

PARSE: Part-Aware Relational Spatial Modeling

Supplementary Material

A. Details of PARSE Framework

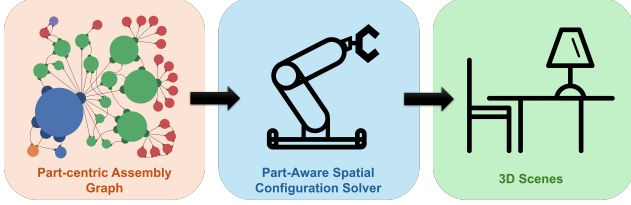


Figure 1. Overview of PARSE Framework

The overview of the PARSE framework is shown in Fig. 1, and the specific details of this framework will be elaborated in this section.

A.1. PAG Representation

Graph Structure and Data Definition. We implement the Part-centric Assembly Graph (PAG) as a hierarchical directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Nodes ($\mathcal{V} = \mathcal{V}_O \cup \mathcal{V}_P$): Each node $u \in \mathcal{V}_O$ represents a distinct object instance in the scene. In our data structure, u stores global attributes: the semantic category label C , a scaling factor S , and a retrieved asset \mathcal{M} . Each node $v \in \mathcal{V}_P$ serves as a part node attached to a parent object node. Implementation-wise, the part node stores a description tuple $\tau = (l, s)$. Here, l is a semantic label (e.g., “seat”, “leg”, or a default “root”), and s denotes the six directional faces of the part mesh (top, bottom, left, right, front, back), which serve as the primary entities processed by the solver when solving part-level constraints (details discussed later).

Edges ($\mathcal{E} = \mathcal{E}_{obj} \cup \mathcal{E}_{part}$): Object-Level Edges (\mathcal{E}_{obj}) are edges that connect two object nodes to enforce macroscopic layout constraints (e.g., *left of*, *near*), primarily used for coarse region proposal. Part-Level Edges (\mathcal{E}_{part}) are edges that are stored as a triplet $e = (v_{src}, v_{tgt}, r)$, connecting a source part node v_{src} to a target part node v_{tgt} via a specific spatial relation type r .

Spatial Relations. The geometric nature of the connection is determined by the relation type r . To handle the complexity of 3D spatial interactions, we define a comprehensive dictionary of relations. Tab. 1 enumerates these relations, categorizing them by their physical support roles and spatial granularity, and detailing the underlying geometric constraints enforced by the solver.

Algorithm 1: Coarse-to-Fine Spatial Solver

Input : Graph \mathcal{G} , Asset Database \mathbb{D}
Output: Selected Assets \mathcal{M} , Scene Poses $\mathcal{T} = \{(R_i, t_i)\}$

```

1  $\mathcal{L} \leftarrow \text{TopologicalSort}(\mathcal{G})$ 
2 foreach Object Node  $v_i \in \mathcal{L}$  do
3    $\mathcal{M}_i \leftarrow \text{SelectAsset}(v_i, \mathbb{D})$ 
4    $\mathcal{S}_{supp} \leftarrow v_i.\text{GetSupportSurface}()$ 
5    $\text{Obs} \leftarrow \text{OccupiedFootprints}(\mathcal{T})$ 
6    $\mathcal{S}_{free} \leftarrow \text{SurfaceClip}(\mathcal{S}_{supp}, \text{Obs})$ 
7   foreach  $e_{obj} \in \text{GetObjectEdges}(v_i)$  do
8      $\text{Region} \leftarrow$ 
9        $\text{GetObjectConstraintRegion}(e_{obj})$ 
10     $\mathcal{S}_{free} \leftarrow \text{SurfaceClip}(\mathcal{S}_{free}, \text{Region})$ 
11 end
12  $\Omega_{parts} \leftarrow \emptyset$ 
13 foreach  $e_{part} \in \text{GetPartEdges}(v_i)$  do
14    $\mathcal{S}_{child} \leftarrow \text{ResolveSurface}(e_{part}.child)$ 
15    $\mathcal{S}_{parent} \leftarrow \text{ResolveSurface}(e_{part}.parent)$ 
16    $\Omega_{parts}.add((\mathcal{S}_{child}, \mathcal{S}_{parent}, e_{part}.relation))$ 
17 end
18  $\mathcal{C}_{rot} \leftarrow$ 
19    $\text{ExtractNormalConstraints}(\Omega_{parts})$ 
20  $R \leftarrow \text{SolveRotation}(\mathcal{C}_{rot})$ 
21 foreach  $(\mathcal{S}_{child}, \mathcal{S}_{parent}, r) \in \Omega_{parts}$  do
22    $\mathcal{S}'_{child} \leftarrow R \cdot \mathcal{S}_{child}$ 
23    $\text{Region} \leftarrow$ 
24      $\text{GetPartConstraintRegion}(\mathcal{S}_{parent}, \mathcal{S}'_{child}, r)$ 
25    $\mathcal{S}_{free} \leftarrow \text{SurfaceClip}(\mathcal{S}_{free}, \text{Region})$ 
26 end
27 for  $k \leftarrow 1$  to  $\text{MaxTrials}$  do
28    $t \leftarrow \text{Sample}(\mathcal{S}_{free})$ 
29   if  $\text{ValidatePose}(R, t, \mathcal{T}, \Omega_{parts})$  then
30      $\mathcal{T}[v_i] \leftarrow (R, t)$ 
31     break
32 end
33 end
34 return  $\mathcal{T}$ 

