SeeGround: See and Ground for Zero-Shot Open-Vocabulary 3D Visual Grounding

Supplementary Material

Table of Contents

A. Implementation Details

In this section, we provide additional details to facilitate the implementation and reproducibility of the proposed ***SeeGround*** framework.

A.1. Extended Definitions of Visual Attribute

In the main body of this paper, we introduce several key visual attributes that are essential for 3D Visual Grounding (3DVG) tasks, including attributes such as texture, shape, viewpoint, and order. Tab. A expands upon these attributes, providing more detailed definitions and additional examples to clarify their roles in 3DVG tasks.

This table highlights the indispensable role of visual attributes in disambiguating object references that rely on detailed visual or spatial cues. However, prior approaches [11, 12], particularly those based on large language models (LLMs), often overlook these attributes due to their reliance on textual inputs alone. Without access to visual information, it becomes challenging for such models to interpret queries like "`the black keyboard`" or "`the chair with the tall back`" This limitation underscores the necessity of incorporating visual information into 3DVG tasks to resolve ambiguities and enrich the alignment between textual queries and 3D spatial contexts.

We hope these analyses could provide insights for future exploration of multimodal systems that integrate textual and visual information in 3DVG.

A.2. Details of Textual Prompt Design

In this work, we design useful prompts to facilitate the learning of visual attributes for 3DVG. As illustrated in Tab. B, these prompts include several key components which are summarized as follows:

- **Role Specification:** The prompt begins by defining the assistant's role as an entity designed to identify objects based on images and descriptions. This specification is crucial for setting the context and ensuring that the assistant's actions align with the intended task.
- **Visual Contextualization:** The prompt provides a description of the image, indicating that it is a rendered image of a room. This contextualization helps the assistant to understand the spatial layout and the perspective from which the objects should be identified, which is essential for accurate object recognition.
- **Object Labeling & Spatial Information:** Each object within the image is labeled with a unique identifier (ID) in red, accompanied by detailed spatial information. This includes object type, dimensions, and center coordinates. Such detailed labeling is vital for distinguishing between objects, especially in complex scenes where multiple objects may have similar appearances.
- **Response Protocol:** The prompt specifies a structured format for the assistant's response, requiring a detailed explanation of the features or context that led to the decision. The response format, "`Predicted ID: <ID>Explanation: <explanation>`", ensures that the assistant's reasoning is transparent and verifiable. This protocol is exemplified by the description, "`This is the large conference table with many chairs`", which serves as a practical application of the identification process.

These components are meticulously designed to guide the assistant in leveraging both visual and descriptive information for object identification. The structured format not only ensures clarity and consistency in the assistant's responses but also facilitates effective communication and decision-making. By providing a comprehensive framework, the prompts enable the assistant to perform complex identification tasks with precision and reliability, which is critical in applications requiring high accuracy and interpret-ability.

Table A. Detailed explanations of **Key Visual Attributes** in 3D Visual Grounding (3DVG) tasks. In 3DVG, the model must understand the relationships between visual attributes and spatial descriptions in the query to correctly identify and locate the target object. These attributes – color/texture, shape, viewpoint, order, orientation, state, and functionality – serve as crucial cues that guide the model.

Attribute	**Definition**	**Examples**
(a) Texture	Refers to the visual appearance of an object's surface, including its color and material properties. These attributes help distinguish objects with similar shapes but different visual properties.	• "The `black` keyboard." • "The cart you're looking for is `white` on top." • "The `floral` chair." • "The correct door has `vertical lines` on it." • "Choose the `glass doors`." • "A brown box with a `white label` on the front of the box."
(b) Shape	Describes the geometric form of an object, which allows for differentiation between objects with similar class names and sizes but different geometric structures.	• "The `round` trash can." • " `double door`." • " `L-shaped` couch." • "The chair with the `tall back`." • "Choose the `three seater` couch." • "The correct couch has a `90-degree angle` bend in it it is not straight."
(c) Viewpoint	Refers to the perspective or angle from which an object or scene is observed. Viewpoint impacts the visibility and relative positioning of objects.	• "When `facing` the windows, the one on the left." • "When `entering the room`, the sofa is on the right side." • " `Facing` the TV, the chair on the right." • "When `sitting` at the bed, the lamp is on the left corner."
(d) Orientation	Describes the rotational alignment of an object in 3D space, which is essential for distinguishing objects that may appear but are oriented differently.	• "The table is `tilted` slightly." • "the keyboard `at an angle`." • "The chair whose `back is facing` the window." • "The picture frame is `leaning` at an angle on the shelf."
(e) State	Refers to the current condition or status of an object (*e.g.*, open/closed, active/inactive), helping to differentiate objects that may appear similar but have different functional states.	• "The `open` door." • "The `closed` book on the desk." • "The `lit` candle on the shelf." • "The `empty` cup on the counter." • "The `stained` carpet in the living room."
(f) Order	Refers to the relative positioning or sequence of objects within a scene. This is important for identifying objects in a specific spatial arrangement.	• "Of the group of pictures, choose `the second` one from the right." • "In the row of chairs, pick the `the fourth` one from the left." • "From the stack of books, take `the third` one from the top."
(g) Functionality	Describes the intended role or purpose of an object in the scene. Functionality helps distinguish between objects with similar appearances but different purposes.	• "It is the door to `get into the bathroom`." • "The door `for people`."

