and visual illustrations of these quality degradations are provided in Sec. B. This effect is most pronounced on object detection metrics, where 3DGS-only inference degrades the

Table 3: **Multi-Modal Analysis on RIO10.** Evaluation of all inference-time input configurations following multi-modal training. Point-cloud-only inference achieves the strongest performance, which we attribute to the noticeably lower reconstruction quality of the 3DGS counterparts (see Section B), reinforcing modality bias toward the more informative point cloud signal. A point-cloud-only trained variant performs similarly for the same reasons.

Training modality		Inference modality		Object		Predicate		Triplet	
PCD	3DGS	PCD	3DGS	R@5	R@10	R@3	R@5	R@50	R@100
✓	✓	✓	✓	0.65	0.71	0.84	0.85	0.72	0.77
✓	✓	×	✓	0.50	0.55	0.85	0.86	0.66	0.73
✓	✓	✓	×	0.69	0.76	0.85	0.87	0.74	0.79
✓	×	✓	×	0.70	0.73	0.85	0.88	0.75	0.80

most significantly, likely due to the difficulty of reliably detecting smaller objects under degraded reconstruction quality. A variant trained and evaluated exclusively on point clouds performs similarly, consistent with the same underlying factors. Nevertheless, we emphasize that multi-modal support is motivated by versatility rather than peak single-modality performance. In practice, different real-world capture pipelines produce different 3D representations, and training with multiple modalities from diverse sources mirrors this setting, enabling a single model with one set of weights to handle whichever representation is available at inference time.

Tab. 4 presents an ablation study over the contrastive loss and token fusion, as well as the individual alignment loss components. The results clearly demonstrate that the contrastive loss is critical, as next-token prediction loss alone is insufficient to align the Sonata and Chorus feature spaces for the scene graph generation task. Among the contrastive loss variants, the asymmetric formulation, which pulls Chorus representations toward those of Sonata rather than applying symmetric gradients on both sides, consistently outperforms its symmetric counterpart. This is expected, as Sonata is already well-aligned to the LLM through SpatialLM pretraining, making it the more stable anchor. The asymmetric loss consequently concentrates gradient signal on adapting Chorus to this established shared space. Regarding token fusion, we experimented with several mechanisms for combining spatially co-located features across modalities, finding that simple average pooling yields the best overall performance, suggesting that more complex fusion schemes introduce unnecessary optimization difficulty without a corresponding benefit. We further perform leave-one-out ablations of the InfoNCE, cosine similarity, and MSE terms in the alignment objective under the 3DGS-only input setting. Removing any loss term degrades results on nearly every metric, confirming that the three terms provide complementary supervision. InfoNCE has the largest overall impact, with its removal producing the largest drop on metrics, particularly affecting predicate and triplet recall. Cosine similarity is comparable and is slightly more influential for Object R@10, where its removal causes a larger decrease compared with InfoNCE, suggesting that it aids fine-grained object-level alignment. MSE contributes the least overall, leaving predicate recall and Triplet R@100 unchanged when removed, while still improving object-level recall. Furthermore, visualizations of encoder feature maps before and after contrastive alignment are provided in Fig. 11.

We additionally analyze reliability and scaling of autoregressive JSON generation. Across the complete 3DSSG evaluation set, strict JSON parsing succeeds in 100% of cases whenever the generated output fits within the model's context window. The only observed failure mode is hard output truncation for very large scenes, occurring roughly beyond 40 objects and 1,600 relationships due to the 8k-token context window inherited from the Qwen2.5-0.5B backbone, rather than malformed JSON or syntax collapse. This reveals output length as the primary scaling limitation of the current model. While larger backbones and context windows could improve scalability to denser scenes, we deliberately adopt the compact 0.5B backbone to target resource-constrained embodied robotic platforms. We discuss this trade-off and the resulting scaling limitations in more detail in Sec D.

4.4 Synergy Effects for Object Understanding

Object semantics are often influenced by relational context, so we investigate whether joint relational supervision in turn benefits object detection and understanding. We evaluate the point-cloud-only variant of GraphWrit3R on ScanNet20, with results reported in Tab. 5. Joint supervision on object

Table 4: **Ablation Study on RIO10.** (a) Ablation over contrastive loss formulation and token fusion. (b) Leave-one-out ablation of the individual alignment loss terms under the 3DGS-only input setting.

Contrastive loss	Fusion	Object		Predicate		Triplet	
		R@5	R@10	R@3	R@5	R@50	R@100
(a) Contrastive loss and fusion strategy							
Without	Concat.	0.60	0.64	0.80	0.81	0.65	0.74
Without	Average	0.62	0.70	0.80	0.81	0.68	0.75
Symmetric	Average	0.50	0.59	0.76	0.78	0.60	0.71
Asymmetric	Attention	0.62	0.70	0.82	0.83	0.68	0.76
Asymmetric	Gated	0.53	0.59	0.79	0.81	0.62	0.72
Asymmetric	Average	0.65	0.71	0.84	0.85	0.72	0.77
(b) Alignment loss design (3DGS-only input)							
Full-contrastive loss		0.50	0.55	0.85	0.86	0.66	0.73
No cosine loss		0.44	0.49	0.81	0.83	0.63	0.72
No InfoNCE		0.43	0.50	0.77	0.78	0.57	0.66
No MSE		0.45	0.54	0.85	0.86	0.62	0.73

Table 5: **Object Detection Evaluation on ScanNet20.** Our GraphWrit3R, jointly supervised on both object detection and relationship prediction, notably outperforms SpatialLM, demonstrating that relational supervision provides a useful learning signal for object grounding and localization.

Method	average	bathtub	bed	bookshelf	cabinet	chair	counter	curtain	desk	door	picture	refrige.	s.curtain	sink	sofa	table	toilet	window
V-DETR [51]	65.7	77.4	80.1	48.8	44.4	82.5	**55.6**	66.1	68.0	62.3	**47.4**	50.8	74.0	**79.5**	72.7	55.3	**98.0**	54.9
SceneScript [3]	49.5	64.5	71.1	39.7	30.3	81.1	43.4	30.9	53.4	41.7	11.8	28.2	58.6	48.9	68.3	55.8	77.5	36.5
SpatialLM [39]	66.2	80.6	79.9	52.0	40.0	**86.6**	51.0	66.8	62.8	**67.1**	29.7	**53.6**	**81.6**	70.7	78.9	63.9	95.4	64.3
GraphWrit3R (ours)	**68.9**	**88.5**	**88.5**	**56.7**	**52.4**	85.7	47.6	**68.0**	**70.4**	65.5	33.7	53.5	70.8	73.5	**82.5**	**68.3**	96.3	**69.3**

detection and relational prediction yields measurably stronger detection than object supervision alone, with a clear margin over SpatialLM. This suggests that relational context provides an inductive signal for grounding and localizing individual objects, producing more discriminative and spatially precise representations than detection supervision alone.

5 Conclusion

We presented GraphWrit3R, an end-to-end method that accepts a 3D point cloud, or Gaussian Splats, or both, and directly outputs a complete scene graph in structured JSON format. Unlike prior work, it requires no explicit intermediate representations, no ground-truth object annotations at inference time, and no proprietary model usage. Motivated purely by versatility, GraphWrit3R is trained with multi-modal inputs but can be deployed with any single modality at inference, allowing one set of weights to handle diverse inputs. On the 3DSSG benchmark, it achieves state-of-the-art performance on object, predicate, and triplet recall, outperforming methods that assume ground-truth annotations, and our ablations and additional analysis validate each key design choice. Furthermore, we see broader opportunities beyond pure scene graph generation. Structured relational outputs in text format could ground 3D vision-language models more deeply in physical space, pushing them beyond text-in text-out visual question answering toward richer and more reliable spatially grounded reasoning. Such spatially intelligent systems, in turn, are a critical building block for the next wave of embodied intelligence, enabling autonomous robots to reason about object relations and plan actions, and augmented reality systems to anchor virtual content coherently in the physical world.

