

# VIZOR: Viewpoint-Invariant Zero-Shot Scene Graph Generation for 3D Scene Reasoning

Vivek Madhavaram<sup>1\*</sup>

Vartika Sengar<sup>2\*</sup>

Arkadipta De<sup>2\*†</sup>

Charu Sharma<sup>1</sup>

<sup>1</sup>Machine Learning Lab, IIIT Hyderabad, India

<sup>2</sup>Fujitsu Research India, Bangalore

vivekvardhan.m@research.iiit.ac.in, vartika.sengar@fujitsu.com, charu.sharma@iiit.ac.in

## 1. Implementation Details and Computation

**Implementation:** We use Mask3D [7] as a segmentation model. For front-direction prediction, to create a reference database, we use Shap-E [5] as a generative model with guidance-scale set to 15.0 and other parameters set to default<sup>1</sup>. For obtaining multiple views of each object, we set the camera on the circular rig with a radius of 1.5 times the bounding box extent of the object with the object centroid as the center of the rig and take 12 different snapshots. For all of our experiments, we use two different LLMs: Llama [3] 3.2 11B Vision Instruct<sup>2</sup> and GPT-4o<sup>3</sup> [1]. We feed the top and front views of each object and 6 additional views to the LLMs to predict different object (node) attributes. We put the maximum output token limit as 500 for Llama.

**Hardware Requirements:** VIZOR has very low hardware and memory requirements. It needs a memory of 11GB from 1GPU to generate a scene graph for a particular scene. Out of 11GB, 8GB is used by Shap-E for reference object generation, which is a one-time task, and the rest of the memory is used by other processes. We executed this code on a 16 GB NVIDIA RTX A4000 server.

**Computation Cost:** We perform computation cost profiling of VIZOR with respect to one of the current state-of-the-art ConceptGraphs [4]. In terms of runtime, ConceptGraph takes 27 minutes on average to finish the entire process for a scene and outputs a relationship graph having 16 edges on average. While VIZOR takes 29 minutes on average to complete its process for a scene and outputs relationships between all object pairs in the scene (i.e., 275 edges on average). Since both method uses LLM, we measure the token usage comparison for the two methods. ConceptGraph uses 1,49,481 tokens for a scene while VIZOR uses 9,17,592 tokens on average, i.e., **6 times more tokens w.r.t Conceptgraph**, while processing 275 edges on average, i.e., **27 times more relations**. This comparative

study provides empirical evidence about the efficiency of our method.

## 2. Rules for Geometric Relations

In this section we will elaborate the rules for generating the initial geometric relations. We first take the front direction of an object as reference to calculate horizontal relations. Then, we calculate a vector joining the centroids of a subject and object (from object to subject). We use this vector to calculate the angle it makes with the front direction of object node in the plane parallel to XY plane. If the angle lies between 0 - 10° or 350-360°, the relation is defined as “front”. Similarly, if the angle lies between 170 - 190°, it is defined as “behind”. If the angle lies between 10 - 30°, relation is “left front” and relation is “left behind” if the angle lies between 150 - 170°. On the right side, relation is “right front” if the angle is between 330 - 350° and “right behind” if the angle lies between 190 - 210°. The relation is “left” if the angle is within the range of 30-150° and “right” for range of 210-330°.

To calculate the vertical relations, we consider centroids and bounding box extents. We calculate the vertical distance between both centroids and evaluate it with bounding box extents. If both the values are approximately same and subject centroid is above that of object, we consider the relation to be “on”. Cases involving objects like bed, couch, chair are dealt in different way as the distance between centroids will be far less than bounding box extents. Similarly, if the vertical distance between centroids is more than bounding box extents, and subject lies above the object, relation is defined as “above” else “below” as shown in Figure 1.

## 3. Human Evaluation Details

Since VIZOR generates open vocabulary relationships in the scene graph and no ground truth for such relations is available, we took a step in evaluating this model with human intervention. Starting with front direction and attribute prediction, node captioning, relationship generation,

<sup>1</sup><https://github.com/openai/shap-e>

<sup>2</sup><https://huggingface.co/meta-llama/Llama-3.2-11B-Vision-Instruct>

<sup>3</sup><https://openai.com/index/gpt-4o-system-card/>

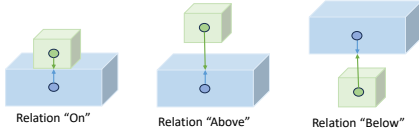


Figure 1. Scenarios for defining vertical relations. If subject is above the object and the distance between centroids is similar to bounding box extents, relationship is defined as “on”, else relationship is “above”. If the subject is below the object and bounding box extents do not match distance between centroids, relationship is defined as “below”.

and open vocabulary object grounding, we provided background information to human evaluators as a benchmark to judge the results. Note that these evaluations are carried out on the Replica dataset [8].

For node captioning, we provided cropped fragment of each object and presented the captions generated by both Llama [3] and GPT4 [1]. We asked the evaluators to mark each caption as either ‘0’ or ‘1’ for wrong and correct predictions respectively. We collected responses from 3 evaluators for each caption and marked each caption as correct if 2 of 3 evaluators agreed. The same process is carried out in evaluating front direction prediction where the front view of each node is presented to 3 evaluators and responses are collected.