Table B. An example of the instruction used for prompting the VLMs to identify the target object with a rendered image.

You are a helpful assistant designed to identify objects based on images and descriptions.
As shown in the image, this is a rendered image of a room, and the picture reflects your current view.
You should distinguish the target object ID based on your current view. Each object is labeled by a unique number (ID) in red color on its surface.

Object IDs and their spatial information are as follows:
`Object ID`: 1, `Type`: cabinet, `Dimensions`: Width 0.19, Length 3.21, Height 0.84, `Center Coordinates`: X 1.42, Y 1.01, Z 0.41,
...

The 3D spatial coordinate system is defined as follows:
The X-axis and Y-axis represent horizontal dimensions, with the Y-axis perpendicular to the X-axis. The Z-axis represents the vertical dimension, with positive values pointing upwards.

Please review the provided image and object descriptions, then select the object ID that best matches the given description. Provide a detailed explanation of the features or context that led to your decision.

Respond in the format: "`Predicted ID: <ID> Explanation: <explanation>`", where `<ID>` is the object ID and `<explanation>` is your reasoning.

The given description is: "`This is the large conference table with many chairs`".

Table C. An example of the instruction used for prompting the VLMs to identify the target and anchor objects based on the query.

You are an assistant designed to identify relationships between objects in a scene.
Your task is to determine both the target and anchor objects based on the query description provided.

Here are some examples:
- "`Find the chair that is next to the wooden table.`"
  `Target: chair, Anchor: wooden table`
- "`Identify the lamp that is on top of the desk.`"
  `Target: lamp, Anchor: desk`
- "`Locate the book that is under the coffee table.`"
  `Target: book, Anchor: coffee table`

...

Now, based on the query below, provide the names of the target object and the anchor object.
Response in the format: "`Target: <target object>, Anchor: <anchor object>`".

Query: "`Find the bowl that is on the dining table.`".

In the main paper, we also discuss the process of determining anchor and target objects based on the query description. To further clarify this process, we provide an illustrative example of the prompt in Tab. C. This prompt guides the VLM to identify both the target object and its associated anchor object by analyzing their spatial and semantic relationships as described in the query. This design ensures the model focuses on identifying key objects while maintaining alignment with the query's context.

A.3. Look-At-View Transform

In the Perspective Adaptation Module, we utilize the $\mathrm{look\_at\_view\_transform}$ function to compute the extrinsic parameters of the virtual camera.

Specifically, the camera's rotation matrix $\mathbf{R}_c$ and translation vector $\mathbf{T}_c$ are determined based on the camera's position $\mathbf{e} = (x_c, y_c, z_c)$, the anchor point $\mathbf{at} = (x_a, y_a, z_a)$, and the up vector $\mathbf{up}$. Below, we provide a formal description of the computation process.

- **Camera Rotation Matrix $\mathbf{R}_c$.** The camera rotation matrix $\mathbf{R}_c \in \mathbb{R}^{3\times3}$ aligns the camera's local coordinate system with the world coordinate system. It is derived from the following steps:
  - The forward direction ($\mathbf{z}_{\text{axis}}$) is the normalized vector from the camera to the anchor:
    $$\mathbf{z}_{\text{axis}} = \frac{\mathbf{at} - \mathbf{e}}{\|\mathbf{at} - \mathbf{e}\|}, \tag{1}$$
    where $\mathbf{at} - \mathbf{e} = (x_a - x_c, y_a - y_c, z_a - z_c)$.
  - The right direction ($\mathbf{x}_{\text{axis}}$) is obtained as the normalized cross product of the up vector $\mathbf{up}$ and $\mathbf{z}_{\text{axis}}$:
    $$\mathbf{x}_{\text{axis}} = \frac{\mathbf{up} \times \mathbf{z}_{\text{axis}}}{\|\mathbf{up} \times \mathbf{z}_{\text{axis}}\|}. \tag{2}$$
  - The up direction ($\mathbf{y}_{\text{axis}}$) is calculated as the cross product of $\mathbf{z}_{\text{axis}}$ and $\mathbf{x}_{\text{axis}}$:
    $$\mathbf{y}_{\text{axis}} = \mathbf{z}_{\text{axis}} \times \mathbf{x}_{\text{axis}}. \tag{3}$$
  - Finally, $\mathbf{R}_c$ is constructed by stacking these three vectors:
    $$\mathbf{R}_c = \begin{bmatrix}\mathbf{x}_{\text{axis}} & \mathbf{y}_{\text{axis}} & \mathbf{z}_{\text{axis}}\end{bmatrix}^\top. \tag{4}$$
- **Camera Translation Vector $\mathbf{T}_c$.** The translation vector $\mathbf{T}_c \in \mathbb{R}^3$ corresponds to the position of the camera in the world coordinate system:
  $$\mathbf{T}_c = \mathbf{e} = (x_c, y_c, z_c). \tag{5}$$
  The $\mathrm{look\_at\_view\_transform}$ function provides a systematic way to compute the extrinsic parameters of a camera in 3D space. The rotation matrix $\mathbf{R}_c$ transforms world coordinates into the camera's view, while the translation vector $\mathbf{T}_c$ represents the camera's position. These parameters are essential for rendering and aligning 3D scenes to match the desired perspective.
- **Query-Aligned Image Rendering.** Once $\mathbf{R}_c$ and $\mathbf{T}_c$ are computed, the 3D scene $\mathcal{S}$ is projected into a 2D image plane to render the query-aligned image $\mathbf{I}$:
  $$\mathbf{I} = \mathrm{Render}(\mathcal{S}, \mathbf{R}_c, \mathbf{T}_c). \tag{6}$$
  This ensures that the rendered image captures the spatial relationships and visual context described in the query.

