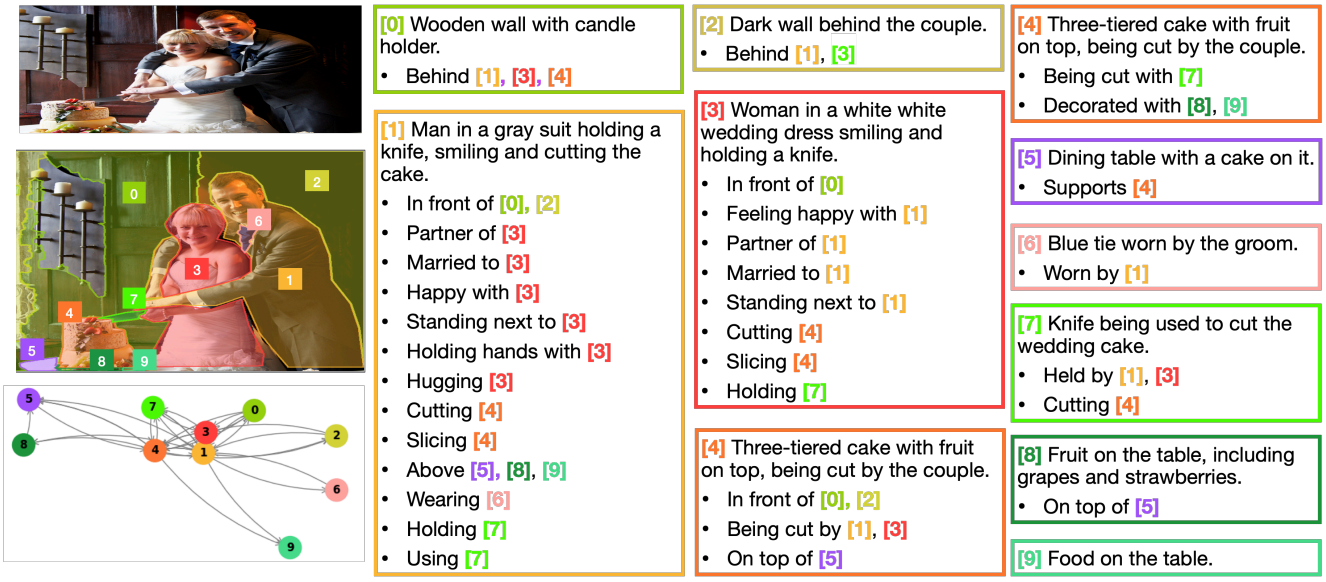
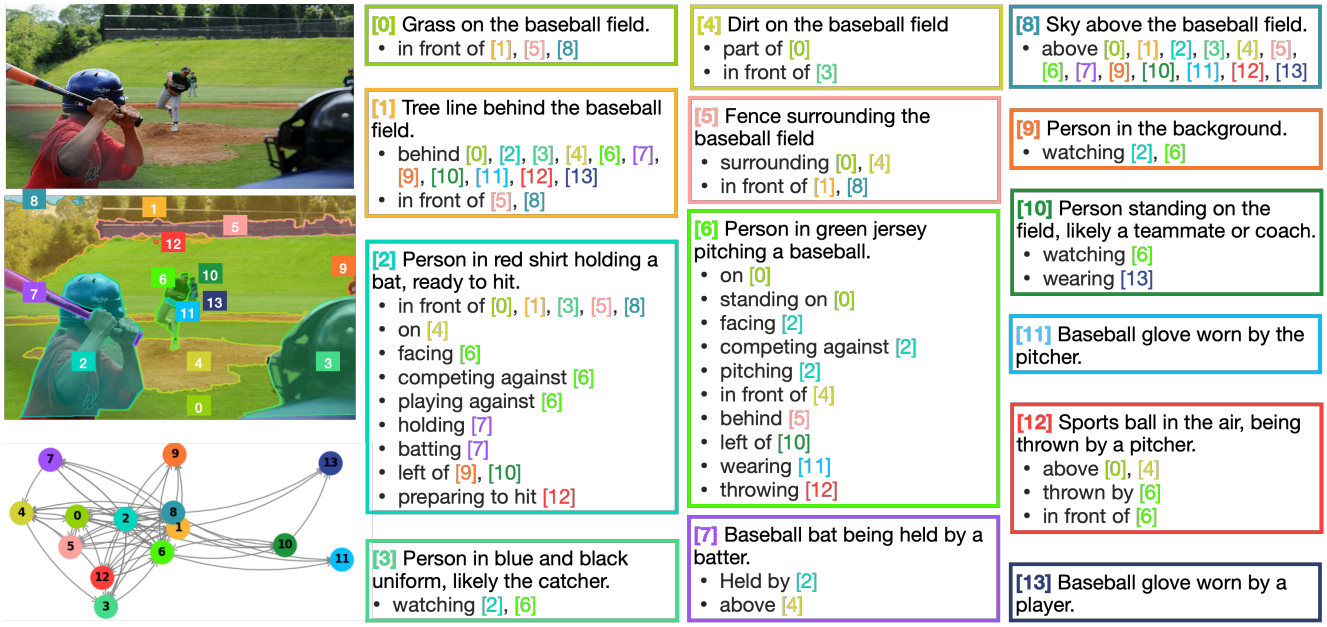
Synthetic Visual Genome
Supplementary Material



(a)



(b)

Figure 7. Dense scene graph generated by ROBIN on the Panoptic Scene Graph dataset [101]

A. Qualitative Examples

We present additional qualitative results of dense scene graphs generated by Robin in Figure 7. We observe that ROBIN captures a diverse set of relationships among the objects in the scene. In Figure 7a, the relationships associated with the man in a gray suit ([1]) include spatial (e.g., *in front of* [0], [2]; *standing next to* [3]), social (e.g., *partner of* [3], *married to* [3]), emotional (e.g., *happy with* [3]), and interactional (e.g., *hugging* [3], *cutting* [4], *holding* [7]). This demonstrates that our scene graph instruction tuning framework, leveraging the strong grounding and reasoning capabilities of ROBIN, effectively captures

the comprehensive relationships involving the man in the scene. Note that such density and richness of semantic information have not been observed in previous classification-based models. The generated descriptions go beyond conventional object labels, providing detailed information such as the cake [4] being a "three-tiered cake with fruit on top, being cut by the couple," or the "wooden wall with candle holder" [0], enhancing the ability to distinguish different objects in the scene. Additionally, the model demonstrates a precise understanding of spatial depth; for instance, the "behind" relation for the dark wall [2] correctly excludes the cake [4], whereas the wooden wall [0] is accurately annotated as being behind the cake.