In the case of attribute prediction, we evaluated the efficiency of LLMs (Llama [3] and GPT4 [1]) using attributes predicted from single and multiple views of each node. We prepared a questionnaire for each scene, where we initially presented the image of each object followed by sections for each attribute. We mainly considered two attributes namely “Color” and “Geometry”. In each section, we presented the predictions of each model in random order, given no clue about the model. We asked each user to rate the prediction on a scale of “1-5”, 1 being “Poor” and 5 being “Excellent”. We collected the responses, aggregated them, and presented the results in Table 6 of the main paper. A sample of the questionnaire is shown in Figure 2. All the users who participated in the survey belong to the age group 22 - 30 comprising both males and females, having a research background and minimum educational qualification being a master’s degree. There are a total of 8 questionnaires, each for a particular scene from the Replica dataset [8] and each questionnaire is distributed randomly to a set of users, making sure each form has responses from at least 3 users.

To evaluate generalized relationships, we provided a scene to each user, few 3D points, each point depicting the front vector head of a particular instance and all the cropped instances. We asked the user to take the front vector head (head is the camera position capturing the front view of an instance and tail is the centroid of that instance) as reference and evaluate the relationships. We consider a relation


is valid if 2 of 3 users consider it to be valid since the relations can be subjective as they contain open vocabulary. Similarly, for open vocabulary object grounding, we provided them with group of queries discussed in main paper containing complex queries, also mentioned in Section 7 and output instance of every query along with the 3D scene. We asked them to mark ‘0’ or ‘1’ for wrong and correct answers, aggregated the results and presented in main paper.

## 4. Detailed Comparison with Zero-Shot Methods

In this section, we discuss different state-of-the-art zero-shot 3D visual grounding methods in conjunction to VIZOR from a comparative perspective. Existing zero-shot 3D visual grounding methods differ significantly from our proposed approach. ZS3DVG [11] employs visual programming combined with large language models to infer object grounding without training, but it defines relations using axis-aligned coordinates, which makes spatial reasoning heavily *viewpoint-dependent* and thus often semantically incorrect. SeeGround [6] mitigates some of these issues by aligning textual queries with 3D object proposals through language–feature matching, yielding improved retrieval, but it lacks an explicit structured representation of the entire scene and therefore cannot perform reasoning over inter-object relations. VLM-Grounder [10] leverages vision-language models in a CLIP-style embedding space for direct similarity-based retrieval, which is efficient but ignores global scene layout and fails to capture nuanced spatial or relational dependencies. In contrast, VIZOR constructs explicit, dense 3D scene graphs that are both view-invariant (by modeling spatial relations relative to each object’s front direction) and attribute-rich (aggregating multi-view descriptors such as geometry, color, functionality, and captions). This structured representation allows VIZOR to go beyond object retrieval, supporting complex reasoning tasks such as affordance-based queries (“something to sit on”), negation (“a place to sit but not lie down”), and compositional spatial queries requiring multi-step inference. Consequently, VIZOR achieves consistent improvements over prior zero-shot methods, offering a more generalizable and semantically coherent framework for 3D scene understanding. In Table 1, we summarize the comparative study of zero-shot methods with VIZOR.

## 5. Additional Results

Apart from the qualitative results presented in the main paper, we evaluate VIZOR with ConceptGraphs [4] on more 6 complex queries and show the results here in Figure 3. Scene graphs generated on the ScanNet [2] and the 3DSSG [9] are presented respectively in Figure 4 and Figure 5. The scene graphs depict partially constructed scene graphs due



Property - Color

Below use cases describe the property - color from various models. Please rate the predictions based on above image.

light brown \*

☐ 1  
☐ 2  
☐ 3  
☐ 4  
☐ 5

Figure 2. **Sample Questionnaire:** Displays the image of node in the starting followed by ratings on properties.

Method	Key Idea / Approach	Scene Representation	Spatial Reasoning	Complex Queries	Limitations
ZS3DVG [11]	Visual programming with LLMs for zero-shot 3D grounding	Object proposals + simple spatial cues	Limited (axis-based)	✗	Lacks global scene structure; view-dependent reasoning
SeeGround [6]	Aligns language with 3D object features	Instance proposals + language alignment	Moderate	✗	Focused only on grounding, not full scene reasoning
VLM-Grounder [10]	Uses VLMs (CLIP-like) for grounding	Direct embedding similarity	Limited	✗	No explicit scene graph; weak at reasoning
<b>VIZOR (Ours)</b>	Training-free, view-invariant 3D scene graph with multi-view attributes + object-centric relations	Full 3D scene graph ( <i>nodes</i> are objects with attributes, <i>edges</i> are open-vocab relations)	Strong (view-invariant, object-oriented)	✓	Struggles with symmetric / incomplete objects

Table 1. Comparison of VIZOR with prior zero-shot grounding methods.

to space constraints and ease of understanding. Nodes in the graph are color-coded with the same colors as the objects, as highlighted in the 3D input scenes on the left of both figures. Qualitative comparison of VIZOR with 3DSSG scene graphs are shown in Figure 6. Missing relations in 3DSSG are represented using dotted lines and wrong relationships are denoted with red arrows.

## 6. Prompts and JSON Format

In this section we will discuss about the prompts provided for an Mutli-modal LLM from the proposed architecture in detail. The first use of Mutli-modal LLM is to predict the front direction of each instance in a scene. To predict the front direction, we provide 12 views around the object along with reference front view from database belonging to same instance class. Prompt for this step is shown in Figure 7.

Next usage of prompt to Mutli-modal LLM is for generating the attributes of each instance. We predict the attributes like color, functionality, geometry, affordance and finally the caption describing the object briefly. We initially predict these attributes using images of each view captured as mentioned in Section 3.2 from main paper. Prompt for generating the attributes is depicted in Figure 8 and the sample output JSON file of one instance for using this prompt is shown in Figure 9.