A.4. Details of Depth-Aware Visual Prompting

In our method, depth-aware visual prompting plays a key role in aligning 3D spatial information with 2D visual representations while addressing challenges like occlusion during projection. This section elaborates on the technical details and additional considerations involved in this process, which extends the explanation in the main text.

- **Generating Visual Prompts.** To create visual prompts, we first retrieve the 3D bounding boxes and associated point sets for candidate objects from the OLT. These 3D points are then projected onto the 2D image plane using the camera parameters $\mathbf{R}_c$ (rotation matrix) and $\mathbf{T}_c$ (translation vector) obtained during the rendering process. Specifically, for a given 3D point $p = (x, y, z)$, the 2D projection $p' = (x', y')$ is computed as:
  $$p' = \mathbf{R}_c p + \mathbf{T}_c. \tag{7}$$
  Once projected, visual markers are initially placed at the center of the projected points for each object. This provides a basic alignment of the object's location within the rendered image.
- **Addressing Occlusion Using Depth Information.** However, projecting 3D points onto a 2D plane often introduces occlusions, where some parts of an object may overlap with other objects or the background. Directly placing visual prompts without accounting for occlusions can lead to ambiguity and misalignment between the visual markers and the object they represent. To address this, depth information is utilized to determine the visibility of each point. For every pixel p' on the 2D image, the scene's depth map $\mathcal{D}(p')$ stores the smallest depth value among all points projected to that pixel. Formally, the depth map is defined as:
  $$\mathcal{D}(p') = \min_{p_s \in \mathcal{S}} d_s, \tag{8}$$
  where $\mathcal{S}$ is the set of all 3D points in the scene, and d_s is the depth of point p_s relative to the camera. To check the visibility of a 3D point p, its depth d_p is compared with $\mathcal{D}(p')$ at its projected location p'. A point is considered visible if:
  $$\text{Visible}(p) = \begin{cases} 1, & \text{if } d_p < \mathcal{D}(p')\ , \\ 0, & \text{otherwise}\ . \end{cases} \tag{9}$$
- **Object-Level Visibility and Prompt Placement.** To determine whether an object should be visually prompted, the visibility of its constituent points is aggregated. An object o is considered visible if a sufficient fraction of its points passes the visibility check:
  $$\text{Visible}(o) = \begin{cases} 1, & \text{if } \sum_{p_o \in \mathcal{P}_o} \text{Visible}(p_o) \geq \alpha \cdot |\mathcal{P}_o|, \\ 0, & \text{otherwise}, \end{cases} \tag{10}$$

Table D. Performance comparison of different VLMs on Nr3D [1].

Agents	Easy	Hard	Dep.	Indep.	Overall
InternVL2-8B	43.6	25.8	32.6	35.4	34.3
InternVL2-26B	46.8	29.8	34.7	39.8	38.0
Qwen2-VL-7B	40.8	26.3	31.4	34.3	33.3
Qwen2-VL-72B	**54.5**	**38.3**	**42.3**	**48.2**	**46.1**

where $\mathcal{P}_o$ is the set of 3D points corresponding to object o, $|\mathcal{P}_o|$ is the total number of points for the object, and α is a threshold factor determining the minimum fraction of visible points required.

By filtering out occluded points and using only visible points to place visual markers, the depth-aware prompting process ensures accurate alignment of 2D visual prompts with the true 3D spatial context of objects. This minimizes errors caused by overlapping objects and improves the model's understanding of scene geometry.

B. Additional Quantitative Results

In this section, to pursue a more comprehensive comparison, we provide additional quantitative results of the proposed ***SeeGround*** framework.

B.1. Agents of Different Sizes

Tab. D showcases the performance of different open-source VLMs on the Nr3D [1] validation set, evaluated across various difficulty levels and dependency types. The results highlight the compatibility of our pipeline with multiple VLM architectures, including InternVL [3, 4, 10] and Qwen2-VL [8], across different model sizes.