Figure 7b presents another set of dense relationships. It accurately locates the batter [2] and the pitcher [6], including the relationship that they are competing against each other. People in the background ([9] and [10]) are accurately identified, and the model includes salient information that they are watching the two players ([2] and [6]). Notably, the model is able to identify small objects, such as the sports ball in the air [12], and correctly infers that it is thrown by the pitcher [6], and that the batter [2] is preparing to hit it [12], demonstrating strong localization ability enabled in pixel and text space.

A.1. Stage 1 vs. Stage 2 Training

Figure 8 presents a qualitative comparison between the Stage 1 and Stage 2 trained versions of ROBIN-3B. Notably, the Stage 2 model demonstrates a more accurate and precise understanding of the segmentation mask, correctly labeling [0] and [1] as the left and right walls, respectively. In contrast, the Stage 1 model inaccurately labels [0] and [1] as "round mirror," and exhibits hallucinations by stating that there are cups on the bathroom counter in regions [10] and [11]. These observations indicate that Stage 2 training is crucial for resolving such issues and improving the visual perception capabilities of the model.

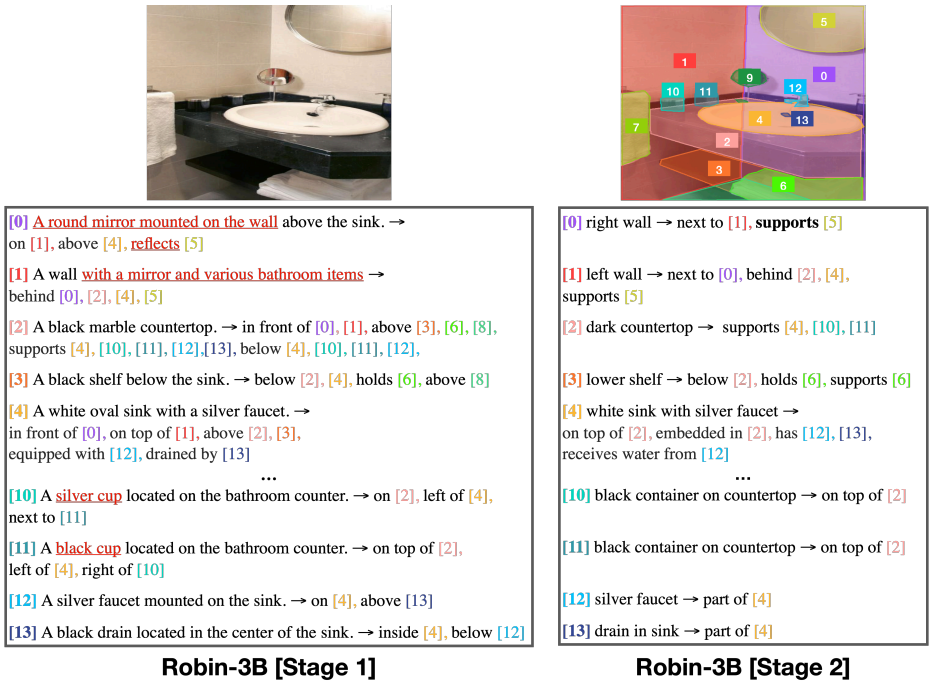


Figure 8. Comparison of models trained in Stage 1 and Stage 2. Errors are highlighted and underlined in red. The Stage 1 model lacks precise understanding of the segmentation mask, incorrectly mentioning a mirror for regions [0] and [1], and is more prone to hallucinations (e.g., mentioning a cup on regions [10] and [11]).

A.2. GPT Generated Scene Graph

Lastly, we evaluate the capabilities of GPT-4 to generate a scene graph from scratch. Inspired by set-of-mark prompting [85], we include the segmentation masks and numerical IDs in the image, as well as the bounding box coordinates of the regions, and prompt the model to generate such scene graph. Figure 9 shows an example. We observe that the GPT-generated scene graph contains blatant mistakes, lacking precise depth and grounding information. For instance, it inaccurately labels [4] as the stroller with child, while it actually refers to the car. The sidewalk is incorrectly labeled as being walked on by [1] and [2], whereas the street [6] is the correct object. The model also hallucinates that [7] is a phone in hand instead of region [9]. Based on this evaluation, we assert that generating scene graph directly from GPT4 is infeasible and our data curation pipeline, which constructs SVG from a seed dataset with accurate human annotations, is necessary to train ROBIN with accurate and

Figure 9. Example of scene graph quality generated with GPT-4o from scratch with provided regions. Errors are highlighted and underlined in red.

high-quality scene graphs. Moreover, this shows that our model possesses the unique capability of generating dense scene graphs that state-of-the-art models like GPT-4 lack, highlighting the significance of our contribution.

B. Instruction Tuning Details

B.1. Implementation details

We train the model for two epochs for each stage. The input image is padded and resized to 512x512 single crop image. The LM has a learning rate of $1e^{-5}$ and supports max sequence length of 8192. In Stage 0, we keep the vision encoder frozen with 128 batch size and took 6 hours to train with 16 H100 GPUs. In Stage 1 and Stage 2, we train the visual encoder jointly with different learning rates of $1e^{-}6$, which is a common practice in SoTA MLMs [13]. Both stages took 12 hours with 32 H100 GPUs. We use batch size of 64, linear warmup ratio of 0.03 with cosine annealing, and Adam optimizer with DeepSpeed Stage-2 configuration [67].

B.2. Encoding Image Regions in Pixel and Text Space

Inspired by previous MLMs that support region-based understanding [52, 92, 97], we represent specific regions within an image using unique numerical identifiers `region{i}`. These identifiers can be referenced through bounding box coordinates embedded in the text and/or via segmentation masks in the pixel space. To facilitate this, we define a special token `<region{i}>`, that is appended with a mask token `<mask>` and a spatial position token `<pos>`. These tokens serve as placeholders that are replaced by the corresponding region and spatial features following Osprey [92]. Additionaly, we include bounding box coordinates `<|box_start|>(x1, y1),(x2, y2)<|box_end|>`, where $(x1, y1)$ and $(x2, y2)$ denote the normalized top-left and bottom-right coordinates of the bounding box, respectively. We normalize all bounding box coordinates to a range from 0 to 1000 to maintain consistency across different image scales. This approach allows the masked regions to be seamlessly integrated with textual content in pixel and text space, forming complete sentences within the same tokenization space. In sum, each region `region{i}` is represented as: `region{i} <mask> <pos> <|box_start|>(x1, y1),(x2, y2)<|box_end|>`.