After predicting attributes of each objects from multiple views, all the output JSON files are fed to LLM for aggregating the attributes and generate summarized final attribute JSON file. Prompt for aggregating the attributes and the result can be seen in Figure 10 and Figure 11 respectively.

The third usage of prompt to LLM is for generalizing the relations to make them open vocabulary. Initially, all the relations are calculated using geometric rules and later they are generalized using LLM. The prompt to generalize the relations is displayed in Figure 12 and the resultant sample JSON file containing generalized relations is shown in Figure 13.

## 7. Complex Visual-Language Queries

The main advantage of VIZOR over ConceptGraphs [4] is having contextual information on the scene that helps in processing complex queries. We compare both methods on queries that are divided into 4 categories namely Descriptive, Affordance, Negation, and Complex-Spatial queries. While the first three categories were proposed by [4], we created the 4<sup>th</sup> category “Complex-Spatial” queries. ConceptGraphs [4] performed this evaluation on 2 scenes (room0 and office0) for the first 3 categories. Similarly, we created complex queries on these scenes and compared the models. Results are shown in Table 2 of the main paper. From the results, it is quite evident that in every category VIZOR outperformed [4]. Complex queries for each scene

are listed below.

### Complex queries for room0:

1. *Where can I sit so that I have a clear view of picture in front of me*
2. *Where can group of three people sit so that distance between them is minimum*
3. *Suitable place to sit so that cabinet is easily accessible*
4. *Select object for organizing eatables for party*
5. *Which object is easily accessible from all other objects*
6. *Place to sit near coffee table but far from door*
7. *Which object can be used for storing magazines*
8. *I want to sit but don't want to use a backrest, where can I sit?*
9. *Which is place that has close proximity with light resource.*
10. *I need to find some object to gift. Which one will be best in this room?*
11. *Where can I sit in so that I can read a book at night and I want to face the cabinet while sitting?*
12. *What object is in the center of the room, in front of the sofas?*
13. *What object is on the left side of the seating area?*
14. *What object is closest to the entrance?*
15. *What object is behind the sofa?*
16. *Which sofa has empty space on its right side?*
17. *What object is placed in front of cabinet?*
18. *What is the object with the most pillows on it?*
19. *What is the closest object which can illuminate the room when I enter the room?*
20. *What object is directly across from the ottomans?*
21. *Which place to sit is closer to the cabinet?*
22. *What is the only round object placed to the left of the sofa?*
23. *Which table is positioned closer to the sofa?*
24. *Which place to sit is directly opposite the armchairs?*
25. *Which is the nearest place where I can sit and watch the plant?*
26. *Which is the farthest place where I can sit when I enter the room?*
27. *What is the nearest place where I can put down my coffee mug when I enter the room?*
28. *What is the best place where I can sit and read a book at night so that I get sufficient light?*
29. *What is the best place to put my coffee mug down when I am chatting with other guests who are sitting with me?*
30. *Where can I sit so I have to walk least and open the cabinet?*

### Complex queries for office0:

1. *Trash can that is far from the tv*
2. *Table that is towards left and far from the door*
3. *Place that can be used to sleep/rest*