Notably, the proposed pipeline is not restricted to the specific VLMs shown in the table. It is inherently designed to be adaptable to any VLM with Optical Character Recognition (OCR) capabilities. Within our framework, OCR functionality plays a crucial role in identifying object IDs in rendered images and associating them with textual descriptions. This process enables precise alignment between 2D visual features and 3D spatial information. Consequently, the pipeline is well-suited for integration with a wide range of existing and future VLMs, further extending its applicability to 3D visual grounding tasks.

B.2. Analysis of Visual Prompt Types

To further explore the role of visual prompts in 3D visual grounding, we provide an analysis of alternative designs, including Mask, Contour, and BBOX, as illustrated in Fig. A and Tab. E. These visual prompts each present unique advantages and limitations, particularly when combined with 3D spatial information, as used in our method. We conducted experiments using a subset (randomly selected 40 scenes from the 130 scenes) of the Nr3D validation set.

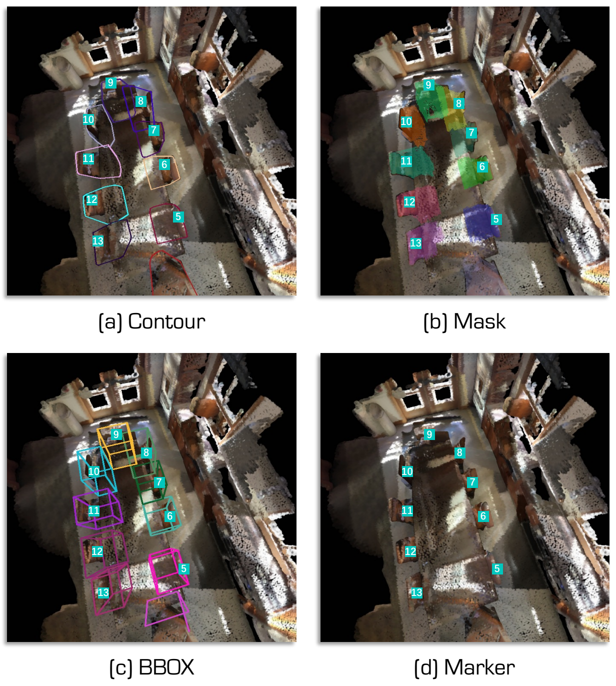


Figure A. Illustrative examples of different visual prompts in our designs. The Marker size is enlarged for clarity.

Table E. Performance comparison of different visual prompts: Marker, Mask, Contour, and BBOX. Results are from 40 randomly selected scenes out of 130 rooms in the Nr3D [1] validation set.

Type	Easy	Hard	Dep.	Indep.	Overall
BBOX	53.3	37.4	41.1	47.3	45.1
Mask	53.9	35.1	39.6	47.4	45.0
Contour	56.2	37.7	43.1	49.4	47.5
Marker	**54.8**	**39.7**	**40.2**	**51.0**	**47.7**

- **Mask.** It intuitively highlights the entire object surface, making the target region explicitly visible. However, even with high transparency (as shown in Fig. A (b)), Mask can obscure surface details like texture and fine-grained patterns, which are critical for distinguishing objects. Additionally, the extra appearance information provided by Mask may be unnecessary when 3D spatial information is already available, potentially distracting the model's attention. Moreover, generating and rendering masks for all surface points increases computational overhead, especially in complex scenes.
- **BBOX.** It clearly defines spatial boundaries but introduces visual complexity due to the overlay of bounding box lines. These lines often obscure surface features (color/texture), interfering with the model's ability to interpret appearance details. In dense or overlapping object scenarios, bounding boxes can create additional confu-

Table F. Performance comparison of different 3D detectors on the ScanRefer [2] validation set. Accuracy (Acc.) is reported for each method paired with different 3D detectors.

Method	3D Detector	Acc.
ZSVG3D [12]	Mask3D [7]	36.4
	OVIR-3D	19.3
SeeGround	Ground Truth	59.5
	Mask3D [6]	44.1
	OVIR-3D [5]	30.7

sion. Furthermore, the spatial information conveyed by BBOX prompts is redundant when 3D spatial positions are already provided, diminishing model performance.

- **Contour.** It represents a balance between simplicity and informativeness. By outlining object boundaries, they provide clear spatial context while avoiding the occlusion issues of Mask and BBOX. Contours also retain surface visibility, preserving critical appearance cues. The experimental results indicate that Contours perform similarly to Markers because both approaches minimize visual distractions while preserving spatial and appearance cues.
- **Marker.** It offers the most minimal and focused design, marking object centers without introducing visual clutter or occluding appearance features. This approach maximally preserves object details like texture and color while providing essential spatial information. The direct mapping of markers to 3D spatial positions aligns seamlessly with the 3D spatial information already used in our method, enhancing localization precision.

While Mask, Contour, and BBOX prompts each have specific strengths, their limitations – such as visual interference or redundancy – make Marker the most suitable choice for our framework. Its simplicity and compatibility with 3D spatial inputs ensure efficient and accurate 3D visual grounding in complex scenarios.

B.3. Results on Different Detectors.