B.3. Model Architecture

Figure 10 illustrates the architecture of ROBIN, comprising an image encoder, a mask encoder, and a text encoder. These components embed their respective inputs before processing by the Large Language Model (LLM) backbone (Qwen2.5-3B[7]). Specifically, the image encoder, based on ConvNeXt-Large [27], generates a representation of the global image that replaces the image token in the textual input. Similarly, the mask encoder produces embeddings for each segmentation mask, which replace the corresponding mask tokens. The text encoder processes the textual input as it is. These embeddings are then passed to the LLM for processing.

[7] https://huggingface.co/Qwen/Qwen2.5-3B-Instruct

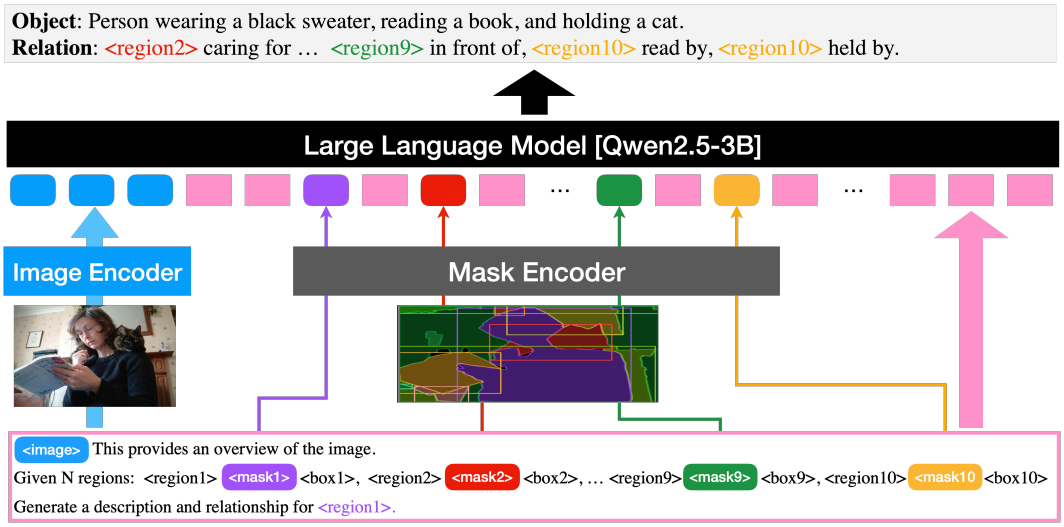


Figure 10. The architecture of ROBIN, which consists of a visual encoder that embeds the global image, a mask encoder that embeds segmentation masks individually, and a text encoder that encodes the textual input. The image token and mask tokens in the text input are replaced by embeddings from the image encoder and mask encoder, respectively.

Our training setup supports up to a maximum of 99 regions per image, facilitated by our model's extended sequence length of 8192 tokens. The ability of current LLMs to comprehend long contexts enables the model to capture detailed region-specific information essential for complex visual understanding tasks.

C. Training Dataset

In Table 7 and 8, we present more details for the dataset used in our instruction tuning described in Section 3.2. For scene graphs, we use maximum of 20 relations per object. We use the SVG initially curated for Dense Object & Relationship Generation only in Stage 1, and the full synthetic scene graph used for self-distillation in Stage 2 training (SVG-Stage2) only in Stage 2. This results in a total of 1.98M examples used in Stage 1 training and 1.19M examples in Stage 2 training. The input instruction and output text format for Grounding and Scene Graph tasks are additionally shown in Table 9-11.

Dataset Type	Task	# Examples	Datasets
Visual Instruction	VQA	427K	VQAv2 [19], GQA[25], GQA-CoT [11, 25], OKVQA [55] ChartQA[56], DocVQA [57], DVQA [30], TallyQA [2]
	Conversation	321K	LLava-Instruct [47, 95], Osprey-724k [92], LLaVAR [98]
	Multiple Choice	296K	AOKVQA [71], VCR [94], AI2D [33], ScienceQA [69], TQA [34]
	Captioning	213K	ShareGPT-4o [7], Flickr30K-Entities [11, 62], , LAION-GPT4V [98]
	OCR	160K	OCRVQA, SynthDoc-En, ST-VQA, TextCaps[72]
Grounding	Referring Expression Comprehension	110K	RefCOCO, RefCOCO$^+$, RefCOCOg [90]
	Region Captioning	77K	VG Region Captions [37]
Scene Graph	Scene Graph Detection	118K	VG [37], PSG [84]
	Scene Graph Classification*	87K	VG [37], PSG [84]
	Dense Object & Relationship Generation[1]	171K	SVG-RELATIONS (Ours)

Table 7. Overview of dataset used in Stage1 training. See Table 9-11 for input and output text format used in Grounding and Scene Graph tasks. *We merge scene graphs in VG and PSG by identifying the duplicate regions based on the IoU similarity (threshold > 0.5) and reassign their relationships with the new set of non-duplicate regions, resulting in 87K unique images for this task.

D. Evaluation Details

This section provides detailed information on the evaluation configurations used for the visual reasoning tasks presented in our experiments. Unless explicitly stated otherwise, all models are evaluated using greedy decoding (i.e., selecting the token with the highest probability at each step) without sampling. The sole exception is the scene graph detection task, where we employ a temperature of 0.2 and a top-p value of 1.0 during generation [23] .

Dataset Type	Task	# Examples	Datasets
Visual Instruction	VQA	377K	VQAv2 [19], GQA[25], GQA-CoT [11, 25], OKVQA [55], TallyQA [2]
	Conversation	144K	Osprey-724k [92]
	Multiple Choice	83K	AOKVQA [71], AI2D [33], ScienceQA [69], TQA [34]
Grounding	Referring Expression Comprehension	110K	RefCOCO, RefCOCO$^+$, RefCOCOg [90]
	Region Captioning	77K	VG Region Captions [37]
Scene Graph	Scene Graph Detection	118K	VG [37], PSG [84]
	Dense Scene Graph Generation[2]	226K*	SVG-FULL (Ours)

Table 8. Dataset used in Stage 2 training with 1.19M instances. *We include two variants: one includes bounding box text coordinates, while other omits them when identfying the region.

Referring Expression Comprehension

Input: Please provide the region this sentence describes: small boy in black shirt.

Output: `<|box_start|>(366,515),(443,742)<|box_end|>`

Region Captioning

Input: Give me a short description of:
`<region> <mask><pos> <|box_start|>(366,515),(443,742)<|box_end|>`

Output: small boy in black shirt.

Table 9. Example input and output for Referring Expression Comprehension and Region Captioning in Grounding task.