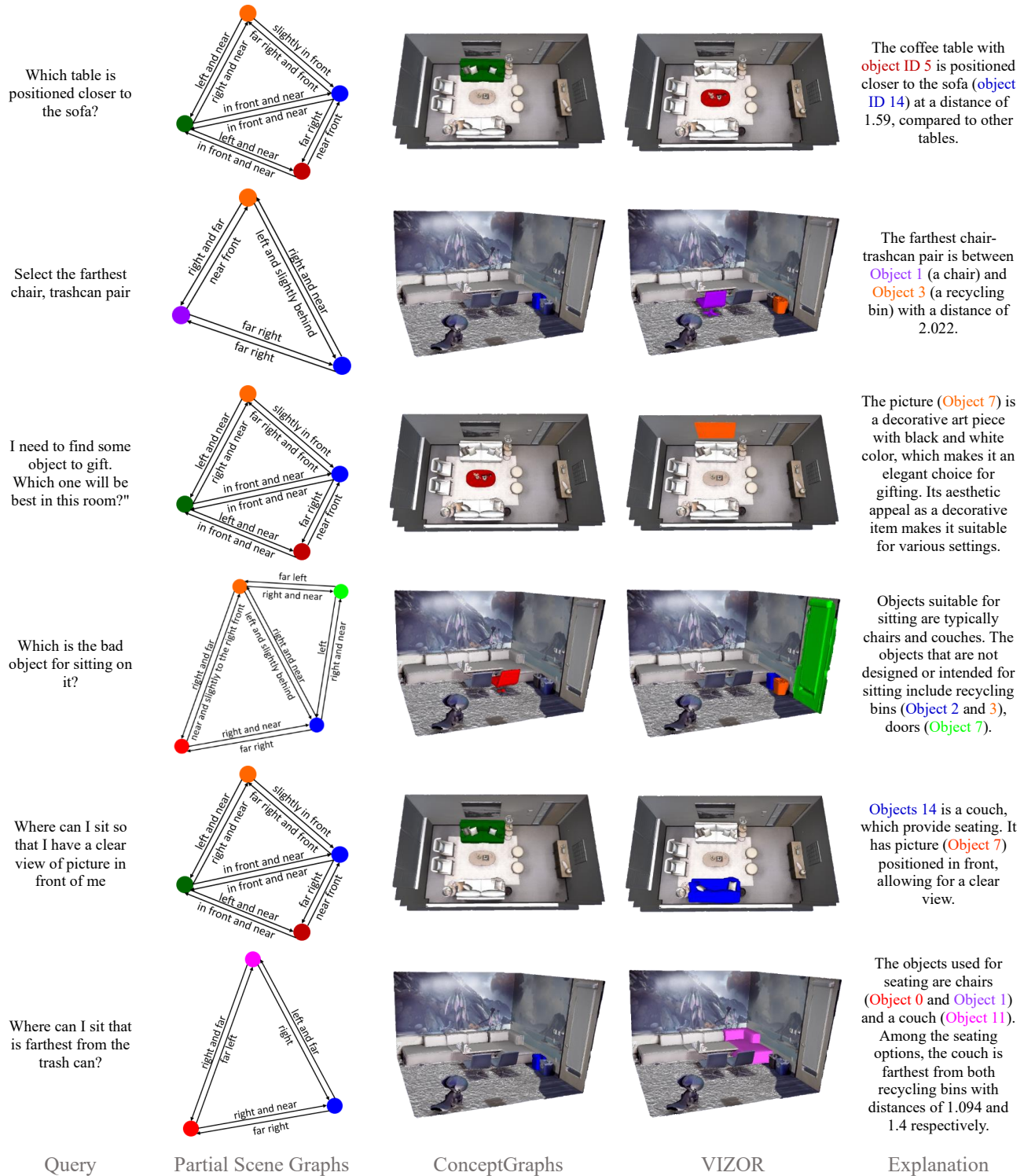


Figure 3. **Qualitative Results:** (a) Input query, (b) Part of scene graph generated by VIZOR, (c) ConceptGraphs [4] output for complex query, (d) VIZOR output, (e) Explanation of object grounding using VIZOR.

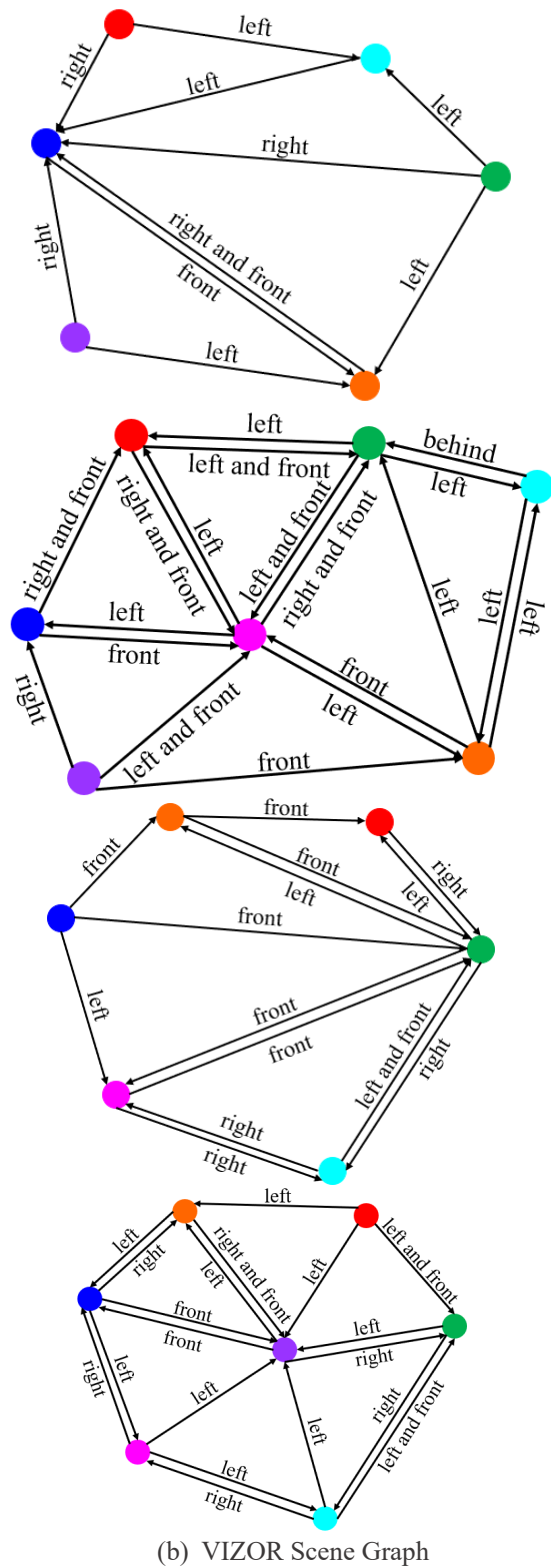
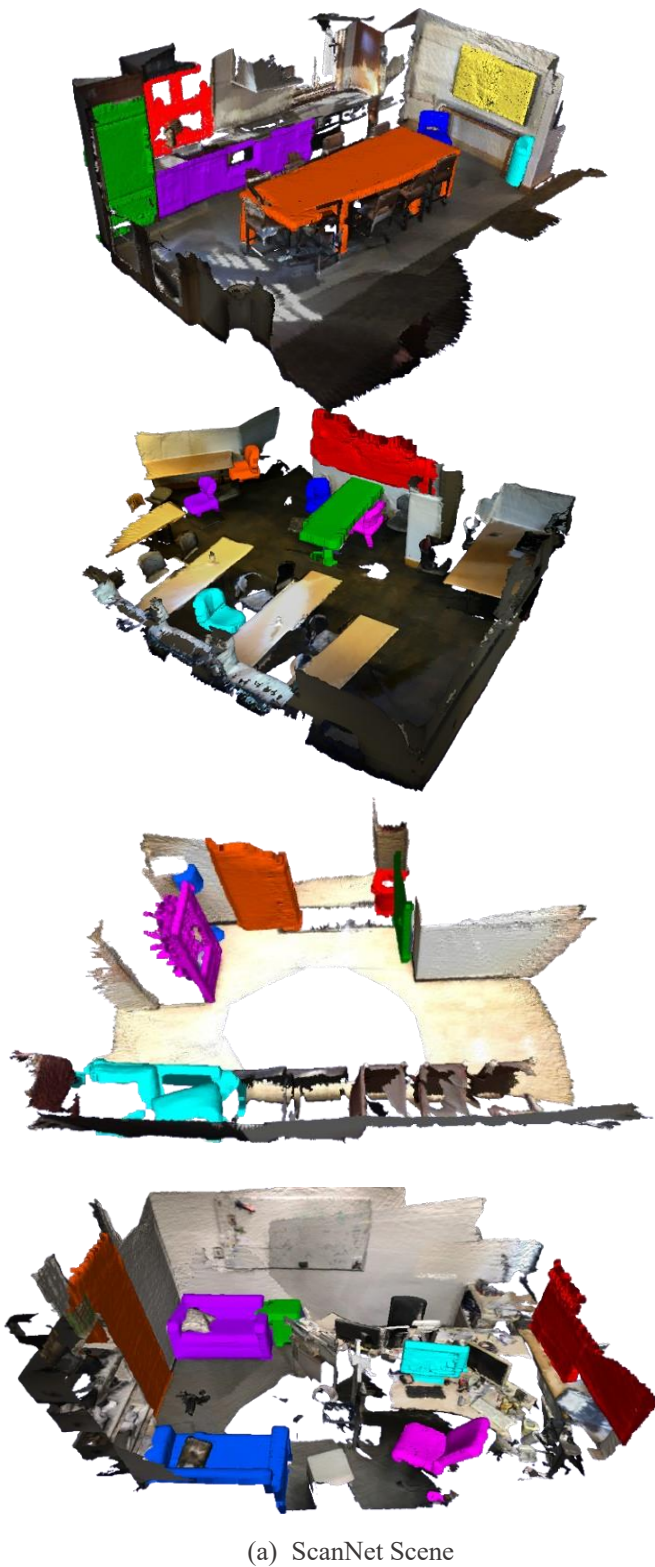