```

A.2. Part-Aware Spatial Configuration Solver Details

The detailed algorithmic workflow of our solver, corresponding to the process described below, is formally presented in Algorithm 1.

Asset Instantiation and Geometric Resolution. Before any spatial reasoning occurs, the abstract PAG nodes must be grounded into 3D assets.

Table 1. **Dictionary of Spatial Relations in PARSE.** We categorize relations into two primary granularities: **Part-Level** (fine-grained geometric control) and **Object-Level** (coarse layout). Specific Part-Level relations function as Support Relations, defining the scene’s spanning tree. To handle unspecified inputs, we define **Default Surfaces** for implicit resolving. Notation: A is Child, B is Parent; S denotes surface, \mathbf{n} denotes normal, \mathbf{x} denotes centroid.

Relation (r)	Semantic Description	Support?	Default Surfaces ($S_A \rightarrow S_B$)	Geometric Constraints & Solver Logic
1. Part-Level: Support Relations				
<i>These relations define the primary parent-child dependency, organizing the scene assembly order.</i>				
ON	Object rests stably on parent.	Yes	Bottom \rightarrow Top	Contact: $\text{dist}(S_A, S_B) \approx 0$. Orientation: $\mathbf{n}_A \cdot \mathbf{n}_B \approx -1$ (Anti-parallel). Stability: $\text{Proj}_{xy}(\text{CoMA}) \in \text{Polygon}(S_B)$.
IN	Object is contained inside parent.	Yes	Bottom \rightarrow Top(internal support surface)	Contact: $\text{dist}(S_A, S_B) \approx 0$. Inclusion: $\text{RayCastEnclosure}(A, B) > \tau_{\text{contain}}$.
HANGING_BELOW	Object suspended under parent.	Yes	Top \rightarrow Bottom	Contact: $\text{dist}(S_A, S_B) \approx 0$. Orientation: $\mathbf{n}_A \cdot \mathbf{n}_B \approx -1$.
SECURED_TO	Rigidly attached to side/wall.	Yes	Back \rightarrow Front	Contact: $\text{dist}(S_A, S_B) \approx 0$. Orientation: $\mathbf{n}_A \cdot \mathbf{n}_B \approx -1$.
2. Part-Level: Fine-Grained Geometric Constraints				
<i>Fine-grained surface constraints refining the pose without defining assembly hierarchy.</i>				
AGAINST	Stable surface contact.	No	Front \rightarrow Front	Contact: $\text{dist}(S_A, S_B) \approx 0$. Orientation: $\mathbf{n}_A \cdot \mathbf{n}_B \approx -1$.
LEANING_AGAINST	Leans on parent (contact + angle).	No	Back \rightarrow Front	Contact: $\text{dist}(S_A, S_B) \approx 0$. Angle: $\mathbf{v}_{up} \cdot \mathbf{v}_{gravity} \approx \cos(\theta), \theta \in (0, \frac{\pi}{2})$.
ALIGNED_WITH	Surfaces flush, facing same dir.	No	Front \rightarrow Front	Coplanarity: $\text{dist}(\text{Plane}(S_A), \text{Plane}(S_B)) \approx 0$. Orientation: $\mathbf{n}_A \cdot \mathbf{n}_B \approx 1$ (Parallel).
OPPOSITE_TO	Surfaces flush, back-to-back.	No	Front \rightarrow Front	Coplanarity: $\text{dist}(\text{Plane}(S_A), \text{Plane}(S_B)) \approx 0$. Orientation: $\mathbf{n}_A \cdot \mathbf{n}_B \approx -1$ (Anti-parallel).