Tab. F presents a performance comparison of different 3D detectors on the ScanRefer validation set, highlighting the impact of detector choice on grounding accuracy. With the same 3D detector (Mask3D), our method significantly outperforms the previous state-of-the-art approach, ZSVG3D, achieving an accuracy of 44.1 compared to 36.4. We also explore OVIR-3D as an alternative detector. The results show that our method achieves an accuracy of 30.7 with OVIR-3D, while ZSVG3D achieves 19.3 under the same setting. Additionally, the table reveals the upper-bound performance of our method when using ground truth (GT) proposals, reaching an accuracy of 59.5. This underscores the importance of improving 3D object detection accuracy, as better detection directly translates to enhanced grounding results.

B.4. Real-world Image *vs*. Rendered Image

SeeGround begins with 3D object detection, which is performed directly on the 3D point cloud. Point clouds, while sparse and noisy, inherently capture geometric details like size, shape, and spatial location. This makes the 3D detection stage less susceptible to the visual artifacts that typically affect rendered images (e.g., inconsistent lighting, color shifts). Following this, the detected objects are used to generate rendered images from selected viewpoints. These rendered images serve as visual inputs for VLMs, combined with explicit textual descriptions and spatial information. This workflow naturally raises questions about the impact of rendering quality on the method's performance, particularly in comparison to methods discussed in works like EmbodiedScan [9] (Table 7), which highlight a domain gap between rendered images and real-world images.

However, unlike methods that rely purely on rendered images for learning and inference (e.g., rendering RGB images and directly training models on them), SeeGround treats rendered images as part of a multimodal input. The rendered images provide a visual representation of the scene but are supplemented by 3D spatial data, which is independent of rendering quality. This additional spatial information reduces reliance on rendering fidelity.

C. More Visualization Results

Fig. B provides additional visual examples to supplement the analysis in the main text, further illustrating the advantages of our method over previous approaches. By comparing predictions made by previous methods and Ours across various query-based 3D visual grounding tasks, we highlight the importance of appearance information (e.g., texture, color, and shape) in resolving ambiguities and improving localization accuracy.

As shown in these examples, previous methods often fail to utilize appearance information effectively, leading to incorrect predictions when objects share similar spatial configurations or belong to the same category. For instance, queries like "`the trash can next to the blackboard`" or "`the monitor in front of the black keyboard`" require fine-grained differentiation based on appearance attributes. Previous methods tend to misidentify nearby or visually similar objects due to their limited ability to integrate these attributes into the grounding process. In contrast, our method incorporates appearance information explicitly through depth-aware visual prompting, enabling more accurate alignment of textual descriptions with 3D spatial and visual cues.

These supplementary results emphasize the critical role of appearance information in 3D visual grounding and demonstrate how our method effectively leverages this information to address ambiguities. By incorporating visual features alongside spatial reasoning, our approach achieves significant improvements in challenging scenarios, further validating the findings presented in the main text.

D. Broader Impact & Limitations

In this section, we elaborate on the broader impact and potential limitations of this work.

D.1. Broader Impact

Our approach bridges 3D data and 2D VLMs, making 3D visual grounding accessible in zero-shot settings. This design reduces reliance on large-scale 3D-specific datasets and annotations, enabling scalable deployment. By focusing on integrating 2D rendered images with spatial descriptions, our method highlights the importance of appearance features like color, texture, and orientation, which are often overlooked in previous zero-shot approaches. Applications range from assistive technologies to robotics and augmented reality, where robust object localization can enhance usability and accessibility. Moreover, the use of visual prompts, especially the Marker-based design, introduces an interpretable mechanism for aligning visual and spatial information. This improves transparency and trust in AI systems, allowing stakeholders to better understand the reasoning behind model predictions.

D.2. Potential Limitations

Despite its advancements, our method has some limitations. It relies on accurate 3D object detection and spatial data, making it vulnerable to errors in preprocessing. Misaligned bounding boxes or missing objects can propagate through the pipeline, reducing localization accuracy. Marker-based visual prompts, while simple and effective, may struggle in cluttered scenes requiring richer contextual information and can obscure very small objects, complicating precise localization. The method leverages 2D-3D alignment without requiring highly accurate rendered images, but consistent alignment remains crucial for effective multimodal fusion. Significant deviations in rendered images – caused by inaccurate camera parameters or low-quality point clouds – can compromise alignment between 2D prompts and 3D spatial descriptions. This issue is exacerbated in cluttered/dynamic scenes, where rendering delays can lead to mismatches between 2D prompts and real-time 3D data, causing errors in grounding. For instance, in scenes with moving objects, outdated rendered views may misrepresent object positions, leading to incorrect target identification. Future work could enhance multimodal alignment robustness under noisy or sparse data, improve real-time efficiency, and better handle dynamic and cluttered environments, broadening the method's applicability to complex real-world scenarios.