Figure 4. **VIZOR scene graph on ScanNet dataset [2]:** (a) Input scene with color coded nodes, (b) VIZOR scene graph with nodes having same color as in scene

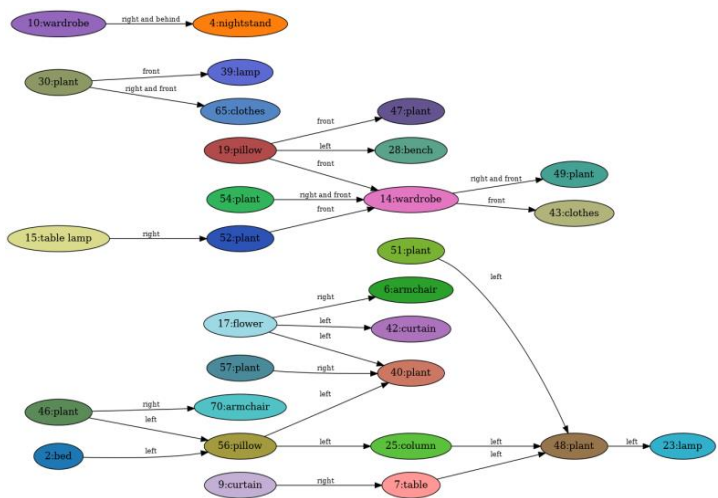
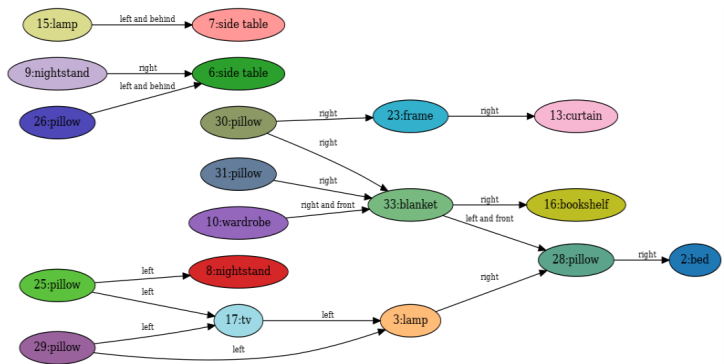
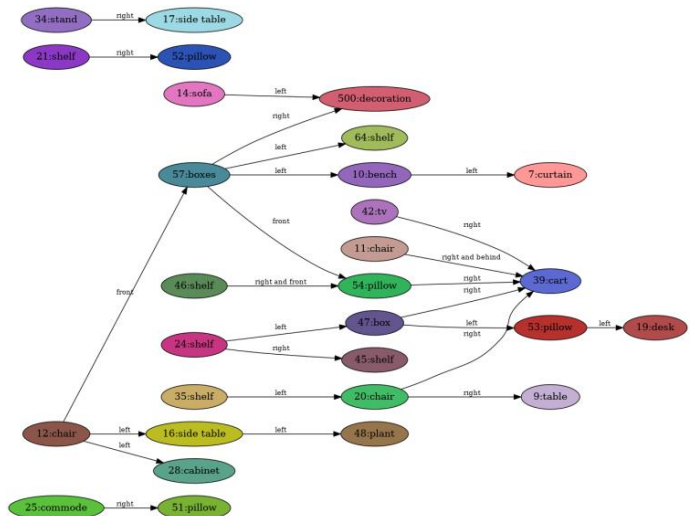


Figure 5. VIZOR scene graph on 3DSSG dataset [2]: (Left) Input scene with color coded nodes, (Right) VIZOR scene graph with nodes having the same color as in the scene.