Acknowledgments and Disclosure of Funding

This research was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure). Sayan Deb Sarkar is partly supported by the Stanford Doerr School of Sustainability.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 4, 7

[2] 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*, pages 5664–5673, 2019. 2, 3

[3] Armen Avetisyan, Christopher Xie, Henry Howard-Jenkins, Tsun-Yi Yang, Samir Aroudj, Suvam Patra, Fuyang Zhang, Duncan Frost, Luke Holland, Campbell Orme, Jakob Engel, Edward Miller, Richard Newcombe, and Vasileios Balntas. Scenescript: Reconstructing scenes with an autoregressive structured language model. In *European Conference on Computer Vision*, 2024. 4, 10

[4] Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, et al. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv preprint arXiv:2504.13181*, 2025. 5

[5] Sijin Chen, Xin Chen, Chi Zhang, Mingsheng Li, Gang Yu, Hao Fei, Hongyuan Zhu, Jiayuan Fan, and Tao Chen. Ll3da: Visual interactive instruction tuning for omni-3d understanding, reasoning, and planning, 2023. 4

[6] Sijin Chen, Hongyuan Zhu, Xin Chen, Yinjie Lei, Gang Yu, and Tao Chen. End-to-end 3d dense captioning with vote2cap-detr. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11124–11133, 2023. 4

[7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 3

[8] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 4

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017. 6, 22

[10] Changqun Feng, Wangxiandi Yin, Xin Hu, Lei Zhao, Dongyang Zhang, and Tao He. GEODE: Geometry-guided discrete diffusion for open-vocabulary 3d scene graph generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR) Findings*, pages 7143–7153, June 2026. 4, 7

[11] Rao Fu, Jingyu Liu, Xilun Chen, Yixin Nie, and Wenhan Xiong. Scene-llm: Extending language model for 3d visual reasoning. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 2195–2206, February 2025. 4

[12] Zhengyu Fu, René Zurbrügg, Kaixian Qu, Marc Pollefeys, Marco Hutter, Hermann Blum, and Zuria Bauer. Funfact: Building probabilistic functional 3d scene graphs via factor-graph reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2026. 2, 22

[13] Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5021–5028. IEEE, 2024. 2, 3, 4, 6, 7, 22

[14] Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *arXiv preprint arXiv:2310.19923*, 2023. 17

[15] Anna-Maria Halacheva, Jan-Nico Zaech, Xi Wang, Danda Pani Paudel, and Luc Van Gool. Gaussianvlm: Scene-centric 3d vision-language models using language-aligned gaussian splats for embodied reasoning and beyond. *IEEE Robotics Autom. Lett.*, 10(12):12588–12595, 2025. 4

[16] Haifeng Huang, Yilun Chen, Zehan Wang, Rongjie Huang, Runsen Xu, Tai Wang, Luping Liu, Xize Cheng, Yang Zhao, Jiangmiao Pang, et al. Chat-scene: Bridging 3d scene and large language models with object identifiers. *Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada*, 2024. 4

[17] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024. 4

[18] Kaixiang Huang, Qifeng Zhang, Jin Wang, Jingru Yang, Yang Zhou, Jiao Yi, Guodong Lu, and Shengfeng He. GranSSG: Correlating volumetric granularities for 3d semantic scene graph prediction. *IEEE Transactions on Visualization and Computer Graphics*, 32(7):6548–6565, 2026. 3

[19] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization. *arXiv preprint arXiv:2201.13360*, 2022. 2, 3

[20] Baoxiong Jia, Yixin Chen, Huangyue Yu, Yan Wang, Xuesong Niu, Tengyu Liu, Qing Li, and Siyuan Huang. Sceneverse: Scaling 3d vision-language learning for grounded scene understanding. In *European Conference on Computer Vision (ECCV)*, 2024. 6, 18, 22

[21] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 2, 5

[23] Karim Knaebel, Kadir Yilmaz, Daan de Geus, Alexander Hermans, David Adrian, Timm Linder, and Bastian Leibe. DINO in the room: Leveraging 2D foundation models for 3D segmentation. In *2026 International Conference on 3D Vision (3DV)*, 2026. 3

[24] Sebastian Koch, Pedro Hermosilla, Narunas Vaskevicius, Mirco Colosi, and Timo Ropinski. Lang3dsg: Language-based contrastive pre-training for 3d scene graph prediction. In *2024 International Conference on 3D Vision (3DV)*, pages 1037–1047. IEEE, 2024. 2, 3

[25] Sebastian Koch, Pedro Hermosilla, Narunas Vaskevicius, Mirco Colosi, and Timo 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*, pages 3404–3414, 2024. 2, 3

[26] Sebastian Koch, Narunas Vaskevicius, Mirco Colosi, Pedro Hermosilla, and Timo 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*, pages 14183–14193, 2024. 2, 3, 4, 6, 7, 17, 22

[27] Sebastian Koch, Johanna Wald, Mirco Colosi, Narunas Vaskevicius, Pedro Hermosilla, Federico Tombari, and Timo Ropinski. Relationfield: Relate anything in radiance fields. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21706–21716, 2025. 2, 4, 6, 7, 22

[28] Sebastian Koch, Johanna Wald, Hidenobu Matsuki, Pedro Hermosilla, Timo Ropinski, and Federico Tombari. Unified semantic transformer for 3d scene understanding. *arXiv preprint arXiv:2512.14364*, 2025. 3

[29] Justin Lazarow, David Griffiths, Gefen Kohavi, Francisco Crespo, and Afshin Dehghan. Cubify anything: Scaling indoor 3d object detection. *arXiv preprint arXiv:2412.04458*, 2024. 2, 3

[30] Junha Lee, Chunghyun Park, Jaesung Choe, Yu-Chiang Frank Wang, Jan Kautz, Minsu Cho, and Chris Choy. Mosaic3d: Foundation dataset and model for open-vocabulary 3d segmentation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14089–14101, 2025. 2, 3

[31] Yue Li, Qi Ma, Runyi Yang, Huapeng Li, Mengjiao Ma, Bin Ren, Nikola Popovic, Nicu Sebe, Ender Konukoglu, Theo Gevers, Luc Van Gool, Martin R. Oswald, and Danda Pani Paudel. Scenesplat: Gaussian splatting-based scene understanding with vision-language pretraining, 2025. 2, 3, 23

[32] Yue Li, Qi Ma, Runyi Yang, Mengjiao Ma, Bin Ren, Nikola Popovic, Nicu Sebe, Theo Gevers, Luc Van Gool, Danda Pani Paudel, and Martin R. Oswald. Chorus: Multi-teacher pretraining for holistic 3d gaussian scene encoding, 2025. 2, 3, 4, 5, 19, 23

[33] Zeju Li, Chao Zhang, Xiaoyan Wang, Ruilong Ren, Yifan Xu, Ruifei Ma, and Xiangde Liu. 3dmit: 3d multi-modal instruction tuning for scene understanding. *arXiv preprint arXiv:2401.03201*, 2024. 4

[34] Sergey Linok, Tatiana Zemskova, Svetlana Ladanova, Roman Titkov, Dmitry Yudin, Maxim Monastyrny, and Aleksei Valenkov. Beyond bare queries: Open-vocabulary object grounding with 3d scene graph. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13582–13589. IEEE, 2025. 2, 3, 4, 22

[35] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *Computer Vision – ECCV 2016*, pages 852–869, Cham, 2016. Springer International Publishing. 6

[36] Shiyang Lu, Haonan Chang, Eric Pu Jing, Abdeslam Boularias, and Kostas Bekris. Ovir-3d: Open-vocabulary 3d instance retrieval without training on 3d data. In *7th Annual Conference on Robot Learning*, 2023. 3