D.1. Relationship Reasoning

We evaluate relationship reasoning using various benchmarks, each with specific evaluation protocols:

GQA [25]: The model's performance is assessed by the accuracy of exact matches between its answers and the ground truth answers. The prompt provided to the model is as follows: *Question:* {`question`} *Answer the question using a single word or phrase.*

Visual-Semantic Reasoning (VSR): We frame the VSR task as a binary classification problem by asking the model yes/no questions if the mentioned subject, relation, and object triplet holds true in the image. The model's responses are evaluated for correctness in indicating true or false statements. The prompt is: *Question: Is the* {`subj`} {`relation`} {`obj`}*? Answer with yes or no.*

Other Benchmarks (MMBench, SeedBench-Image, CRPE, What's Up): For these multiple-choice tasks, we prompt the model to select the correct option from a set of lettered choices. The model's accuracy is measured based on its selection of the correct option. The prompt used for multiple choice benchmarks is: *Question:* {`question`} *Choices:* {`choices`} *Answer with the option's letter from the given choices directly.*

D.2. Referring Expression Comprehension

For the referring expression comprehension task, the model is prompted to identify the region described by a given sentence. The prompt is: *Please provide the region this sentence describes:* {`sentence`}

The model's response is expected to contain bounding box coordinates corresponding to the described region. We extract these coordinates and compare them against ground truth annotations for evaluation.

D.3. Region Classification

In the region classification task, we aim to determine the category of a given region in the image. The model is prompted as follows: *What is the category of* `<mask><pos> <box>` *? Answer using only one word or phrase.*

Here, `<mask><pos>` represents the segmentation mask and positional information, and `<box>` is the bounding box coordinates. Unlike Osprey [92], which uses only the segmentation mask, we incorporate bounding box information inferred from the mask to enable more precise object localization.

To evaluate the model's predictions, we use the Sentence-BERT[8] model [68] to compute the semantic similarity between the predicted category and the ground truth class. We encode both the predicted and ground truth class names and compute the cosine similarity between their embeddings.

D.4. Panoptic Segmentation

For the panoptic segmentation task, we employ the ADE20K dataset [101]. The model is prompted with: *What is in* `<mask><pos>` *? Answer using a short phrase.*

After obtaining the model's response, we map the predicted description to the closest object class using semantic similarity, as in the region classification task, utilizing the same Sentence-BERT model. We encode the object class names and compute the cosine similarity between the embeddings of the predicted and ground truth classes. Standard segmentation metrics, including those for panoptic, instance, and semantic segmentation, are used to evaluate the model's performance[9].

D.5. Scene Graph Detection

In the scene graph detection task, the model is prompted to generate a scene graph, including object regions and their relationships. The prompt is: *Generate a scene graph with proposed regions in bounding box coordinates.*

An illustrative example of the expected input and output format is provided in Table 10. We extract the bounding box regions, object descriptions, and relationship triplets from the model's output using regular expressions. To assign labels in the Panoptic Scene Graph (PSG) dataset, we match the object descriptions to the closest object classes by computing the maximum cosine similarity between their embeddings, as in the panoptic segmentation task. Specifically, we encode phrases of the form "The object is {`class_name`}" for both the ground truth and predicted object descriptions.

For determining the best predicate (relationship), we encode the relationships using the phrase "Object is {`predicate`} another object" and compute the cosine similarity between the embeddings of the predicted and ground truth relations.

Following the recall-based evaluation protocol in scene graph literature [37], a predicted triplet `(subject, predicate, object)` is considered correct if:

1. The predicted subject and object bounding boxes have an Intersection over Union (IoU) greater than 0.5 with the ground truth bounding boxes.
2. The predicted subject class, predicate, and object class match those of the ground truth.

E. Synthetic Visual Genome Data Generation Details

In this section, we provide more details of the data generation pipeline. Figure 11) shows an overview of data curation for.

E.1. Prompts for Data Generation

We provide the prompts for calling GPT-4V to curate both stages of SVG in Tables 12. and 13

E.2. Data Filtering Details

We show impacts of data filtering described in Stage 1. Table 15 shows the relationship reasoning performance of the Stage 1 model trained with and without filtering. We see that filtering is crucial for achieving high performance and can lead to significant drop in relationship reasoning without the module. We next describe our filtering approach used to curate the data.

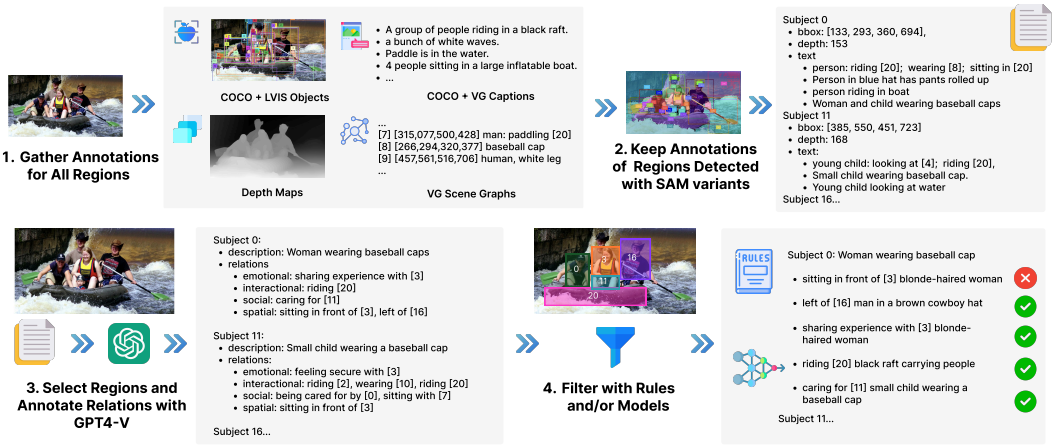


Figure 11. The generation pipeline for the first stage of SVG dataset consists of four steps: (1) gathering annotations for all the regions; (2) keeping only the annotations of the regions detected by SAM-SEEM; (3) selecting and annotating regions with GPT4-V and (4) filtering out incorrect relations with rules and/or vision-language models such as CLIP or LLaVA.

Rule-based filtering We implement seven simple rules – above, below, left, right, overlap, above or overlap, and below or overlap – based on naive physics knowledge. These rules cover 22 frequent spatial relationships, which represent 42.3% of all generated relationships. We present the definitions of all 7 rules as well as the rules used for the 22 frequent relation phrases in Listing 1. We verified the validity of these rules by getting the rule-based scores of the 102,233 Visual Genome examples containing the 22 relation phrases. Assuming Visual Genome's human annotations are correct, we treat all examples as positive. Our rule-based program predicts 1 for 95.1% of these examples, meaning that its accuracy is 95.1%.