E. Public Resource Used

In this section, we acknowledge the use of the following public resources, during the course of this work:

• Pytorch [1]	Pytorch License
• Pytorch3D [2]	BSD-Style License
• Open3D [3]	MIT license
• ScanRefer[4]	ScanRefer License
• Nr3D [5]	MIT License
• Qwen2-VL[6]	Apache License 2.0
• InternVL2 [7]	MIT license
• ZSVG3D [8]	Unknown
• vLLM [9]	Apache License 2.0
• OpenScene [10]	Apache License 2.0
• vil3dref [11]	Unknown
• OpenIns3D [12]	MIT License
• EmboddiedScan [13]	Apache License 2.0
• LAR [14]	MIT License

References

[1] Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *European Conference on Computer Vision*, pages 422–440. Springer, 2020. 5

[2] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *European conference on computer vision*, pages 202–221. Springer, 2020. 6

[3] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. *arXiv preprint arXiv:2312.14238*, 2023. 5

[4] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*, 2024. 5

[5] Shiyang Lu, Haonan Chang, Eric Pu Jing, Abdeslam Boularias, and Kostas Bekris. Ovir-3d: Open-vocabulary 3d in-

[1] https://github.com/pytorch/pytorch
[2] https://github.com/facebookresearch/pytorch3d
[3] https://github.com/isl-org/Open3D
[4] https://daveredrum.github.io/ScanRefer/
[5] https://github.com/referit3d/referit3d
[6] https://github.com/QwenLM/Qwen2-VL
[7] https://huggingface.co/OpenGVLab/InternVL2-26B
[8] https://github.com/CurryYuan/ZSVG3D
[9] https://github.com/vllm-project/vllm
[10] https://github.com/pengsongyou/openscene
[11] https://github.com/cshizhe/vil3dref
[12] https://github.com/Pointcept/OpenIns3D
[13] https://github.com/OpenRobotLab/EmbodiedScan
[14] https://github.com/eslambakr/LAR-Look-Around-and-Refer

stance retrieval without training on 3d data. In *Conference on Robot Learning*, pages 1610–1620. PMLR, 2023. 6

[6] 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*, pages 8216–8223. IEEE, 2023. 6

[7] Ayça Takmaz, Elisabetta Fedele, Robert W. Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. Openmask3d: Open-vocabulary 3d instance segmentation. *arXiv preprint arXiv:2306.13631*, 2023. 6

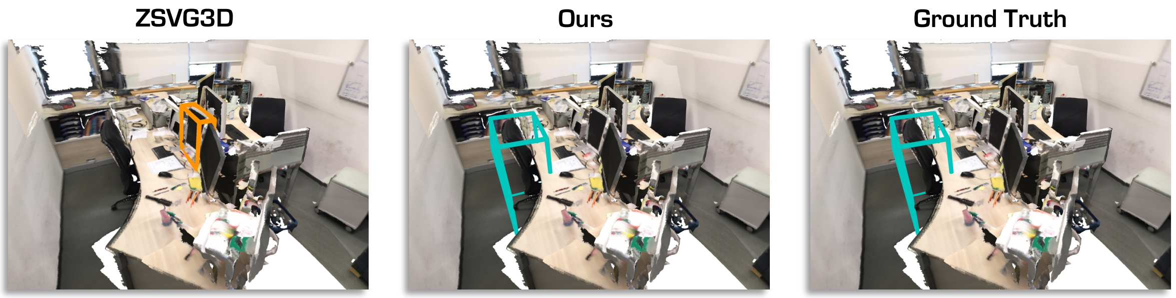
[8] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 5

[9] Tai Wang, Xiaohan Mao, Chenming Zhu, Runsen Xu, Ruiyuan Lyu, Peisen Li, Xiao Chen, Wenwei Zhang, Kai Chen, Tianfan Xue, et al. Embodiedscan: A holistic multimodal 3d perception suite towards embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19757–19767, 2024. 6

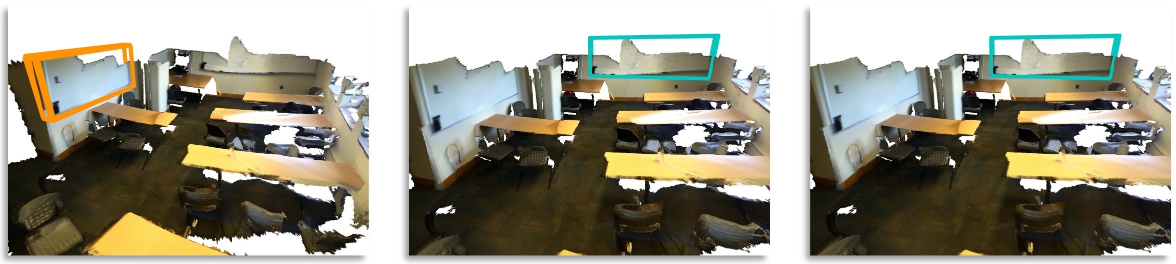
[10] Weiyun Wang, Zhe Chen, Wenhai Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Jinguo Zhu, Xizhou Zhu, Lewei Lu, Yu Qiao, and Jifeng Dai. Enhancing the reasoning ability of multimodal large language models via mixed preference optimization. *arXiv preprint arXiv:2411.10442*, 2024. 5