[37] Mengjiao Ma, Qi Ma, Yue Li, Jiahuan Cheng, Runyi Yang, Bin Ren, Nikola Popovic, Mingqiang Wei, Nicu Sebe, Luc Van Gool, Theo Gevers, Martin R. Oswald, and Danda Pani Paudel. Scenesplat++: A large dataset and comprehensive benchmark for language gaussian splatting. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS 2025)*, 2025. 3, 6, 8, 19, 23

[38] Yanni Ma, Hao Liu, Yulan Guo, Theo Gevers, and Martin R. Oswald. Edge-centric relational reasoning for 3d scene graph prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 7847–7855, 2026. 3

[39] Yongsen Mao, Junhao Zhong, Chuan Fang, Jia Zheng, Rui Tang, Hao Zhu, Ping Tan, and Zihan Zhou. Spatiallm: Training large language models for structured indoor modeling. In *Advances in Neural Information Processing Systems*, 2025. 2, 4, 5, 6, 10, 17, 18, 19, 23

[40] Phuc D. A. Nguyen, Tuan Duc Ngo, Evangelos Kalogerakis, Chuang Gan, Anh Tran, Cuong Pham, and Khoi Nguyen. Open3dis: Open-vocabulary 3d instance segmentation with 2d mask guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3

[41] Songyou Peng, Kyle Genova, Chiyu "Max" Jiang, Andrea Tagliasacchi, Marc Pollefeys, and Thomas Funkhouser. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2023. 3

[42] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 3

[43] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 3

[44] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, 2017. 3

[45] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics. 17

[46] Dennis Rotondi, Fabio Scaparro, Hermann Blum, and Kai O Arras. Fungraph: Functionality aware 3d scene graphs for language-prompted scene interaction. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4083–4090. IEEE, 2025. 2, 4, 22

[47] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Fcaf3d: fully convolutional anchor-free 3d object detection. In *European Conference on Computer Vision*, pages 477–493. Springer, 2022. 2, 3

[48] Evgenia Rusak, Patrik Reizinger, Attila Juhos, Oliver Bringmann, Roland S. Zimmermann, and Wieland Brendel. InfoNCE: Identifying the gap between theory and practice. *arXiv preprint arXiv:2407.00143*, 2024. 5

[49] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D: Mask Transformer for 3D Semantic Instance Segmentation. In *International Conference on Robotics and Automation (ICRA)*, 2023. 2

[50] Yichao Shen, Zigang Geng, Yuhui Yuan, Yutong Lin, Ze Liu, Chunyu Wang, Han Hu, Nanning Zheng, and Baining Guo. V-detr: Detr with vertex relative position encoding for 3d object detection, 2023. 2, 3

[51] Yichao Shen, Zigang Geng, Yuhui Yuan, Yutong Lin, Ze Liu, Chunyu Wang, Han Hu, Nanning Zheng, and Baining Guo. V-detr: Detr with vertex relative position encoding for 3d object detection. In *International Conference on Learning Representations*, 2024. 10

[52] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025. 5

[53] Aaditya Singh et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025. 7

[54] Ayça Takmaz, Elisabetta Fedele, Robert W. Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. OpenMask3D: Open-Vocabulary 3D Instance Segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3

[55] Qwen team. Qwen2.5 technical report, 2025. 2, 19, 23

[56] Qwen team. Qwen3-vl technical report, 2025. 7

[57] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025. 5

[58] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. Rio: 3d object instance re-localization in changing indoor environments. In *Proceedings IEEE International Conference on Computer Vision (ICCV)*, 2019. 22

[59] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3961–3970, 2020. 2, 3, 6, 22

[60] Johanna Wald, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs with instance embeddings. *International Journal of Computer Vision*, 130(3):630–651, 2022. 6

[61] Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, , and Justin M. Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *The Conference on Robot Learning (CoRL)*, 2021. 3

[62] Yue Wang and Justin M. Solomon. Object dgcnn: 3d object detection using dynamic graphs. In *2021 Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 3

[63] Zehan Wang, Haifeng Huang, Yang Zhao, Ziang Zhang, and Zhou Zhao. Chat-3d: Data-efficiently tuning large language model for universal dialogue of 3d scenes. *arXiv preprint arXiv:2308.08769*, 2023. 4

[64] Ziqin Wang, Bowen Cheng, Lichen Zhao, Dong Xu, Yang Tang, and Lu Sheng. Vl-sat: Visual-linguistic semantics assisted training for 3d semantic scene graph prediction in point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21560–21569, 2023. 3

[65] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical Open-Vocabulary 3D Scene Graphs for Language-Grounded Robot Navigation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. 4

[66] 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*, pages 5064–5074, 2023. 3

[67] 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*, pages 7515–7525, 2021. 3, 18

[68] Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *CVPR*, 2025. 2, 3, 4, 5, 19, 23

[69] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *CVPR*, 2024. 2, 3, 4, 23

[70] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *NeurIPS*, 2022. 3

[71] Yaxu Xie, Abdalla Arafa, Alireza Javanmardi, Christen Millerdurai, Jia Cheng Hu, Shaoxiang Wang, Alain Pagani, and Didier Stricker. Relags: Relational language gaussian splatting. *arXiv preprint arXiv:2603.17605*, 2026. 2, 4, 6, 7, 22

[72] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 6

[73] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding, 2023. 3

[74] Yingda Yin, Yuzheng Liu, Yang Xiao, Daniel Cohen-Or, Jingwei Huang, and Baoquan Chen. Sai3d: Segment any instance in 3d scenes, 2024. 2

[75] Fei Yu, Quan Deng, Shengeng Tang, Yuehua Li, and Lechao Cheng. Open-world 3d scene graph generation for retrieval-augmented reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 3894–3902, 2026. 4

[76] Hanxun Yu, Wentong Li, Song Wang, Junbo Chen, and Jianke Zhu. Inst3d-lmm: Instance-aware 3d scene understanding with multi-modal instruction tuning. *arXiv preprint arXiv:2503.00513*, 2025. 4

[77] Chaoyi Zhang, Jianhui Yu, Yang Song, and Weidong 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*, pages 9705–9715, 2021. 2, 3

[78] Chenyangguang Zhang, Alexandros Delitzas, Fangjinhua Wang, Ruida Zhang, Xiangyang Ji, Marc Pollefeys, and Francis Engelmann. Open-vocabulary functional 3d scene graphs for real-world indoor spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19401–19413, 2025. 2, 4, 22

[79] Yujia Zhang, Xiaoyang Wu, Yixing Lao, Chengyao Wang, Zhuotao Tian, Naiyan Wang, and Hengshuang Zhao. Concerto: Joint 2d-3d self-supervised learning emerges spatial representations. In *NeurIPS*, 2025. 2, 3

[80] Yujia Zhang, Xiaoyang Wu, Yunhan Yang, Xianzhe Fan, Han Li, Yuechen Zhang, Zehao Huang, Naiyan Wang, and Hengshuang Zhao. Utonia: Toward one encoder for all point clouds, 2026. 2, 3

[81] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021. 3

[82] Xu Zheng, Chenfei Liao, Yuqian Fu, Kaiyu Lei, Yuanhuiyi Lyu, Lutao Jiang, Bin Ren, Jialei Chen, Jiawen Wang, Chengxin Li, Linfeng Zhang, Danda Pani Paudel, Xuanjing Huang, Yu-Gang Jiang, Nicu Sebe, Dacheng Tao, Luc Van Gool, and Xuming Hu. Mllms are deeply affected by modality bias, 2025. 8

[83] Xu Zheng, Yuanhuiyi Lyu, Lutao Jiang, Danda Pani Paudel, Luc Van Gool, and Xuming Hu. Reducing unimodal bias in multi-modal semantic segmentation with multi-scale functional entropy regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 8