Model-based filtering For relations that cannot be easily verified through rules such as functional and emotional relationships, we rely on model-based filtering using multi-modal models.

We first test out existing MLMs and formulate the verification as a binary VQA task, asking "Does this relation correctly describe the image? Answer with Yes or No.". We experiment with three state-of-the-art multi-modal models – LLaVA, QwenVL, and CogVLM [3, 49, 76] – and different combinations of their answers. Next, we try filtering with CLIP based models, and calculate the similarity scores between each "subject relation object" phrase and the region crop containing the subject and object and filter out examples with similarity scores lower than 0.3 [70].

We compare these models' performance on (1) existing image-text evaluation datasets that require relation understanding such as SugarCREPE [24] and CREPE [53] and (2) a small subset (N = 600) of the data with one author's annotations. We present these results in Tables 16 and 17 respectively. Through these experiments, we find it most optimal to use both LLaVa-v1.6-vicuna-13b and Qwen-VL-Chat and filter out relationships where either model answers "No". Meanwhile, we find CLIP-based filtering ineffective for most relationship types, and not use their filtering scheme.

Relationship distribution before and after filtering We present the distributions of the most frequent relationships for each type of relationships before and after running our filtering pipeline in Figures 12 - 16.

F. Object Proposal Generations

- SAM [36]: We explore the subpart, part, and whole object modes available in the SAM model as region candidates for scene graph generation. We found the part and whole mode to be relevant for covering interesting objects in the scene.
- Semantic-SAM [40]: To extract more semantic-aware object proposals, we also employ Semantic-SAM at different levels of granularity. Specifically, 3 different granularity prompts were used, (1) granularity prompt 1, (2) granularity prompt 2 and (3) all 6 granularity prompts, as introduced in [40], where granularity prompts 1, 2 correspond to different semantic-level object proposals, while all 6 granularity prompts give more fine-grained, part-like region proposals.
- Proposal Refinement via Union Strategy: We take the union of aforementioned two SAM and three Semantic-Sam configurations to get the final candidates. We perform non-maximum suppression with IoU threshold of 0.6. We then sort the objects based on their area instead of stability and detection scores and take the K largest objects, which captures objects with greater semantic significance in general.

[8] sentence-transformers/all-MiniLM-L6-v2
[9] https://github.com/facebookresearch/detectron2/blob/main/detectron2/evaluation/panoptic_evaluation.py

```python
from shapely.geometry import box
from functools import partial

def get_obj_center(obj: Dict):
    center = ((obj["bbox"][0] + obj["bbox"][2]) / 2, (obj["bbox"][1] + obj["bbox"][3]) / 2)
    return center

def above(subj: Dict, obj: Dict):
    subj_c = get_obj_center(subj)
    obj_c = get_obj_center(obj)
    return subj_c[1] < obj_c[1]

def below(subj: Dict, obj: Dict):
    subj_c = get_obj_center(subj)
    obj_c = get_obj_center(obj)
    return subj_c[1] > obj_c[1]

def left(subj: Dict, obj: Dict):
    subj_c = get_obj_center(subj)
    obj_c = get_obj_center(obj)
    return subj_c[0] < obj_c[0]

def right(subj: Dict, obj: Dict):
    subj_c = get_obj_center(subj)
    obj_c = get_obj_center(obj)
    return subj_c[0] > obj_c[0]

def overlap(subj: Dict, obj: Dict):
    subj_bbox = box(minx=subj["bbox"][0], miny=subj["bbox"][1], maxx=subj["bbox"][2], maxy=subj["bbox"][3])
    obj_bbox = box(minx=obj["bbox"][0], miny=obj["bbox"][1], maxx=obj["bbox"][2], maxy=obj["bbox"][3])
    return subj_bbox.intersects(obj_bbox)

def below_or_overlap(subj: Dict, obj: Dict):
    return below(subj, obj) or overlap(subj, obj)

def above_or_overlap(subj: Dict, obj: Dict):
    return above(subj, obj) or overlap(subj, obj)

rules = {
    "above": partial(above),
    "below": partial(below),
    "under": partial(below_or_overlap),
    "underneath": partial(below_or_overlap),
    "beneath": partial(below_or_overlap),
    "left of": partial(left),
    "to the left of": partial(left),
    "on the left of": partial(left),
    "right of": partial(right),
    "to the right of": partial(right),
    "on the right of": partial(right),
    "contains": partial(overlap),
    "in": partial(overlap),
    "inside": partial(overlap),
    "inside of": partial(overlap),
    "on": partial(above_or_overlap),
    "has on it": partial(above_or_overlap),
    "on top of": partial(above_or_overlap),
    "has on top": partial(above_or_overlap),
    "covered by": partial(below_or_overlap),
    "covering": partial(above_or_overlap),
    "over": partial(above_or_overlap),
}
```

Listing 1. Definitions of filtering rules for spatial relations.