[11] Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F Fouhey, and Joyce Chai. Llmgrounder: Open-vocabulary 3d visual grounding with large language model as an agent. In *IEEE International Conference on Robotics and Automation*, pages 7694–7701. IEEE, 2024. 1

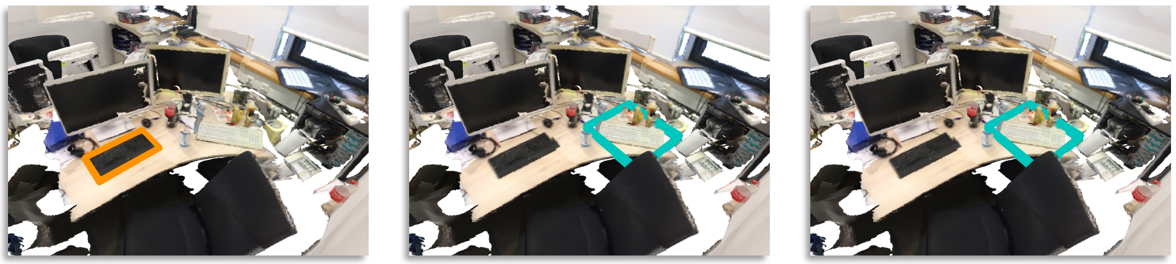
[12] Zhihao Yuan, Jinke Ren, Chun-Mei Feng, Hengshuang Zhao, Shuguang Cui, and Zhen Li. Visual programming for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20623–20633, 2024. 1, 6



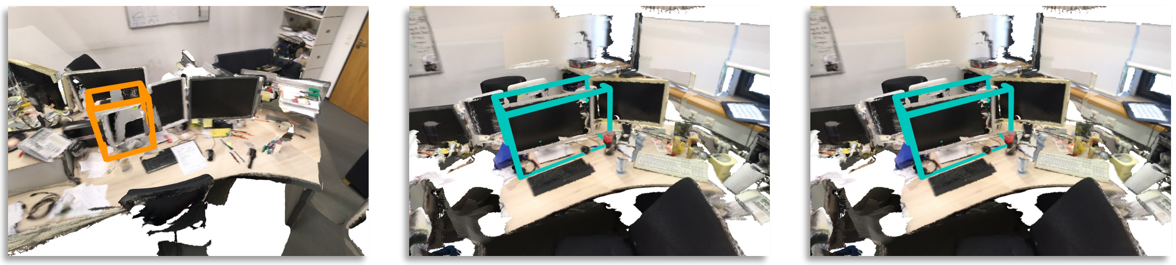
(a) **Black office chair** that is pushed all the way up to the desk. It is facing a large computer screen and there is a white notepad and keyboard on the desk in front of it.



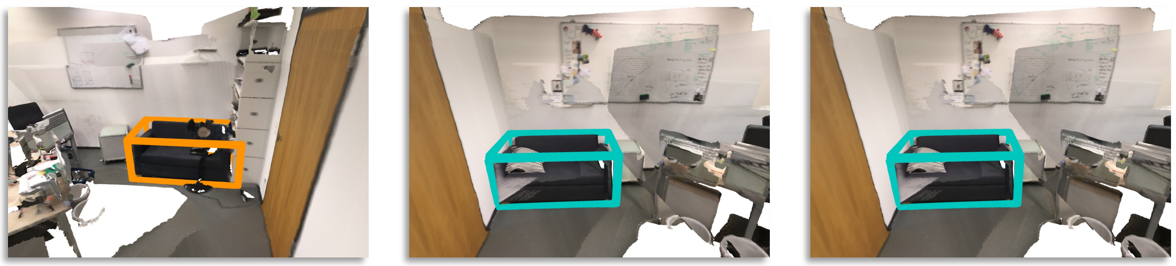
(b) The longer **whiteboard** on the wall furthest away from the doorway.



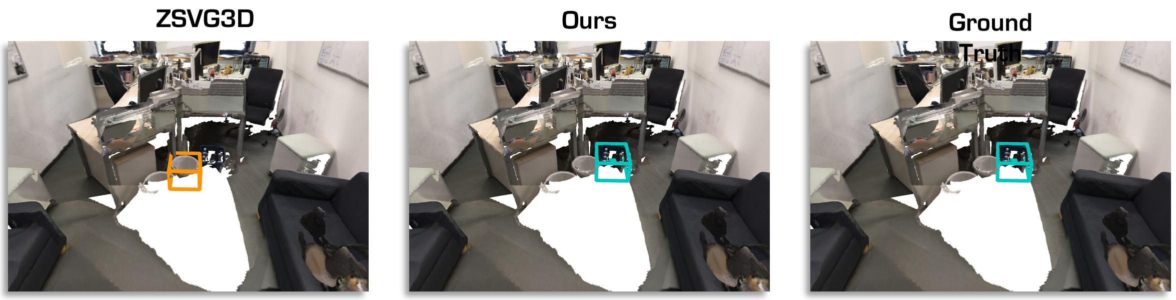
(c) White **keyboard** on the desk.