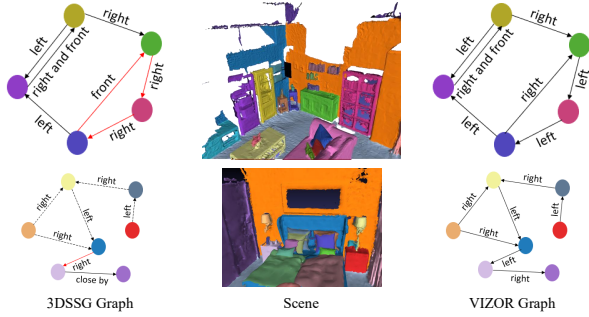


Figure 6. **VIZOR scene graph vs 3DSSG dataset [2]:** (Left) Scene graph from 3DSSG, (Center) Input scene, (Right) VIZOR scene graph with nodes having the same color as in the scene. Dotted lines indicate missing relations and red line indicate wrong relationships

"Read the Images and answer the question:

[Instructions]

[Given 1] You are given the exact front view of an object as the first image. See it and refer to it.

[Given 2] The next 12 images (names are view\_1, view\_2 to view\_12) are the different viewpoints of another object of the same type. Only one of the images is guaranteed to contain the front view of an object. A hint is that usually the front view will have bright colors.

[Task]

1. Which one of them (view\_1 to view\_12) is the best match for exact front view of the object facing the camera? Only answer with the name of the viewpoint image (one of view\_1 to view\_12). Example - view\_1 meaning the first viewpoint image is the front view and view\_10 means the 10th viewpoint image is the front view. You can use your own knowledge to answer but the answer is guaranteed to be one of among view\_1 to view\_12. It should strictly be a front view. Don't consider even if it is slightly front view.
2. Generate a score for each view with probability that it could be front view. Output in the following JSON format

```
{
  "view_1": score_of_view1,
  ....
  "view_12": score_of_view12,
}
```

Figure 7. Prompt for predicting front direction.

""""Provide the description as a JSON object with the following keys:

```
{
  "color": "",
  "functionality": "",
  "geometry": "",
  "affordance": "",
  "caption": ""
}
```

Do not include any other text outside the JSON format. Provide only the valid json following above schema.""

Figure 8. Prompt for generating captions from single view.

4. Table to place snacks while watching tv
5. Trash can that is easily accessible from place where I sit
6. Object that is far from rest other objects
7. Pair of objects lying beside each other
8. What are the suitable places for client and dealer to sit
9. Select the farthest chair, trashcan pair
10. Which is the bad object for sitting on it

```
{
  "color": "dark brown",
  "functionality": "storage for items such as clothes, dishes, or miscellaneous objects",
  "geometry": "rectangular with a tall and narrow structure",
  "affordance": "provides space for organizing and storing various items",
  "caption": "A dark brown cabinet with a tall and narrow design, suitable for storage purposes."
}
```

Figure 9. Output of LLM predicting attributes from single view.

```
"""
### Task:
Combine the above descriptions into a single JSON. Do not include any other text outside the JSON format.

Follow these rules:
1. Resolve conflicts by choosing the most meaningful and consistent value.
2. If a field is missing or ambiguous (e.g., "not specified," "none," or "not known"), infer the value based on other views.
3. Ensure the final JSON has the following fields:
{
  "color": "",
  "functionality": "",
  "geometry": "",
  "affordance": "",
  "caption": ""
}
4. Do not include any extra text.
5. Do not explain anything else. Do not provide any additional JSONs.
"""
```

Figure 10. Prompt for aggregating attributes from multiple views.

```
{
  "color": "dark brown",
  "functionality": "storage for items such as clothes, dishes, books, or miscellaneous objects",
  "geometry": "tall and rectangular with a vertical orientation, featuring multiple compartments or shelves",
  "affordance": "provides storage space, supports placing objects on top, and allows access to stored contents through opening and closing of doors",
  "caption": "A tall, dark brown cabinet suitable for storing various household items."
}
```

Figure 11. Output of LLM aggregating attributes from multiple views.

11. Where should I sit to ignore watching tv
12. List any three objects that are placed in straight line
13. What object is placed in front of the two blue chairs?
14. What is behind the blue chairs?
15. What is to the right of the small trash bins?
16. What is above the wall-mounted screens?
17. what objects are in front of sofa?
18. What is to the right of blue chairs?
19. What object is directly to the left of the blue chairs?
20. What is to the left side of the screens?
21. What is to the 45 degree left side of the screens?
22. Which object in the room can be used for sitting?
23. Where in the room would I sit if I want to have a



[Given]

1. Information about objects where each object has following fields:  
'object\_id': id of the object which is consistent in relationship information,  
'object\_class': class of the object,  
'color': color of the object,  
'functionality': functionality of the object,  
'geometry': geometric information about the object,  
'affordance': what operation it allows,  
'caption': a caption of the object considering it's appearance, geometry, functionality and color,  
'object\_extent': size of the object in x,y,z direction,  
'object\_centroid': object centroid}

2. You are given a relationship data as JSON format where object ids are consistent in object information that you received above. The relationship is about 2 objects (where object ids are consistent in relations and object information). The different fields of a relation are as follows:

```
{
  "target_id": integer - The ID of the object for which relationship is evaluated,
  "target_classname": string - The class name of the object for which relationship is evaluated,
  "anchor_id": integer - The ID of the object with respect to which relationship is evaluated for target object,
  "anchor_classname": string - The classname of the object with respect to which relationship is evaluated for target object,
  "relationship": string - The actual relationship which is evaluated,
  "distance": float - distance between anchor and target object,
  "angle": float - angle between anchor and target object
}
```

[Task]

1. Use the existing relation field as base and generalize it using the object information and the relationship data available.
2. Modify the "relationship" field ONLY and nothing else
2. Use other fields such as distance and angle to generalize the relations (you can add near, far etc as well)
3. Maintain spatial relationships such as left, right, above, below, attached on, on top of etc as well.
3. Strictly do not change any other fields. Keep them as it is.
4. Output only a single JSON and no other extra text. Make sure to strictly maintain the existing JSON structure and only return a valid json string.