PARALLEL_TO	Planes parallel (any distance).	No	Front \rightarrow Front	Orientation: $\mathbf{n}_A \cdot \mathbf{n}_B \approx 1$ (Parallel).
3. Object-Level: Coarse Layout				
<i>Region-based constraints acting on object centroids or bounding boxes.</i>				
SIDE_OF	Generic half-space constraint.	No	N/A	Region: $(\mathbf{x}_A - \mathbf{x}_B) \cdot \mathbf{n}_{side} > 0$.
ALONG	Arranged along a linear strip.	No	N/A	Region: $0 < (\mathbf{x}_A - \mathbf{x}_B) \cdot \mathbf{n}_{side}^\perp < \delta_{width}$.
NEAR / FAR_FROM	Proximity constraint.	No	N/A	Distance: $\ \mathbf{x}_A - \mathbf{x}_B\ _2 \lesseqgtr \tau_{thresh}$.
TOWARD	Directional orientation.	No	N/A	LookAt: $\mathbf{n}_{A,front} \cdot \frac{\mathbf{x}_B - \mathbf{x}_A}{\ \mathbf{x}_B - \mathbf{x}_A\ } \approx 1$.

- **Asset Retrieval and Scaling:** For the current object node u_i , we retrieve a specific 3D asset \mathcal{M}_i from the database \mathbb{D} that matches the node’s semantic category. To ensure physical realism, we define a scale range for each object category and, for every instance, sample its scale from the corresponding category-specific range.
- **Geometric Resolution:** To construct mathematical constraints, we aim to convert geometric relationships between irregular part meshes into constraints defined over surfaces. Specifically, for each part, we first compute its bounding box (bbox) in canonical pose. For each bbox face, we then search on the mesh surface for all connected face patches whose normals are aligned with the normal of that bbox face. Among these candidate patches, we select the one that is closest to the bbox face and has the largest area. We designate it as the directional surface for that face. If no suitable mesh patches can be found, we directly use the bbox face itself as the directional surface. In this way, each part is associated with six directional

surfaces corresponding to the six faces of its bounding box.

Coarse Localization (Object-Level). This stage focuses on object-level relational constraints, which establish coarse spatial arrangements between objects and thereby reduce the search space for final feasible poses. We initiate the process by identifying the parent’s support surface S_{supp} and subtracting the regions already occupied by other objects to obtain the initial collision-free space S_{free} . Subsequently, we systematically apply the Object-Level Coarse Relations (see Tab. 1) to further contract this region. Directional relations (e.g., **SIDE_OF**) are implemented by specifying the relative positions of object centroids; proximity relations (e.g., **NEAR**, **FAR**) specify inclusion or exclusion regions based on prescribed distances; and alignment relations (e.g., **ALONG**) constrain objects to lie on the same line; and orientation relations (e.g., **TOWARD**) impose a coarse “Look-At” constraint that binds the object’s forward vector