[84] Hongyan Zhi, Peihao Chen, Junyan Li, Shuailei Ma, Xinyu Sun, Tianhang Xiang, Yinjie Lei, Mingkui Tan, and Chuang Gan. Lscenellm: Enhancing large 3d scene understanding using adaptive visual preferences. *arXiv preprint arXiv:2412.01292*, 2024. 4

[85] Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. In *International Conference on Learning Representations (ICLR)*, 2024. 4

A Additional Implementation Details

End-to-end evaluation protocol. Most prior 3D scene graph methods assume ground-truth object segments or ground-truth object nodes as input, so predicted labels are already defined over the correct object set. In contrast, GraphWrit3R predicts the complete scene graph end-to-end, including both objects and relations. We therefore first establish a one-to-one correspondence between predicted and ground-truth objects using Hungarian matching based on 3D IoU. For rare cases in which a ground-truth object remains unmatched because all remaining predicted objects have zero IoU with it, we use centroid distance as a metric instead of 3D IoU for Hungarian matching. Ground-truth objects that remain unmatched are assigned rank ∞ and therefore cannot contribute to recall at any finite K. Since GraphWrit3R decodes free-form text labels rather than closed-set class logits, we compute ranks by semantic comparison against the evaluation vocabulary, following [26]. For node recall, the predicted object label is compared to all object classes in the 3DSSG vocabulary using Jina [14] embeddings. For predicate recall, predicted relation labels between matched object pairs are ranked using SBERT embeddings [45]. Triplet recall is stricter: a ground-truth triplet is recovered only if both endpoint objects are matched and the subject, predicate, and object labels are semantically recovered within the top-K ranking.

Scene Graph Prompting Format. The decoder is prompted to generate the full scene graph directly as JSON. The requested response follows the template shown in Figure 5. Following SpatialLM [39], we represent spatial quantities as integer tokens in the language sequence. Coordinates are quantized into $1{,}280$ bins with 2.5cm resolution, and the LLM predicts these discrete values as part of the JSON output. During post-processing, the predicted integers are mapped back to continuous metric values using the inverse quantization and normalization transform, restoring positions and box sizes to the original scene coordinate system. This formulation lets the model express objects, geometry, and relations within a single language-decoding interface. Figure 6 qualitatively shows the output from GraphWrit3R given the prompt and the input 3D scene.

```
{
  "num_objects": ⟨int⟩,
  "objects": [
    {
      "id": ⟨int⟩,
      "label": ⟨str⟩,
      "position": [⟨int⟩, ⟨int⟩, ⟨int⟩],
      "size": [⟨int⟩, ⟨int⟩, ⟨int⟩],
      "angle_z": ⟨int⟩
    }
  ],
  "relationships": [
    [⟨subj⟩, ⟨obj⟩, ⟨predicate⟩]
  ]
}
```

Figure 5: **Scene Graph JSON Template.** GraphWrit3R decodes object attributes and pairwise relations as a structured JSON scene graph.

VLM Baseline Implementation Details. We implement the zero-shot VLM baselines using a common prompting and parsing pipeline. For Qwen, we run Qwen3-VL-32B-Instruct locally on a single NVIDIA H200 GPU and evaluate it with the colored PLY point-cloud input. For GPT-5.4, we obtain predictions through API calls, which allows us to evaluate longer multi-modal prompts containing both the serialized point cloud and selected RGB views. Each validation scene is converted into a prompt asking the model to produce a single JSON scene graph with the same template as GraphWrit3R. We evaluate several input variants: a compact text serialization of the colored PLY point cloud, RGB views and combined PLY+RGB inputs. For the RGB variants, we select up to 12 views per scene from the parent 3RScan sequence. To choose informative views, we project the split point-cloud centroid and bounding-box corners into the available camera frames, score candidate views by how well they cover the target scene region, and greedily select a diverse set of high-coverage views. The PLY input provides a compact geometric representation of the scene, serialized as metric 3D coordinates with RGB values.

ScanNet Object Detection Comparison. To make the object-detection comparison with SpatialLM controlled at the evaluation level, we follow the SpatialLM ScanNet protocol and evaluate only the decoded object nodes, i.e., class labels and 3D boxes, while ignoring predicted relationships at test time. Starting from the same SpatialLM checkpoint, we fine-tune GraphWrit3R on SceneVerse

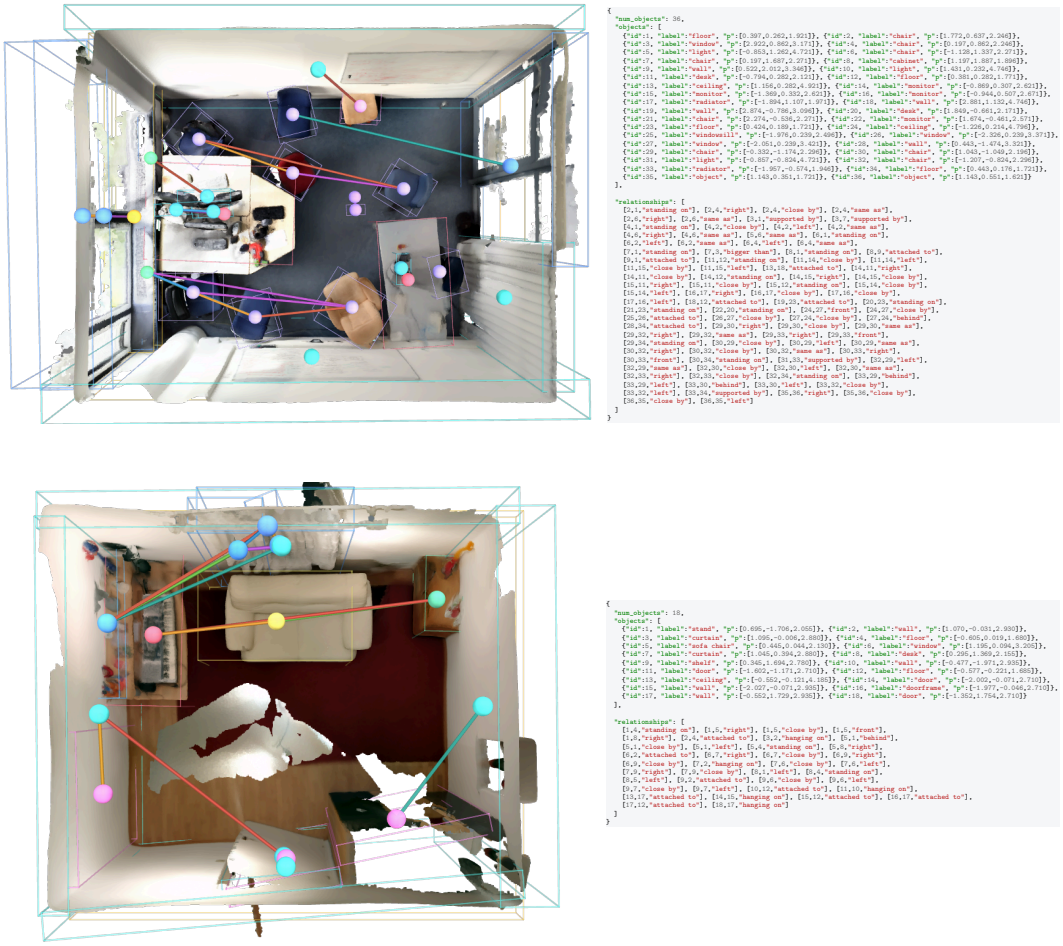

Figure 6: **GraphWrit3R Scene Graph Prediction.** We show results of GraphWrit3R scene graph prediction on two scenes, given the input 3D scene and the text prompt. Each row shows an input 3D scene with predicted object boxes and relations on the left, and the corresponding generated JSON scene graph on the right.

using their scene-graph annotations and the same JSON decoding format as in the main scene-graph task. This fine-tuning exposes the model to complete scene-graph supervision, including both object and relationship tokens, but the ScanNet metric is computed solely from the object portion of the output. Therefore, Tab. 5 should be interpreted as measuring whether our end-to-end scene-graph fine-tuning preserves or improves object detection under the SpatialLM evaluation protocol, rather than as a causal ablation isolating relational supervision from all other factors such as modality mixture, Chorus parameters, or the auxiliary alignment loss. During the evaluation of our model in this setup, we follow the evaluation approach from [39].