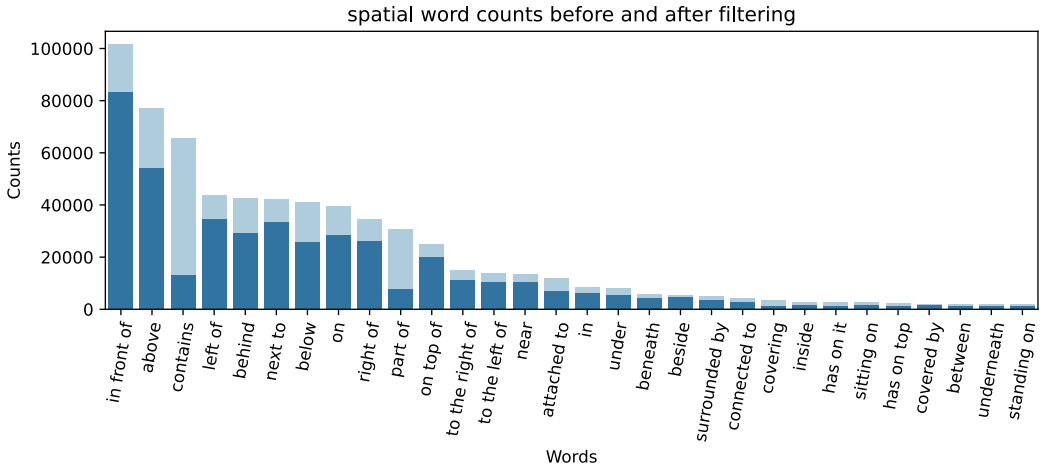


Figure 12. Distribution of spatial relations before and after filtering

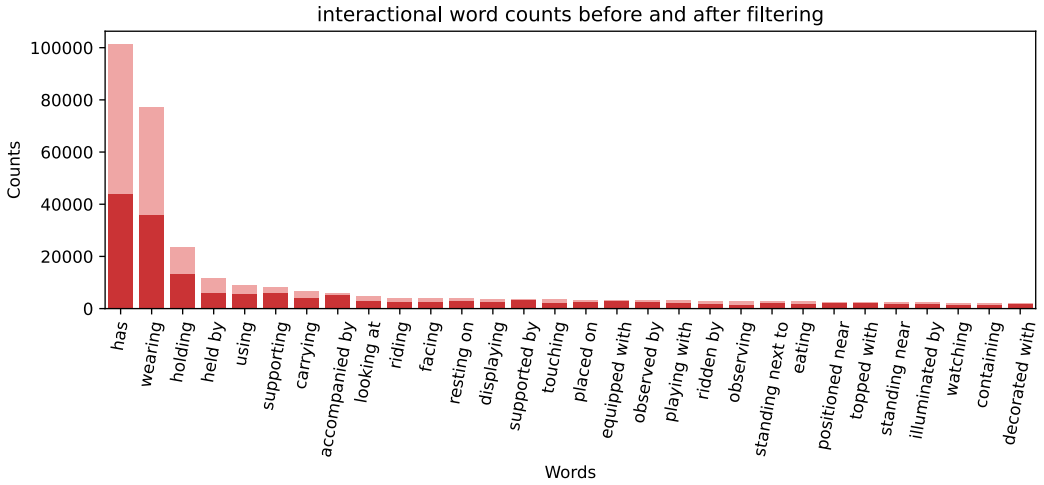


Figure 13. Distribution of interactional relations before and after filtering

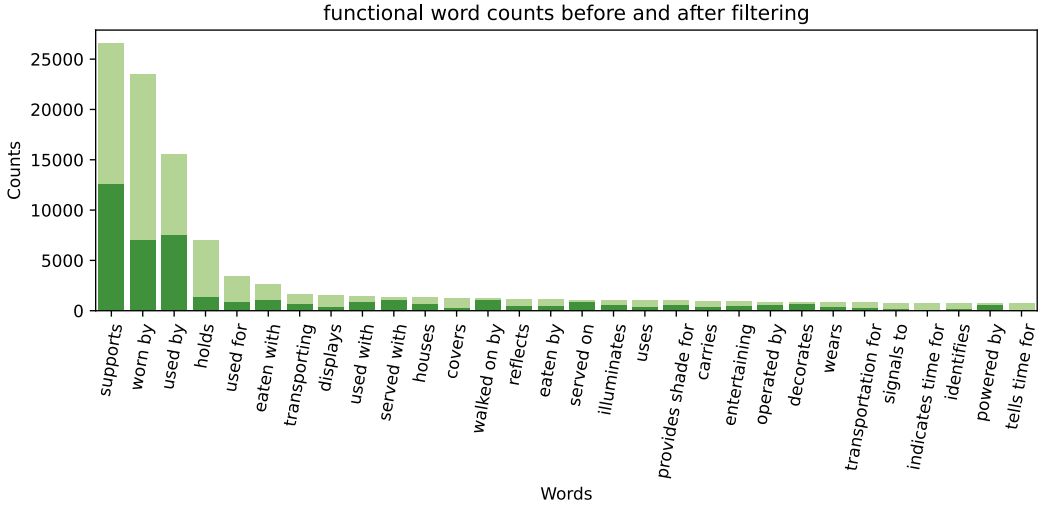


Figure 14. Distribution of functional relations before and after filtering

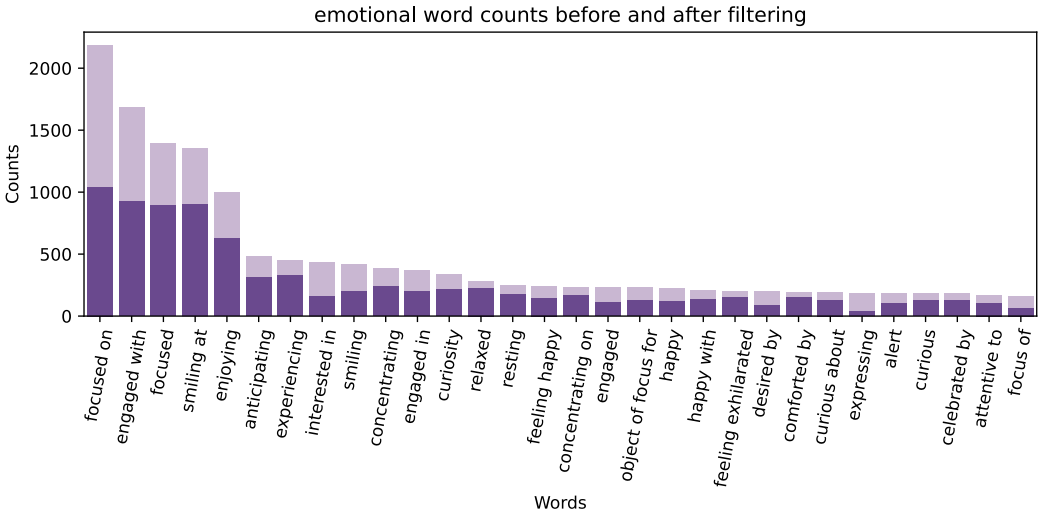


Figure 15. Distribution of emotional relations before and after filtering

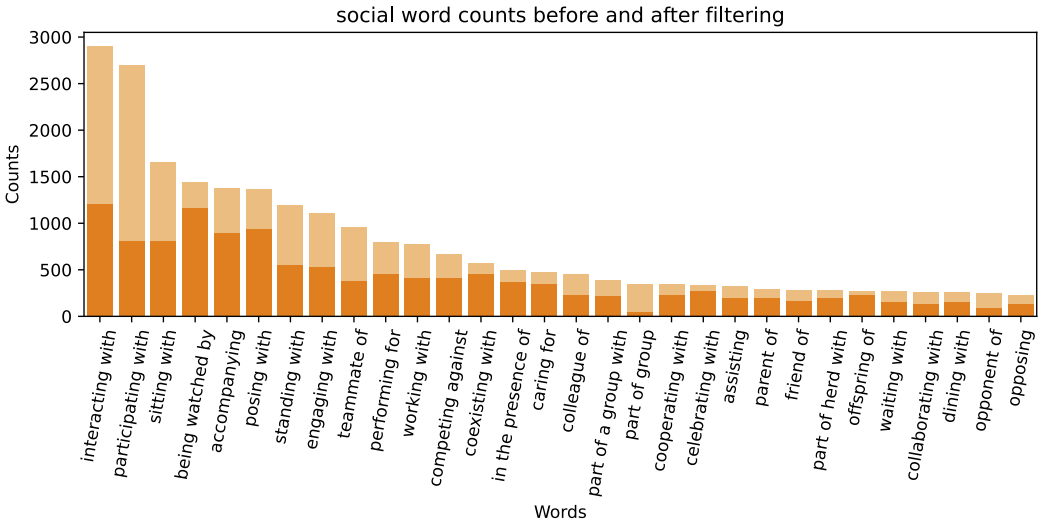


Figure 16. Distribution of social relations before and after filtering

G. Broader Impacts

Scene graphs provide useful annotations that can be applied to beyond 2D reasoning, as demonstrated in embodied environments by our experiments. The annotation schema can be broadened to include reasoning over the shapes and structures of 3D objects, enhancing consistency in video understanding, and facilitating image generation with controllable layouts. Accurately creating scene graphs in real-time videos has beneficial implications in fields such as autonomous driving and in helping MLLMs generate culturally sensitive and appropriate remarks by understanding social relationships among people. However, inferring personal relationships and emotional states from scene graphs raises significant privacy and ethical concerns. It is crucial to handle such data responsibly to prevent potential misuse, such as unauthorized surveillance or psychological profiling.

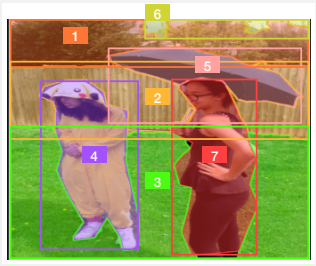
Scene Graph Detection

Input: Generate scene graph with proposed regions in bbox coordinates.
Output:
Objects:
region1: tree <|box_start|>(0,0),(1000,204)<|box_end|>
region2: fence <|box_start|>(0,175),(1000,504)<|box_end|>
region3: grass <|box_start|>(0,445),(1000,1000)<|box_end|>
region4: person <|box_start|>(102,253),(431,964)<|box_end|>
region5: umbrella <|box_start|>(327,117),(977,435)<|box_end|>
region6: sky <|box_start|>(447,0),(1000,84)<|box_end|>
region7: person <|box_start|>(539,246),(827,985)<|box_end|>
Relations:
region2: region1 in front of
region4: region3 standing on, region7 beside
region6: region1 over
region7: region3 standing on, region5 holding

Scene Graph Classification

Input: In the image, there are 7 regions, such as
region1 <mask> <pos> <|box_start|>(0,0),(1000,204)<|box_end|>,
region2 <mask> <pos> <|box_start|> (0,175),(1000,504) <|box_end|>,
...
region7 <mask> <pos> <|box_start|> (539,246),(827,985) <|box_end|>.
Generate scene graph for given regions.

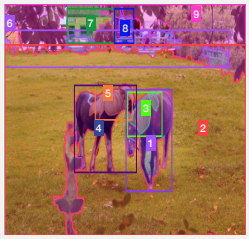
Output:
Objects:
region1: tree
region2: fence
region3: grass
region4: person
region5: umbrella
region6: sky
region7: person



Relations:
region2: region1 in front of
region4: region3 standing on, region7 beside
region6: region1 over
region7: region3 standing on, region5 holding

Table 10. Example input and output for Scene Graph Detection and Scene Graph Classification.

Dense Object & Relationship Generation



Input: In the image, there are 9 regions, such as:
region1 <mask><pos> <|box_start|>(0,172),(1000,1000)<|box_end|>,
region2 <mask><pos> <|box_start|>(579,73),(617,113)<|box_end|>,
...
region9 <mask><pos> <|box_start|>(0,0),(1000,1000)<|box_end|>.
Generate a description for: region4.

Output: A dark brown horse with white spots and a mane.