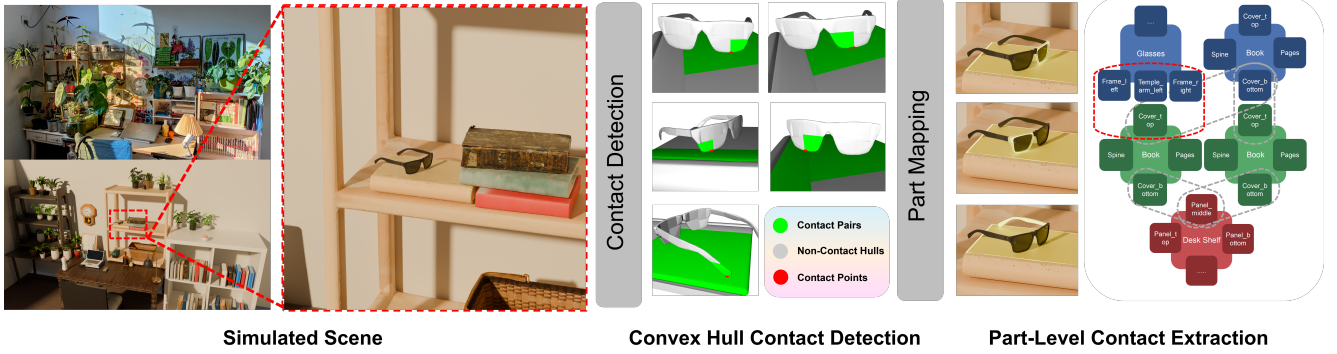


Figure 2. **Overview of the part-level contact detection pipeline.** After simulating the scene, we perform contact detection on collision meshes by uniformly sampling surface points on every pair of convex hulls across object pairs. Each detected convex-hull contact pair is then mapped back to its corresponding visual-mesh parts, producing a set of part-level contact pairs (each row maps its hull-level pairs to the corresponding final part-level pair on the right). These part contacts are aggregated to construct the part-level contact graph (the red highlight indicating the contacts between the glasses and the book).

to the target’s centroid. Based on these 3D constraints, we further filter the original \mathcal{S}_{free} to obtain the final reduced space for this stage, denoted as \mathcal{S}_{coarse} .

Fine-Grained Alignment (Part-Level). Building upon the resolved surfaces, this phase further determines the object’s full 6-DoF pose based on part-level constraints. For every part node on the object that participates in at least one relation edge, we sequentially solve its associated constraint. For each surface of each part, we compute its feasible pose space by solving the relation constraints with the corresponding surface of the other object’s part. The intersection of the result pose spaces of all six surfaces is then taken as the valid pose space for that part; if this intersection is empty, the object is immediately marked as a failure case, and we proceed to the next object. After processing all relation-involved parts, we intersect their individual pose spaces with \mathcal{S}_{coarse} to obtain the object’s final feasible pose space for this stage, marked as \mathcal{S}_{fine} . If this final intersection is empty, the object is likewise labeled as a failure case and discarded before moving on.

Sampling and Validation. To finalize the object’s placement, we uniformly sample a candidate pose \mathbf{t} from the minimal subspace \mathcal{S}_{fine} . This candidate undergoes a validation procedure to ensure physical and semantic consistency. First, we verify adherence to all active geometric constraints defined in Tab. 1, ensuring that the generated pose satisfies the full set of relational requirements. Second, we perform a global collision check against the static scene using FCL-accelerated mesh-level interference detection. If a candidate fails validation, it is discarded, and we resample a new pose; if the process exceeds a predefined maximum number of trials without success, the placement is flagged as a failure.

B. Details of Asset Library

Our dataset is constructed by combining a subset of Part-NeXt [13] data, which provides part annotations over integrated assets from 3D-FRONT [5], ABO [2], and Objaverse [3]. In addition, for object categories that commonly appear in daily environments but are not covered by Part-NeXt, we first collect reference images from the Internet and use Rodin [16] to synthesize corresponding 3D assets. We then apply P3-SAM [8] to perform part segmentation and use GPT [9] to generate part-level semantic annotations. Finally, we manually verify and adjust the canonical pose of instances in each object category to ensure category-level pose consistency, which is crucial to applying relation constraints in the subsequent solver.