ScanNet Dataset Construction. We map the SceneVerse [20] object and relationship annotations to the 3DSSG vocabulary using normalized label matching and manually defined aliases, following the approach from [67]. Because complete ScanNet scenes generally contain more objects than the 3DSSG training examples, we divide them into local sub-scenes containing five to nine objects. The largest floor instance is included in every sub-scene as a common spatial reference, while each remaining object is assigned to exactly one sub-scene. Objects are grouped according to spatial proximity and neighborhood connectivity, where two objects are considered neighbors if their axis-aligned bounding boxes overlap after expansion by $0.5\,\mathrm{m}$. Using a fixed random seed for reproducibility, we grow each group by adding the neighboring object closest to its centroid in the horizontal plane. When no unassigned neighbor remains, we instead add the closest unassigned object. Each resulting point cloud contains only the selected object instances, and its graph retains

relationships whose endpoints are both present. This procedure produces $1,339$ sub-scenes from the 312 ScanNet validation scenes.

Architecture Details. We use Qwen2.5-0.5B [55] as the language decoder, Sonata [68] for point-cloud encoding, and Chorus [32] for Gaussian-splat encoding. Per-voxel 3D features are projected into the language-model embedding space using a lightweight MLP, following SpatialLM [39]. The Sonata encoder, projector, and language decoder are initialized from the released SpatialLM checkpoint. We fully fine-tune the 3D encoders, projector, and decoder, and use modality dropout during training so that a single model supports point-cloud-only, Gaussian-only, and fused inference. The Qwen2.5-0.5B language decoder has hidden size $h = 896$. The Sonata point-cloud encoder uses a voxel size of 2.5cm and produces per-voxel features of dimension $d = 512$. The Chorus Gaussian-splat encoder uses the same voxel size and output feature dimension as Sonata. Per-voxel features from Sonata and Chorus are projected into the Qwen2.5-0.5B embedding space using a per-token MLP with dimensions $d \to d \to h$, following SpatialLM [39].

Optimization Details. We fully fine-tune the 3D encoders, projector, and LLM decoder for 5 epochs. We use AdamW with learning rate 2×10^{-5}, an effective batch size of $\mathcal{B} = 8$, and a cosine learning-rate schedule with a 3% warmup ratio, following SpatialLM [39]. For multimodal robustness, we apply modality dropout with dropout probabilities of 35% for the point-cloud modality and 15% for the 3DGS modality. We use a two-stage loss schedule for cross-modal alignment. We employ a two-stage training schedule in which the auxiliary cross-modal alignment loss is applied with weight $\alpha = 0.25$ during the first half of the training and disabled afterwards. For the InfoNCE loss, we set the temperature to $\tau = 0.07$. Training is conducted on one NVIDIA H200 GPU for approximately 4 hours.

Compute Resources. All experiments were run on an on-premises institutional GPU cluster. Each training run used a single bare-metal node with one NVIDIA H200 GPU with 143GB of GPU memory, Intel Xeon Gold 6548Y+ CPUs, and NVMe SSD storage. Each run requested 8 CPU cores, 80GB of host memory, and one H200 GPU, with a maximum wall-clock time of 6 hours. A full fine-tuning run for GraphWrit3R takes approximately 4 hours. All reported training configurations use the same compute allocation. Across the reported training runs and ablations, the total compute is approximately 1000 H200 GPU-hours. Inference and evaluation are lightweight relative to training and are run on the same cluster allocation. The full research project required additional exploratory compute for debugging, preliminary experiments, and ablations beyond the final reported runs; these exploratory runs were not required to reproduce the final experimental results.

B Additional Experimental Analysis

3DGS Reconstruction Quality Table 3 shows that point-cloud-only inference achieves the strongest performance after multi-modal training. We attribute this behavior primarily to the reconstruction quality of the available Gaussian splats. 3D Gaussian Splatting reconstructions for 3DSSG and RIO10 were taken from [37] and contain substantial artifacts, including blurred geometry, floaters, missing structure, and strong view-dependent distortions. This is depicted in Figure 7. As a result, the Gaussian-splat encoder often receives a noisier and less spatially reliable signal than the point-cloud encoder, biasing the model toward the point-cloud modality. This also explains why a point-cloud-only trained variant performs similarly, since in this setting, the additional Gaussian-splat input often does not provide sufficiently reliable complementary information.

Embodied Planning Using 3D Scene Graphs We qualitatively demonstrate how the generated scene graph can serve as an explicit grounding representation for embodied planning in 3D scenes. Given a natural-language instruction and the predicted graph, we query GPT-5.4 to identify relevant objects, reason over their semantic relationships, and ground the resulting plan back into metric 3D space. Figure 10 shows an example in which a robot is instructed to navigate from the right side of the room and open the window closest to the desk with the most monitors. The planning model first identifies the monitor-rich workspace by aggregating relations between desks and nearby monitors, then selects the nearest feasible window among the predicted window nodes. It then uses the predicted object geometry to avoid large obstacles and identifies two movable chairs that obstruct the passage. This provides a preliminary downstream probe of the embodied-intelligence use cases discussed in the main paper: structured relational outputs can support spatial reasoning and action sequencing.



Figure 7: **Examples of degraded 3DGS reconstructions on RIO10.** Most 3D Gaussian splat reconstructions in RIO10 exhibit substantial artifacts, including blurred geometry, floaters, missing structure, and distortions.

However, this example is intended only as a qualitative demonstration of how predicted scene graphs may support embodied planning, not as a full solution to robotic navigation and manipulation. A more detailed excerpt from the GPT-5.4 planning response is shown in Figure 9. This example illustrates how the explicit scene-graph representation enables reasoning beyond direct object detection. The model output is not only a list of objects, but a structured representation that supports compositional spatial queries such as counting monitors near desks, comparing candidate windows by distance, and producing a grounded sequence of navigation and manipulation steps. We emphasize that this is a qualitative downstream demonstration rather than a separately trained robotics system. The trajectory and chair-relocation actions are derived from the predicted graph, object locations, and specified robot constraints.

Effect of Auxiliary Loss on Per-Voxel Feature Alignment Figure 11 illustrates the effect of the auxiliary cross-modal alignment loss on voxel-level representations. We visualize features at the final encoded voxel resolution, where Sonata, Chorus, and fused features are associated with corresponding voxel locations. For feature visualization, we compute a PCA basis from the pooled Sonata, Chorus, and fused voxel features from the same scene. We then map the first three projected coordinates to RGB using normalization across all three feature maps. We additionally report per-voxel cosine similarity between aligned feature vectors, with red denoting higher similarity. Before training, the Sonata and Chorus features are weakly aligned. After training, the modality-specific features become more consistent, suggesting that the auxiliary loss encourages co-located point-cloud and Gaussian splat tokens to align in the shared voxel space.

C Comparison to Relevant Methods

In Table 6, we provide an approximate comparison of model size, trainable parameters, and per-scene runtime across open-vocabulary 3D scene graph methods. Since existing methods differ substantially in their use of frozen backbones, external APIs, per-scene optimization, and multi-stage preprocessing, the numbers are intended as order-of-magnitude estimates rather than strict benchmarks. We further provide a high-level comparison between prior open-vocabulary scene graph methods, as shown in Figure 8.