"""

Figure 12. Prompt for generalizing geometric rules.

```
{
  "0": {
    "target_id": 1,
    "target_classname": "chair",
    "anchor_id": 0,
    "anchor_classname": "chair",
    "relationship": "left and near",
    "distance": 0.7477035115179348,
    "angle": 0.6351481728168897
  },
  "1": {
    "target_id": 2,
    "target_classname": "recycling bin",
    "anchor_id": 0,
    "anchor_classname": "chair",
    "relationship": "far right",
    "distance": 1.2870074771892308,
    "angle": -1.7625342427833504
  }
}
```

Figure 13. Sample of generalized relations JSON file for single scene.

*conversation with someone sitting on the sofa?*

24. *What object is positioned in front of the wall-mounted*

*screens?*

25. *Which object is placed to the left of the sofa?*

26. *Which object is placed behind the decorative figure when facing the wall-mounted screens?*

27. *Which object is positioned in front of the plant?*

28. *Where can I sit while I can watch my colleague scribbling some ideas in front of me?*

29. *Nearest place to sit when I enter the room?*

30. *Where can I sit that is farthest from the trash can?*

## 8. Precision-Recall Analysis Replica

In this section, we present the precision-recall analysis for all 8 scenes of Replica that were used in the experiments. We use human evaluators for evaluating the precision and recall of the generated relationships using VIZOR-GPT since the replica scenes do not have any ground truths. In Table 2, we present scene-wise precision and recall values, as well as the overall precision and recall for the dataset.

Scene	Relations	Precision	Recall
office1	132	0.85	0.70
office0	110	0.925	0.835
office2	380	0.92	0.64
office3	271	0.865	0.56
office4	380	0.99	0.55
room0	420	0.98	0.66
room1	306	0.94	0.71
room2	210	0.93	0.77
<b>Overall</b>	-	0.925	0.68

Table 2. Precision and recall values across different scenes for Replica

## 9. Robustness of Front Direction Prediction Algorithm

In this section, we present the robustness of our Front Direction prediction algorithm with respect to Reference Database. Reference Database plays a major role in determining the front direction of each object in the scene as each object's front view from Reference DB is provided to LLM as guidance. To create this database, we use Shap-E with "guidance\_scale" parameter set to 15 to generate an object and capture its front view. Quality of generated object depends on this parameter. We experimented by creating different reference databases, where each dataset contains the objects generated with different guidance scale values. We considered values 0.5, 2, 5 and 15. We evaluated accuracy of front direction prediction of few Replica scene objects by our model using each reference database and found that the reference database created using guidance\_scale 15 per-

forms better. Average values are presented in below table, Table 3.

Guidance Scale	0.5	2	5	15
Average	0.58	0.71	0.72	0.745

Table 3. Precision and recall values across different scenes for Replica

We also visualize the front views captured in each reference database for 2 objects in the below figure, Figure 14. From the figure, we can understand that as the guidance scale increases, the quality of generated object increases.

## 10. Failure Cases

As briefly discussed the failure cases in main paper, here we discuss them in detail. Starting with symmetrical objects, usual front direction in ambiguous cases was considered as view closer to the center of the scene. But there can be couple of views that are equidistant from the scene center and still leading to ambiguous cases. Also, certain combinations of objects result in misleading front view prediction. For example, as shown in Figure 15 (a), the coffee table is symmetrical object and it is lying in front of a circular sofa, at one corner of the scene. The circular sofa near the coffee table determines the front direction rather than direction towards scene center. Similarly, in Figure 15(b), the passage through sofa determines the front direction of the object rather than scene center. These kind of examples contribute to failure cases in terms of symmetrical objects.

We also present samples of failure cases where the object is not distinguishable correctly due to color and lighting issues. Because of dark textures and poor lighting, it is difficult to distinguish the parts of an object, resulting in failure while predicting the geometry. For example, consider the couches from Figure 16 (a) highlighted in green bounding boxes. These couches have dark complexion and it is very difficult to determine the parts like backrest, seat etc. Similarly, couch in Figure 16 (b) causes a failure case because of its darker texture. All such cases lead to wrong prediction of front direction, ultimately resulting in wrong relationships.

Here is an other example of one particular instance captured from multiple views, a reference object front view from database to show the failure case of VIZOR in Figure 17. The key observations from this use case is that the improper lighting conditions leading to ambiguous situation in differentiating front view from back view. As a mesh surface (texture) is common inside and out, the color of face is same on both sides (front and back) leading to uncertain decision. Also, missing fine details like depth information when projecting 3D onto 2D leads to failure cases. For example, in Figure 17, “Backrest” can be utilized to predict

the front view but when projecting it onto 2D, benefit of backrest becomes negligible. Note that, here there are two actual front views, one as shown and the other view is at the top right corner. Reason for selecting this particular view is that it closer to the centroid of the scene.