To better characterize the properties of our assembled asset collection, we perform a detailed statistical analysis of both object categories and the number of part segments per object. As shown in Fig. 3, the category composition reveals a broad coverage of everyday object types. In addition, Fig. 4 presents the distribution of segmented part counts per object. On average, each object contains 17.4 parts.

C. Scenes

C.1. Scene Simulation Procedure and Contact Extraction

After generating each scene, we simulate it in Sapien [15] and subsequently extract the part-level contact relations. Concretely, we first apply COACD [14] to perform convex decomposition for every object. During simulation, we use the COACD-generated meshes as collision meshes for stable and accurate physical interactions, while the original meshes are kept only for visualization. The simulation runs with a timestep of 0.01 s for 1000 steps, and all objects are

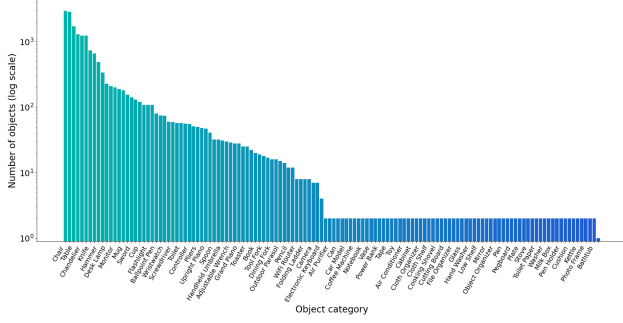


Figure 3. Distribution of Object Categories

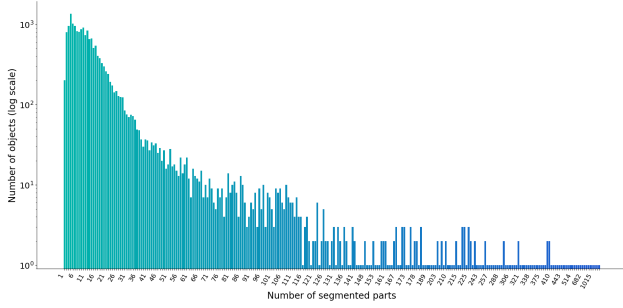


Figure 4. Distribution of Number of Parts Count per Object

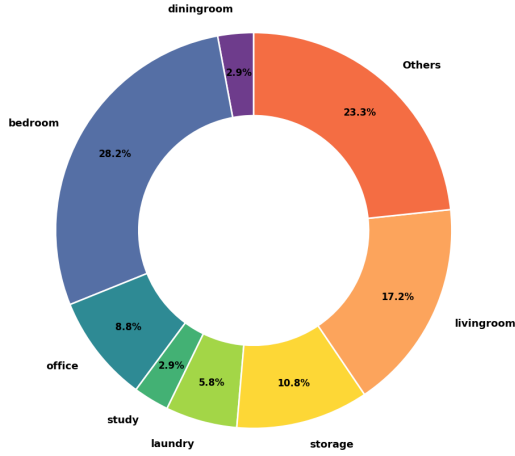


Figure 5. Distribution of Room Types

assigned a linear damping of 0.05 m/s and an angular damping of 0.5 deg/s. After simulation, we extract the contact graph of the entire scene, and the whole process is shown in Fig. 2. First, we perform contact detection by sampling points on the surfaces of each convex hull of each collision mesh. Next, for each pair of objects, we examine every pair of their convex hulls. For both convex hulls in each pair, we uniformly sample surface points and record all points whose distance to the opposite hull is below 0.001 m. Each recorded point is then mapped to the nearest visual-mesh part on its own object, producing a pair of contacting parts