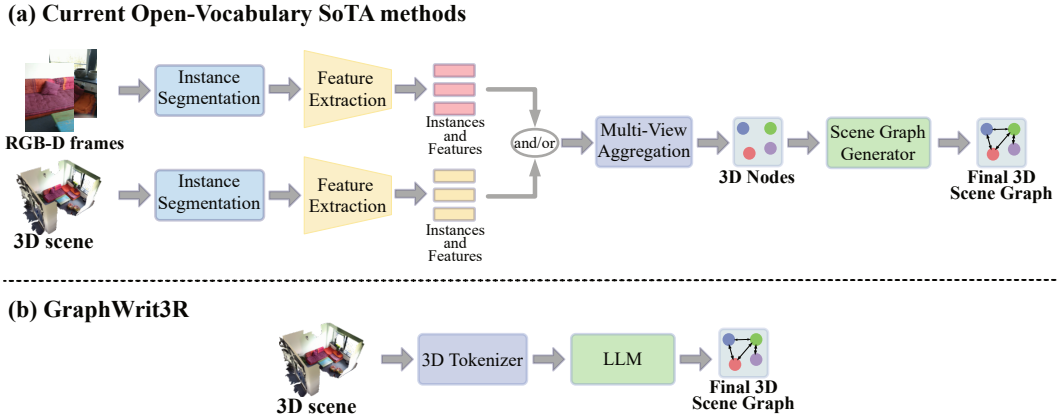

Figure 8: **Comparison of Open-Vocabulary 3D Scene Graph Generation Pipelines.** (a) Recent state-of-the-art approaches typically rely on multi-stage pipelines: 3D scenes and/or RGB-D frames are first processed by instance segmentation and feature extraction modules, followed by feature aggregation and scene graph generation. (b) In contrast, GraphWrit3R directly tokenizes the input 3D scene and uses an LLM to generate the final 3D scene graph in a single forward pass, avoiding explicit intermediate processing stages.

Parameter Count. We define total number of parameteres as the inference-time model footprint required by the method, including frozen foundation models that must be loaded or queried. Trainable parameters denote the number of parameters in the reported training setup. For API-based methods, we report zero trainable parameters, since no method-specific parameters are optimized locally, while the total parameter count reflects the approximate scale of the proprietary model family used during inference.

For GraphWrit3R, the parameter count includes the Qwen2.5-0.5B decoder, the Sonata point-cloud encoder, the Chorus Gaussian-splat encoder, the SpatialLM projection modules, and the additional fusion parameters. This gives a total of approximately 736.6M parameters, all of which are trainable in our setup.

Runtime. Runtime is reported at the granularity of seconds, minutes, or hours per scene. Methods based on per-scene optimization, repeated VLM/LLM querying, or dense 2D–3D association typically incur minute to hour-scale inference costs. In contrast, GraphWrit3R performs end-to-end scene graph prediction, yielding second-scale inference in our implementation.

Input Assumptions. We separately indicate whether a method requires ground-truth object nodes or object proposals as input during evaluation. This distinction is important because many scene graph methods evaluate relation prediction conditioned on known objects, whereas GraphWrit3R predicts both objects and relations without ground-truth nodes at inference time. We also mark methods that rely on proprietary models, since such dependencies affect reproducibility, deployment cost, and the feasibility of fully local evaluation.

D Limitations

GraphWrit3R has several limitations. First, our experiments focus on indoor reconstructed scenes, primarily 3DSSG for scene graph prediction and ScanNet for object detection, so robustness to outdoor, dynamic, and building-scale scenes remains untested. Second, the multi-modal setting is limited by the quality and availability of Gaussian-splat reconstruction. In our experiments, point-cloud-only inference performs best, while 3DGS inputs are affected by noisy or incomplete reconstructions. Third, our scene-graph experiments inherit a broader limitation of the 3DSSG benchmark protocol. Existing methods are commonly trained and evaluated on small annotated subgraphs, often containing only around nine object nodes, rather than on complete scene-level graphs. This makes the benchmark useful for controlled comparison, but it also means that reported results may underestimate the difficulty of full-scene graph generation, long-range relational reasoning, and scaling to cluttered environments with many objects. This is also relevant to our autoregressive JSON

Table 6: **Computational Comparison of Open-Vocabulary 3D Scene Graph Methods.** We report approximate inference-time model footprint, trainable parameters, per-scene runtime, input assumptions, and reliance on proprietary models. Among the compared methods, GraphWrit3R is the only fully local approach that simultaneously avoids ground-truth object nodes as inputs at evaluation time and operates at second-scale runtime.

Method	Total Params	Trainable Params	Runtime / Scene	No GT nodes as inputs	Uses proprietary models
GraphWrit3R (ours)	$\approx$ 736.6M	$\approx$ 736.6M	$\sim$seconds	✓	×
RelationField [27]	$\approx$ 200M	$\approx$ 200M	$\sim$hours	×	×
Open3DSG [26]	$\sim$ 8B	$\approx$ 111M	$\sim$minutes	×	×
ReLaGS [71]	$\sim$ 1.8B	$\approx$ 8.2M	$\sim$minutes	×	×
Beyond Bare Queries [34]	$\sim$ 1T	0	$\sim$hours	×	✓
ConceptGraphs [13]	$\sim$ 1T	0	$\sim$hours	✓	✓
OpenFunGraph [78]	$\sim$ 1T	0	$\sim$hours	✓	✓
FunFact[12]	$\sim$ 1T	0	$\sim$hours	✓	✓
FunGraph [46]	$\sim$ 1T	0	$\sim$hours	✓	✓

formulation. In full-scene stress tests, strict JSON parsing succeeds in 100% of cases whenever the output fits within the model's 8k-token context window. The only observed failure mode is hard truncation for very large scenes, roughly beyond 40 objects and $1,600$ relationships. Thus, scaling is currently limited primarily by output length rather than JSON syntax reliability. Larger backbones or context windows could alleviate this bottleneck, at the cost of increased computational requirements. Finally, because GraphWrit3R predicts objects and relations end-to-end, evaluation requires Hungarian object matching and semantic text-embedding-based ranking of free-form labels. The resulting scores may therefore depend on these protocol choices.

E Broader Societal Impact

GraphWrit3R aims to improve structured 3D scene understanding by producing explicit object-and-relation scene graphs from point clouds or Gaussian splats. This can benefit applications that require spatially grounded reasoning, such as robotics, augmented reality, assistive navigation, embodied question answering, and scene editing, while reducing reliance on multi-stage pipelines and proprietary APIs. However, stronger 3D scene understanding also introduces risks if deployed without safeguards. Scene graphs inferred from indoor scans may reveal private information about room layouts, object ownership, or activity patterns, and the same capabilities that support robot planning could be misused for surveillance, inventorying private spaces, or unauthorized physical interaction with objects. In embodied settings, incorrect detections or relations may also lead to unsafe plans, such as navigating through obstacles or manipulating the wrong object. Our work is evaluated as a benchmark perception method rather than an autonomous deployment system, but practical use should require consent for 3D data capture, privacy-preserving data handling, domain-specific safety checks, uncertainty-aware planning, and human oversight before physical actions are executed. The model should not be treated as a complete safety-critical robotics system without further validation in the target environment.

F Existing Asset Licenses

We use existing assets only for non-commercial academic research and evaluation, and do not redistribute restricted raw datasets, benchmark assets, simulator data, or pretrained checkpoints. All datasets, codebases, pretrained models, and baselines are cited in the main paper and used under their stated licenses or access terms.