Input: Generate a list of relationships for: region4.

Output: region5 has, region2 stands on top of, region1 nudging, region1 left of, region6 in front of

Dense Scene Graph Generation

Input: In the image, there are 10 regions, such as:
region1 <mask><pos> <|box_start|>(222,19),(894,1000)<|box_end|>,
region2 <mask><pos> <|box_start|>(346,187),(820,566)<|box_end|>,
...
region10 <mask><pos> <|box_start|>(0,480),(392,912)<|box_end|>.
Generate an extremely detailed scene graph for given regions.

Output:
Objects:
region1: person wearing a black sweater, reading a book, and holding a cat.
region2: cat sitting on woman's shoulder.
...
region10: book being read by a woman.
Relations:
region1: region2 caring for, region4 in front of, region9 in front of, region10 holding, region10 engaged with, region10 reading, region10 looking at
region2: region1 being comforted by, region1 attached to, region1 sitting on
...
region10: region1 read by, region1 held by, region7 in front of

Table 11. Example input and output for Dense Object & Relationship Generation and Dense Scene Graph Generation.

Prompt for Dense Object & Relationship Generation

System Prompt: You are provided 'Captions', 'Region Captions', 'Scene Graph', and 'QAs' annotated by humans, which are guaranteed to be correct.
- [Captions]: General image overviews.
- [Region Captions]: Detailed descriptions of particular image regions.
- [Objects]: Grounded objects annotated by humans. In format of [object_id (int)]: [bbox] object_name
- [Scene Graph]: In format of [object_id (int)]: ([bbox], optional depth) [list of attributes] object_name [relationship with other objects]
Bounding boxes (bbox) are labeled in the form of bounding box as [x1,y1,x2,y2] normalized from 0 to 1000. You might also have depth information from 0 to 255, where low depth means it's farther back in the image.
Objective: Your task is to meticulously document all discernible relationships between a selected object and other objects within the scene, and provide a dense, detailed description of the object as well. This involves a comprehensive examination of the scene to identify and record every possible, interesting relationship that the chosen object has/ may have with others. These relationships can be of various types, including but not limited to interactional, (e.g., holding, looking at), functional (e.g., part of, used by, owned by, contains), hierarchical, and spatial relationships (e.g., above, behind, attached to, etc.).
Please specify the category of relationship you are looking to annotate and then provide the list.

Relation Requirement: - Try to include at least 10 total relationships per object, describing beyond what is said explicitly in the descriptions if needed, by reasoning about the bounding boxes and depth.
- Prioritize including interactional and action-based relationships first.
- Prioritize human-annotated captions for accuracy and consistency, and carefully consider model-generated descriptions for additional details.
- Categorize your relationships into one of the following categories:
1. **Spatial Relationships**: These refer to the physical location or position of the person in relation to other objects or individuals. Examples include 'above', 'below', 'over', 'across from', 'behind', 'in front of', 'inside', 'outside' etc.