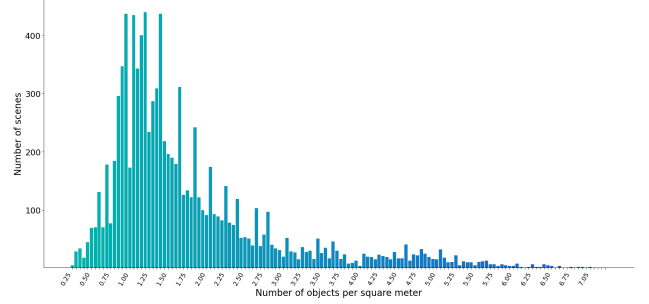


Figure 6. Distribution of Object Density Across Rooms

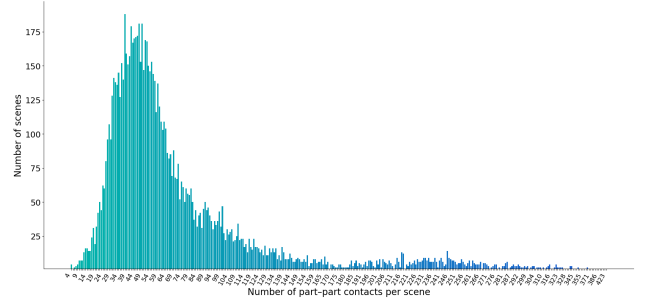


Figure 7. Distribution of Part-Part Contacts per Scene

corresponding to the two convex hulls. By aggregating all such part pairs across the scene, we construct an undirected graph that represents the part-level contact graph of the entire environment.

C.2. Scene Dataset Statistics and Analysis

Our scene data are constructed as follows. With our UI shown in Fig. 8, we first design 200 distinct PAGs inspired by real-world images and categorize them into 17 room types, including bedroom, living room, office, study, laundry, storage, dining room, bathroom, tea house, game room, kitchen, music room, bar, dormitory, meeting room, hotel, and grocery. These PAGs collectively drive the generation of 10,000 scenes, providing broad coverage of diverse and commonly encountered indoor environments. The distribution of these scenes across different room types is illustrated in Fig. 5. To validate the quality of our generated scenes, we aim to show that they are both structurally complex and rich in physical interactions. To this end, we compute the object density (number of objects per square meter) for each room, as shown in Fig. 6. We find that more than 55% of the scenes have an object density exceeding 1.35 objects per square meter. To demonstrate the richness of physical interactions, we analyze the distribution of the number of contacts per scene of our generated scenes, shown in Fig. 7. The results show that more than 57% of the scenes contain more than 50 part-level contacts, making them ideal for studying

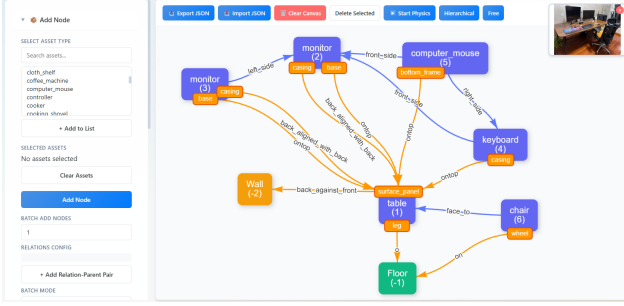


Figure 8. User Interface for Designing PAGs

complex, multi-object interaction and physical stability.

D. Details of Experiments on Spatial Tasks

D.1. Implementation Details of VLM Spatial Reasoning

We fine-tune a large vision language model on 3D scene understanding tasks using the **Qwen3-VL-235B-A22B-Instruct** [11], which is a 235B-parameter, instruction-tuned multimodal model capable of jointly processing images and text. To make training tractable at this scale, we adopt a parameter-efficient LoRA [7] strategy rather than full-parameter fine-tuning: low-rank adapters are inserted into all linear layers of the transformer blocks, with rank 8 and scaling factor 32, and only these adapter parameters are updated. The underlying pre-trained weights of Qwen3-VL remain frozen throughout fine-tuning.