We use 3DSSG annotations together with the underlying 3RScan/RIO10 assets, following the 3RScan Terms of Use required by the official 3DSSG release [59, 58]. ScanNet assets are used under the ScanNet Terms of Use, which restrict use to non-commercial research and educational purposes [9]. SceneVerse code is released under the MIT License, while its data are derived from multiple source datasets and therefore inherit the terms of the corresponding underlying datasets [20]. SceneSplat++/SceneSplat-49K is released through a gated dataset card listing CC-BY-SA-4.0, with

additional non-commercial research and educational-use conditions; for re-packaged components, we follow both the derived release terms and the original dataset licenses [37, 31]. Sonata code is Apache License 2.0 and Sonata weights are CC-BY-NC 4.0 [68]. SpatialLM1.1-Qwen-0.5B uses CC-BY-NC-4.0 model weights and Apache-2.0 Pointcept-based code [39, 69]. Qwen2.5-0.5B-Instruct is released under Apache License 2.0 [55]. Chorus is cited as the pretrained 3D Gaussian Splatting encoder used in our method [32]; since no public code or checkpoint license is currently available, we do not redistribute Chorus weights and use them only under the authors' release terms or explicit permission. For repository-based assets without stable release tags, the exact commit hashes used in our experiments are documented in the released configuration files.

```
{
  "summary": "The place with the most monitors is desk 10: monitors 12 and 13 are explicitly close
↪  by/adjacent to desk 10 via relationships [10,12,\"close by\"], [10,13,\"close by\"], [12,10,\"close
↪  by\"], [13,10,\"close by\"], and monitor 14 is also nearest in XY to desk 10 (desk 10 at
↪  [-0.794,0.282], monitor 14 at [-0.944,0.507], about 0.27 m away), while desk 18 has only one monitor
↪  20. Window candidates are 2, 23, and 24. Windows 23 and 24 are much closer in XY to desk 10 than
↪  window 2, and window 24 is the closest feasible one to desk 10 (window 24 at [-2.051,0.239], about
↪  1.26 m from desk 10 center; window 23 at [-2.326,0.239], about 1.53 m; window 2 is far on the
↪  opposite side). The top-side approach to window 24 is initially blocked by chair 6 in the narrow
↪  corridor above desk 10 and then by chair 5 near the final approach lane, so both chairs are
↪  relocated to nearby staging patches. After clearing, the robot approaches from free floor at base
↪  pose [-1.30,1.45,1.95], outside the windowsill/window boundary footprint, and opens/verifies window
↪  24 with the end effector.",
  "steps": [
    {
      "step": 1,
      "coordinate_m": [2.4, 1.65, 1.95],
      "action": "Start at the exact robot base start pose.",
      "evidence": "Robot start is exactly [2.40,1.65,1.95] per task and robot-body specification; first
↪  waypoint must match it exactly."
    },
    {
      "step": 2,
      "coordinate_m": [0.95, 1.45, 1.95],
      "action": "Navigate robot base to a safe manipulation pose beside chair 6.",
      "evidence": "Chair 6 is at [0.197,1.687,2.271], size [0.641,0.547,0.578]. The route is routed away
↪  from cabinet 7 at [1.197,1.887], size [0.563,0.469,0.344], and stays clear of wall 8 at
↪  [0.522,2.012], size [4.703,0.219,3.281], using the effective clearance radius 0.25 m. Chair 6 blocks
↪  the only top-side passage above desk 10 because chair 6 sits just above desk 10 (desk 10 at
↪  [-0.794,0.282], size [2.031,1.328,0.844]) leaving essentially no safe gap after expansion."
    },
    {
      "step": 3,
      "coordinate_m": [0.197, 1.687, 2.271],
      "action": "Use the end effector to grasp chair 6 for relocation.",
      "evidence": "End-effector target is chair 6 center [0.197,1.687,2.271]; from base
↪  [0.95,1.45,1.95], 3D reach is within manipulator_reach_m 1.1 m."
    },
    {
      "step": 4,
      "coordinate_m": [1.65, 1.45, 2.271],
      "action": "Place chair 6 at a free-floor staging patch with moved-chair center at
↪  [1.65,1.45,2.271].",
      "evidence": "Chosen staging patch is away from the remaining route and does not overlap cabinet 7
↪  at [1.197,1.887], wall 8 at [0.522,2.012], chair 1 at [1.772,0.637], or desk 18 at [1.849,-0.661].
↪  This clears the corridor above desk 10 that chair 6 previously blocked."
    },
    {
      "step": 5,
      "coordinate_m": [-1.30, 1.45, 1.95],
      "action": "Navigate robot base through the cleared top-side passage toward window 24.",
      "evidence": "After relocating the blocking chairs, the route to window 24 remains on free floor
↪  and avoids the expanded footprints of desk 10, nearby chairs, wall boundaries, and the window-area
↪  blockers."
    },
    {
      "step": 6,
      "coordinate_m": [-1.30, 1.45, 1.95],
      "action": "Approach window 24 from a safe base pose outside the window boundary.",
      "evidence": "The selected base pose is outside the windowsill/window boundary footprint and
↪  remains close enough for the manipulator to reach window 24."
    },
    {
      "step": 7,
      "coordinate_m": [-2.051, 0.239, 1.95],
      "action": "Open and verify window 24 with the end effector.",
      "evidence": "Window 24 is the closest feasible window to desk 10 after clearing chair 6 and chair
↪  5; the final action is performed from the collision-free approach pose."
    }
  ]
}
```


Figure 9: **Embodied Planning Using Predicted Scene Graphs as Grounding.** The JSON trace records the selected target desk, selected window, blocking objects, and the resulting navigation/manipulation sequence.

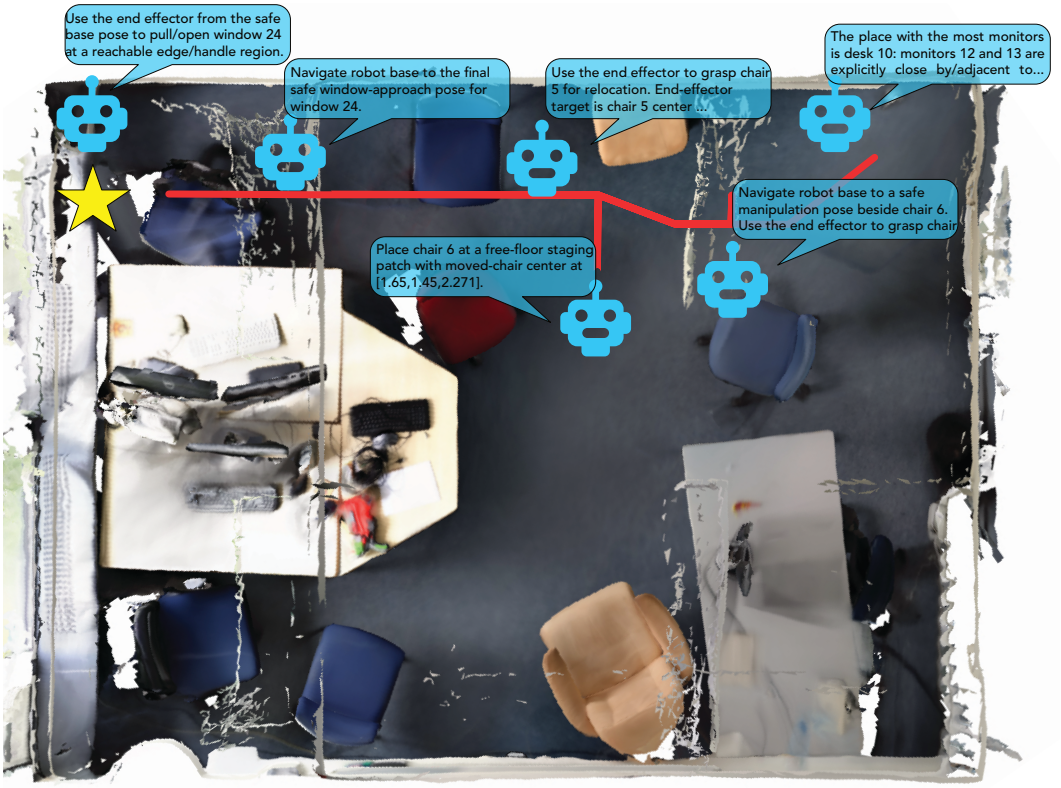

Figure 10: **Qualitative Embodied Planning Using 3D Scene Graph Reasoning.** Given a natural-language instruction, the predicted scene graph is used to ground the LLM reasoning and identify the desk with the most nearby monitors, select the closest feasible window, and derive a grounded right-to-left navigation plan. The robot first clears two blocking chairs along the upper passage, then approaches the selected window from a safe base pose and performs the final opening/verification action. This example illustrates how explicit object nodes, relations, and 3D geometry support compositional reasoning over the scene.