2. **Interactional Relationships**: These involve some form of action or interaction between the person and other objects or individuals. Examples include 'holding', 'touching', 'looking at', 'talking to', 'playing with', 'using', etc.

3. **Functional Relationships**: These refer to the purpose or function of an object in relation to the person. Examples include 'worn by', 'used by', 'owned by', 'part of', etc.

4. **Social Relationships** (for human): These refer to the social connection or interaction between the person and other individuals. Examples include 'friend of', 'sibling of', 'parent of', 'colleague of', 'boss of', etc.

5. **Emotional Relationships** (for human): These refer to the emotional connection or feelings between the person and other individuals. Examples include 'loves', 'likes', 'dislikes', 'hates', etc.

Description Requirement: - The description should cover the interesting features and focus on relationships with other objects. - Your description should be factual and to the point. It SHOULD NOT have suggesting, speculative or interpretative additions about the object's significance or its contribution to the scene's atmosphere and vibe. - **Physical Attributes**: Briefly detail the object's color, size, texture, attributes, and any other pertinent physical characteristics. - **Position**: Describe the object's location within the scene with precision. - **Relationships with Other Objects**: Document all direct spatial and interactional relationships the object has with other objects. This includes being next to, above, below, holding, or any other clear relationship. Each relationship should be stated plainly, specifying the type of relationship and the related object's ID. - Avoid including obvious traits/properties of the object that can be inferred without looking at the image.

Table 12. Prompt for collecting SVG-RELATIONS.

Prompt for Scene Graph Edit

System Prompt: You are given the original image and a set of same images with highlighted regions specified by unique id labels (colors matching the referred region). You are provided:
- dense caption of the image that describes the scene as detailed as possible.
- scene graph **generated** by AI model that might contain hallucinated objects and relationships.

Your goal is to fix and improve this AI generated scene graph. Each object in the scene graph is labeled with a unique numeric id and provided with bounding box coordinates ‘[x1, y1, x2, y2]‘ (top-left and bottom-right). We want to verify the correctness of the objects and relationships in the scene graph. The input scene graph has the format:
’obj_id’: ’name’: ’A description of the object’, ’bbox’: [x1, y1, x2, y2], ’rel’: ’rel_name’: [integer list of other obj_ids], ,

Your task: Generate only the ”Edits” needed to correct the objects and relationships in the scene graph. Your response will be called by python dictionary ‘update‘ method, so make sure you only generate just the NECESSARY edits.

1. Go through all the objects in the scene graph and determine if you can confidently say the object mentioned by noisy description is really present and visible in the image.
- If you think you can confidently verify the presence of an object and the description can be improved, provide a revised description with more accurate object name and more visual specificity.
- If you are unable to confidently verify the presence of an object or suspect it might be inaccurately represented for the given region, please provide a revised description using a more accurate object name if possible, or alternatively, classify it under a broader, safer category. These can be ’decoration’, ’tool’, ’foliage’, ’furniture item’, etc., to ensure clarity and avoid misidentification. Then, provide more specificity to your name so that viewer can disambiguate the object from other objects.
- Try to include the object’s position, color, texture, and/or any other distinguishing feature.
- Remember to use the first original image to accurately identify the color and texture of the objects.
- Make sure to take the bounding box size and location into account to correctly describe the specified region.

2. Then, make edits to the relationships by adding the prominent relationships current scene graph is missing, or/and removing erroneous relationships between objects in the scene. For example, object 1 has relationship: ’on’: [2,3,4,5] and you think it should be ’on’: [2,3], then you should include object_ids [4,5] in the ’remove’ list. Also, if object 2 has relationship: ’contains’: [2,6,7] and you think it should be ’contains’: [2,6,7,8], then you should include object_ids [8] in the ’add’ list, and also add the relationship type ’spatial’ to the ’add’ list. If no relationships need to be added and/or removed, please return an empty list for ’add’ and/or ’remove’.
Output should be a flattened JSON object with no unnecessary spacing.

Table 13. Prompt to create SVG-FULL using SG-EDIT

	Version	Images	Relations	Objects	Relations per region / subject	Spatial / functional / interaction / social / emotional relations
Visual Genome [37]	COCO images	38K	843,576	957,951	0.88 / 1.53	—
SVG-Relations	Raw	33K	1,457,849	626,516	2.33 / 6.50	702,916 / 189,262 / 500,240 / 34244 / 31,187
	Filtered		855,573	445,131	1.92 / 4.98	460,098 / 81,789 / 277,029 / 18,496 / 18,161

Table 14. Effect of filtering pipeline while curating SVG-RELATIONS dataset.

Dataset	VSR ZS-test	CRPE Relation	SugarCrepe Relation	What’s Up ? Controlled
SVG without Filtering	67.8	59.8	86.8	67.0
SVG with Filtering	69.7	64.8	87.9	76.7

Table 15. Results on relationship reasoning benchmarks comparing the Stage 1 model trained with and without filtering of the SVG dataset. We keep the same set of instruction tuning training data in Stage 1 but only change the Dense Object & Relationship task.

Dataset	Total # of examples	Model	Balanced acc	Precision
SugarCREPE (replace relation)	6,354	Best CLIP	0.7475	—
		LLaVa	0.7224	0.6583
		Qwen	0.7398	0.6703
		CogVLM	0.6139	0.5652
		LLaVa + Qwen	**0.7581**	**0.7034**
		Majority	0.7181	0.6444
CREPE (systematicity-laion)	47,664	LLaVa	0.7698	0.7335
		Qwen	0.7605	0.6955
		CogVLM	0.5377	0.5228
		LLaVa + Qwen	**0.7954**	**0.7788**
		Majority	0.7361	0.6672

Table 16. We report the balanced accuracy and precision of different model-based filtering methods on existing benchmarks – SugarCREPE and CREPE – that require relation understanding.

Relation types	Total # of examples	Method	Balanced accuracy	Precision
All	600	CLIP similarity >0.3	0.5008	0.7089
		LLaVa VQA	0.6163	0.8099
		Qwen VQA	0.5486	0.7319
		CogVLM VQA	0.5029	0.7095
		LLaVa + Qwen	**0.6165**	**0.8165**
		Majority VQA	0.5484	0.7311

Table 17. We report the balanced accuracy and precision of different model-based filtering methods on the human-annotated set of 600 randomly sampled examples across all five relation types.