Images are fed through the official Qwen3-VL visual encoder at a fixed maximum resolution, with the number of visual tokens capped at 1024 (via `IMAGE_MAX_TOKEN_NUM`). Pre-processing follows the original resizing and padding pipeline, after which the image tokens are concatenated with text tokens into a single multimodal sequence. For each 2D view rendered from the 3D scenes, we project the 3D object boxes to the image plane and obtain 2D bounding boxes in pixel coordinates. These coordinates are normalized and quantized before being injected into the text. Concretely, a coordinate (x, y) is divided by the image width and height, respectively, mapped into the range $[0, 1]$, and then discretized into integer bins in $[0, 1000]$. A bbox is serialized as a short textual span `<|box_start|>x_min, y_min, x_max, y_max<|box_end|>`, which is then tokenized by the standard tokenizer. This encoding makes spatial information explicit while remaining compatible with the instruction-tuned interface of the base model.

The model is trained in a multi-task fashion on three types of supervision: visual relation multiple-choice questions, part-level contact multiple-choice questions, and scene graph generation. In the visual relation MCQ task,

each training example selects a pair of objects A and B in the image, provides their bounding boxes highlighted on the image, and asks for their relationship (for example “left of”, “on” or “no contact”). The ground-truth option is determined from the underlying 3D annotations, and the assistant response is supervised to be exactly the letter of the correct option.

A typical prompt for the visual relation MCQ task is:

User: The image displays two objects highlighted with a red and a blue bounding box. Your task is to identify the object within the red box, the object within the blue box, and the spatial relationship between them. From the options below, choose the one that correctly describes this scene. (A) ..., (B) ..., (C) ..., (D) ... Your answer should only be the letter of the correct option.

Assistant: C

The part-level contact MCQ task refines this setup by asking which specific parts of two objects are in physical contact, rather than only their coarse relation. Each example highlights two objects (for instance, a bookcase and a bucket) with red and blue boxes in the image, and the prompt enumerates several hypotheses involving part names drawn from a predefined part ontology (e.g., “Shelf Body” or “Back Panel” for the bookcase, “Main Body” or “Handle” for the bucket), together with a “no contact” option. The correct option is derived from part-level contact labels computed in the 3D space, and the model is again trained to output only the option letter.

A typical prompt for the part-level contact MCQ task is:

User: The image displays two objects highlighted with a red and a blue bounding box. Your task is to identify the object within the red box, the object within the blue box, and whether they are in contact with each other. From the options below, choose the one that correctly describes this scene. (A) ..., (B) ..., (C) ..., (D) ... Your answer should only be the letter of the correct option.

Assistant: C

Finally, the scene graph generation task supervises the model to produce a global relational description of the scene. Here the user message lists the object and relation candidates and asks the model to detect and localize all objects, then identify the spatial relationships between the detected objects. The assistant is trained to return a complex scene graph, such as a sequence of “(bookcase, supporting, cup); (table, under, lamp); ...” triplets. Unlike the MCQ tasks, this supervision is open-ended and encourages the model to maintain precise analysis over the whole scene.

All tasks are packed into a unified JSON-based dialogue format, where each entry contains a `messages` field (a short user–assistant conversation) and a list of associated

images. The task type is encoded purely in natural language, without special control tokens. During fine-tuning we simply mix the three datasets according to fixed sampling ratios and train a single set of LoRA adapters on the combined corpus. Optimization is performed with AdamW at a learning rate of 1×10^{-4} with linear warm-up over 5% of the updates and a minimum learning rate of 1×10^{-5} , using a global batch size of 16 (micro-batch size 2 per GPU on 8 NVIDIA H20 GPUs). We run one epoch over the mixed dataset, perform evaluation on held-out validation splits corresponding to each task, and report final results on disjoint test sets sampled at the scene level to avoid leakage across different camera views.