## 11. Limitations and Future Work

**Limitations:** VIZOR generates scene graphs based on the objects’ front direction. The relations in the output graph are highly dependent on this front direction and a small error in predicting this front direction makes all the relations nugatory. Improper orientation or lighting conditions of an object can confuse the LLM in predicting the actual front direction. Detecting the front view of a symmetrical object is challenging and should be refined by taking the context of neighboring objects as shown in Figure 15. VIZOR is dependent on a pre-trained instance segmentation module to identify the potential nodes in the graph. Limitations of this module may miss out on principle nodes in the scene graph.

**Future Works:** In the future, we would like to extend this work by considering background objects to be included in the scene graph and improving the front direction prediction to enhance the accuracy of the current model. Given the wide applicability of scene-graph, we would also like to use the detailed scene-graph creation algorithm as a prior work to perform other downstream tasks like natural language-based manipulations on a given 3D scene, etc.

## 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. 1, 2
- [2] 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*, pages 5828–5839, 2017. 2, 6, 7, 8
- [3] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 1, 2
- [4] 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. 1, 2, 4, 5

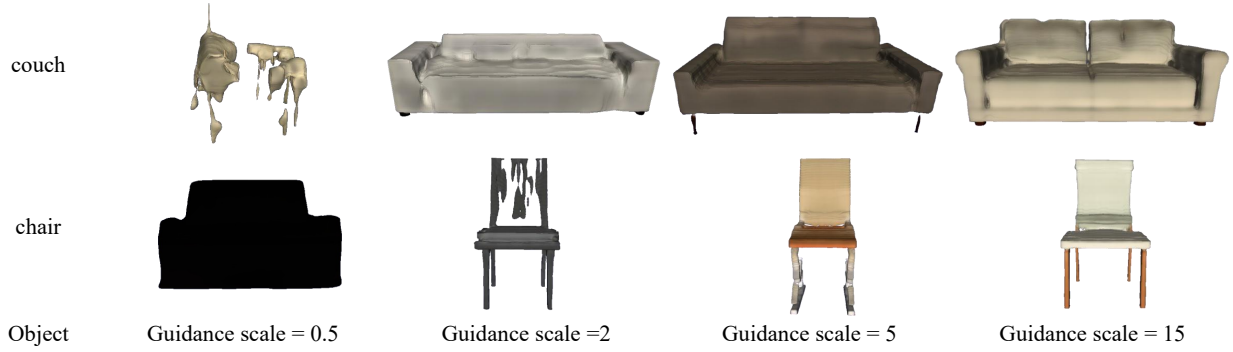


Figure 14. **Front Views:** Front views of 2 objects “couch” and “chair” from different Reference Databases.

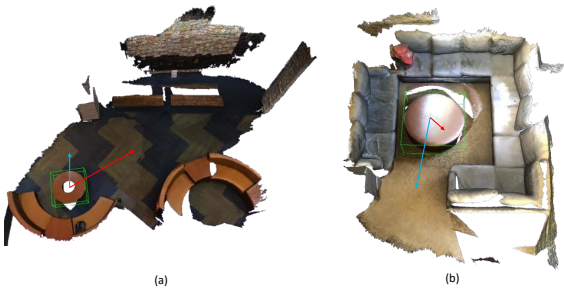


Figure 15. Failure cases of symmetrical objects. Red arrow indicates the predicted front view of highlighted objects (green bounding box) whereas blue arrow indicates semantic front view.

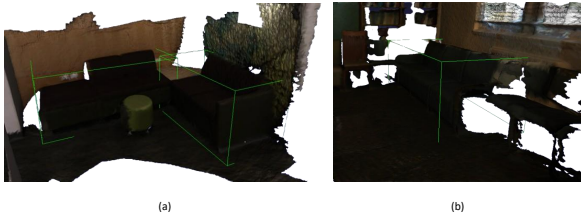


Figure 16. Failure cases of texture and colors. Dark complexion of objects (marked with green bounding boxes) creates difficulty in identifying part level information to predict front view.

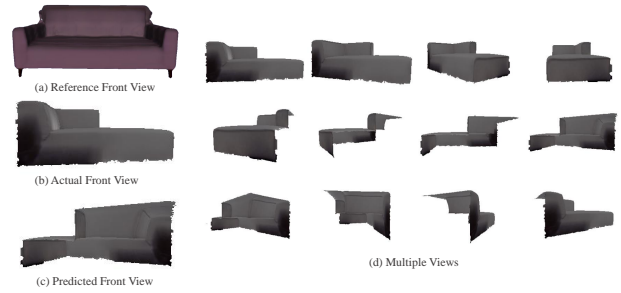


Figure 17. **VIZOR failure case:** (a) Reference view of particular category stored in database, (b) Actual front view of isolated object, (c) Predicted front view by VIZOR and (d) Multiple view capturing the isolated object.

- [5] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 1
- [6] Rong Li, Shijie Li, Lingdong Kong, Xulei Yang, and Junwei Liang. Seeground: See and ground for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 3707–3717, June 2025. 2, 3
- [7] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8216–8223, 2023. 1

- [8] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 2
- [9] Johanna Wald, Helisa Dharmo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2
- [10] Runsen Xu, Zhiwei Huang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Vlm-grounder: A vlm agent for zero-shot 3d visual grounding. In *8th Annual Conference on Robot Learning*, 2024. 2, 3
- [11] Zhihao Yuan, Jinke Ren, Chun-Mei Feng, Hengshuang Zhao, Shuguang Cui, and Zhen Li. Visual programming for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20623–20633, 2024. 2, 3