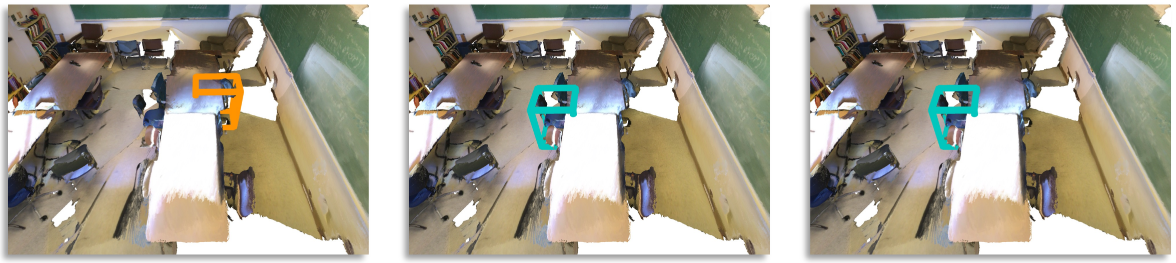
(d) On the desk with just two monitors, it's the **monitor** with the black keyboard in front of it.



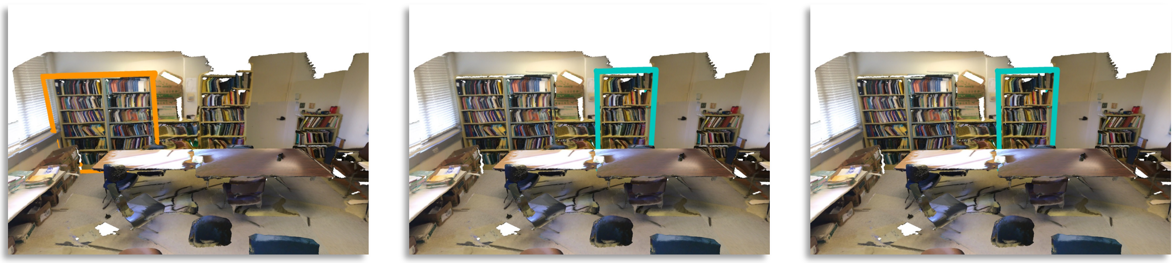
(e) The **couch** under the whiteboard.



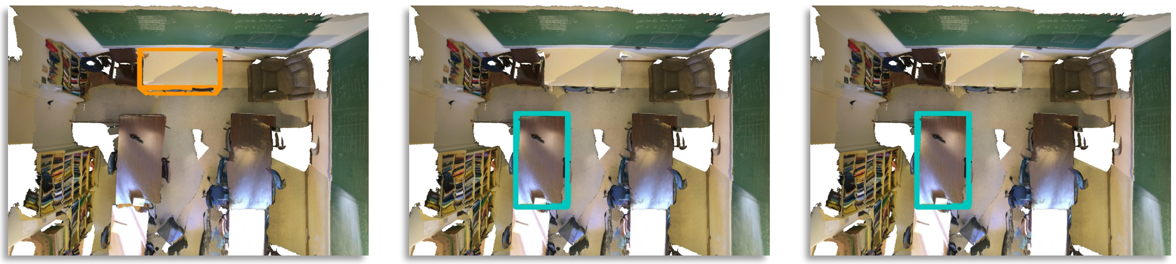
(f) When standing in the doorway, it is the **trash can** on the right side, nearest to the white board.



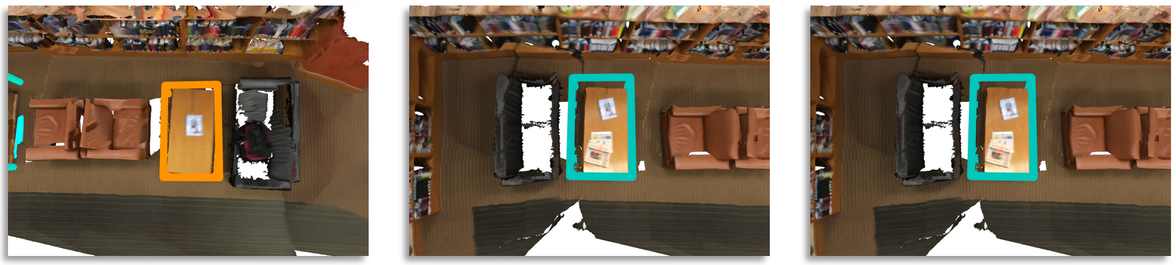
(g) The blue **office chair** in the middle of the room, behind another chair.



(h) First **bookshelf** to the left of the door.



(i) A black object sits on this **table**.



(j) This **couch** is by the table with 3 magazines on top.

Figure B. Illustration of SeeGround's capability to resolve ambiguities in 3D visual grounding task. Incorrectly identified objects (Orange) and correctly identified objects (Green) are indicated to differentiate prediction accuracy, key cues are underlined.