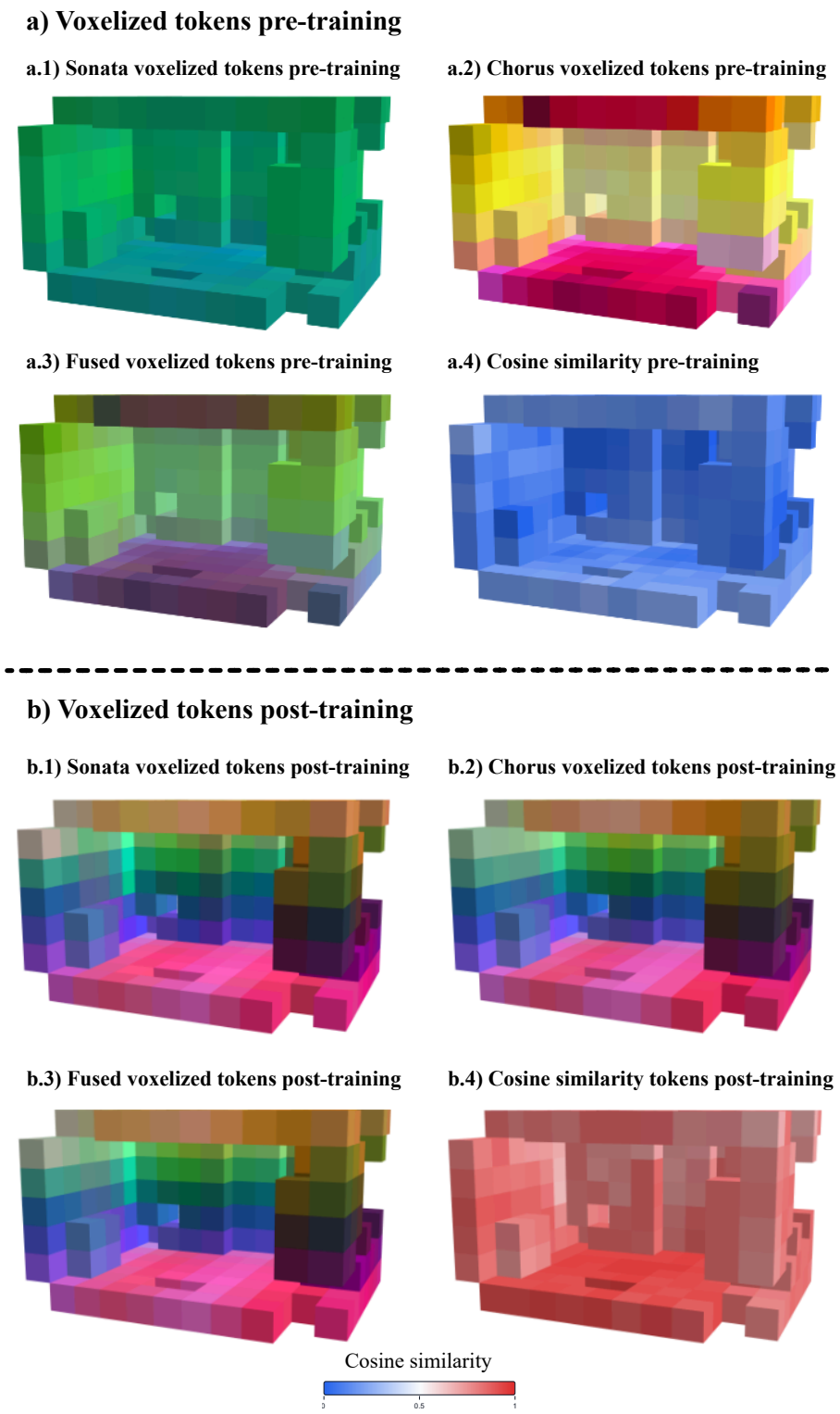

Figure 11: **Effect of Auxiliary Loss on Per-Voxel Feature Alignment** We visualize Sonata, Chorus, fused per-voxel features, and the corresponding cosine-similarity map between per-voxel features. Prior to training (a), the point-cloud and Gaussian-splat encoders produce inconsistent voxel-level feature patterns. After training, the modality-specific representations become aligned, which is shown by the substantial increase in cosine-similarity across corresponding voxels.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately state the paper's main contributions: end-to-end 3D scene graph generation from point clouds and/or Gaussian splats, structured JSON decoding, no ground-truth object annotations at inference, no proprietary model dependence, and evaluation on 3DSSG with ablations over modality and fusion choices.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The appendix D includes a dedicated limitations section discussing the restriction to indoor reconstructed scenes, the dependence on 3DGS reconstruction quality, the limited scale of 3DSSG subgraphs, and the dependence of free-form evaluation on Hungarian matching and semantic text-embedding ranking.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is empirical and does not present theoretical results, theorems, or formal guarantees. The mathematical expressions define the scene-graph representation, model components, and training losses rather than theoretical claims requiring proof.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the architecture, input modalities, training objective, datasets, evaluation metrics, object-matching protocol, semantic ranking protocol, baselines, and main hyperparameters needed to reproduce the reported experimental setting. The information on architecture and implementation details can be found in Section 3, Section 4 and section A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The paper states in abstract that code and model weights will be released upon publication, but the current submission does not provide open access to the code/model.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the datasets, evaluation vocabularies, metrics, matching protocol, baselines, model components, initialization, optimizer, learning rate, batch size, number of epochs, modality dropout, auxiliary-loss schedule, and compute setup. This data is included in Section 4 as well as the appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer:[No]

Justification: The paper reports benchmark recall and F1 values, ablations, and qualitative results, but it does not report error bars, confidence intervals, statistical significance tests, or variability across random seeds or repeated training runs. The paper follows the standard practices of the scene graph literature for performing evaluations.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The appendix reports the compute environment, including the use of a single NVIDIA H200 GPU with 143GB memory, Intel Xeon Gold 6548Y+ CPUs, 8 requested CPU cores, 80GB host memory, 6-hour wall-clock allocation, approximately 4 hours per fine-tuning run, and approximately 1000 H200 GPU-hours for reported runs and ablations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work uses existing 3D scene datasets, pretrained models, and benchmark evaluations, and does not involve human-subject experiments, crowdsourcing, or newly collected participant data. The submission is anonymized and we are not aware of any deviation from the NeurIPS Code of Ethics.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer[Yes]

    Justification: The conclusion 5 discusses positive broader impacts for spatially grounded 3D vision-language models, robotics, and augmented reality. Furthermore, section E discuss possible negative societal impacts and security considerations.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not introduce a scraped dataset, image generator, or high-risk general-purpose generative model. The released model is intended for structured 3D scene-graph prediction from 3D scene inputs, as explained in Section 3 rather than unrestricted text or image generation.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite the creators and owners of all datasets, code, pretrained models, and baselines used in our experiments, including 3DSSG, ScanNet, SceneVerse, SceneSplat++, SpatialLM, Sonata, Chorus, and Qwen2.5, and all usage follows the corresponding terms. Datasets, code, pretrained models, and baselines used in our experiments thoroughly discussed throughout the paper in Section 2, Section 3, Section 4 and section F.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The paper describes the usage of a language model as a core component of the proposed architecture to produce outputs, as detailed in Section 3.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.