D.2. Implementation Details of Scene Generation

We represent each scene as an object-centric graph. Each object node carries its pose parameters (translation with 3 DOF, scale with 3 DOF, and rotation with 6 DOF), forming a 12-dimensional geometric descriptor, and a Michelangelo-based shape embedding [17] (described below). The geometric descriptor and positional code are concatenated and projected into the 512-dimensional model space by a linear layer. In parallel, for each object we extract a 256×64 -dimensional latent code from the pre-trained Michelangelo encoder, which maps 3D shapes into a joint shape-image-text latent space. This latent is again projected to 512 dimensions through a linear layer and interacts with the pose representation inside the multimodal graph blocks, providing high-level priors about plausible object shapes and categories.

Pairwise relations between objects are modeled as directed edges in the scene graph. For every ordered pair of nodes, we associate a CLIP [12] relation embedding in \mathbb{R}^{768} , which captures semantic relations such as “left of”, “below”, or “supporting”. These 768-dimensional vectors are processed by a two-layer MLP that maps them into the 512-dimensional edge space used by the Graph Transformer [4]. Inside each block, graph attention operates jointly on nodes and edges: attention logits between nodes are modulated by the corresponding edge embeddings, and node and edge streams are updated by separate feed-forward networks. All submodules are wrapped with residual connections and either standard LayerNorm [1] or AdaLayerNorm [10], depending on whether timestep conditioning is required.

We treat 3D layout prediction as a conditional diffusion process [6] over object poses. The forward noising process follows the DDPM formulation, using 1000 diffusion steps and a squared cosine beta schedule. At each training step, we sample a timestep t uniformly from 1 to 1000, add Gaussian noise to the ground-truth boxes according to the DDPM variance schedule, and train the network to predict the added noise (epsilon prediction). The timestep t is first

converted into a sinusoidal embedding and then mapped to the 128-dimensional vector used by AdaLayerNorm; this embedding is broadcast to all layers and injected before graph attention and before the feed-forward networks, so that the model learns a time-dependent denoising trajectory in the space of 3D layouts.

For optimization, we use AdamW, with the learning rate set to 8×10^{-4} , the weight decay to 1×10^{-4} , and the Adam betas to (0.9, 0.999). Training is performed on eight NVIDIA H20 GPUs using the `accelerate` library, with a per-GPU batch size of a single scene graph. For each scene in the batch, we further sample 16 independent noise vectors and timesteps, resulting in 16 noisy versions of the same ground-truth layout; this yields an effective batch size of 128 diffusion samples per optimization step across all GPUs. We train for 2000 epochs, with 500 gradient updates per epoch. Throughout training, we maintain an exponential moving average of all model parameters with a maximum decay rate of 0.9999 and a warm-up strategy controlled by `inv_gamma = 1.0` and `power = 0.75`, following the default configuration used in `diffusers`. All quantitative metrics and qualitative visualizations reported in the paper are obtained using these EMA-smoothed weights at test time.

References

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. 6
- [2] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21126–21136, 2022. 3
- [3] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023. 3
- [4] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021. 6
- [5] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Bingqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021. 3
- [6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2020. Curran Associates Inc. 6
- [7] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen.

- LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 5
- [8] Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P3-sam: Native 3d part segmentation. *arXiv preprint arXiv:2509.06784*, 2025. 3
- [9] OpenAI. Introducing gpt-5. <https://openai.com/index/introducing-gpt-5/>, 2025. Accessed: 2025-11-12. 3
- [10] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2018. 6
- [11] QwenLM. Qwen3-vl: Sharper vision, deeper thought, broader action. <https://qwen.ai/blog?id=99f0335c4ad9ff6153e517418d48535ab6d8afef>, 2025. Accessed: 2025-11-12. 5
- [12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 6
- [13] Penghao Wang, Yiyang He, Xin Lv, Yukai Zhou, Lan Xu, Jingyi Yu, and Jiayuan Gu. Partnext: A next-generation dataset for fine-grained and hierarchical 3d part understanding. *arXiv preprint arXiv:2510.20155*, 2025. 3
- [14] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022. 3
- [15] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020. 3
- [16] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Transactions on Graphics (TOG)*, 43(4):1–20, 2024. 3
- [17] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, BIN FU, Tao Chen, Gang YU, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 6