

A Details of Struct2D Prompting Strategy.

Figure 7 illustrates the overall Struct2D prompting framework, which transforms egocentric 3D scene input into structured 2D representations for spatial reasoning. Given an input video and a spatial question, we first reconstruct a 3D point cloud from RGB-D frames and remove the ceiling to obtain a clear top-down view of the scene. Object detection is then performed in 3D space, and detected objects are projected onto a bird’s-eye-view (BEV) image to produce a layout of the environment. These object marks are filtered to retain only those relevant to the input question.

We optionally extract egocentric keyframes to capture detailed object appearances. Keyframes are selected by projecting 3D object bounding boxes onto sampled video frames and depth maps, and identifying views where each object is both visible and unobstructed. Object-centric metadata—including object categories and 3D coordinates—is encoded as text and used as part of the prompt input.

Algorithm 1 outlines the core procedure for constructing the Struct2D prompt. Given an input video \mathbf{V} , depth frames \mathbf{D} , a reconstructed 3D scene \mathcal{P} , and a set of target objects \mathcal{O} , we begin by rendering a BEV image v and projecting each object $o_i \in \mathcal{O}$ into the view using the RGB camera parameters C_{rgb} . The 2D projections are then drawn as object marks on the image.

To select keyframes, we sample N RGB-D frames and iteratively check for visibility of objects not yet covered in the BEV. For each candidate frame I_i , we project the remaining unseen objects onto both the frame and its depth map. If a valid projection exists (i.e., the projected location lies within the image and has valid depth), the object mark is rendered and the frame is added to the keyframe set $\mathcal{I}_{\text{keys}}$. This process continues until all relevant objects are covered. The final prompt consists of ❶ a BEV image with filtered marks, ❷ optional keyframes containing visible objects, and ❸ object metadata text, all of which are passed to a multimodal large language model for reasoning.

This framework allows the MLLM to perform 3D spatial reasoning from 2D visual and textual inputs, without requiring direct access to raw 3D data at inference time. It enables scalable, flexible spatial understanding grounded in realistic perception outputs.

Algorithm 1 Struct2D Visual Prompting

Input: Input video \mathbf{V} , Depth frames \mathbf{D} , Reconstructed 3D scene \mathcal{P} , Objects of interest \mathcal{O} , RGB camera parameters C_{rgb} , Depth camera parameters C_{d}

```

1: Render a Bird’s Eye View image:  $\mathbf{v} \leftarrow \text{BEV}(\mathcal{P})$ 
2: for  $o_i \in \mathcal{O}$  do
3:   Project  $o_i$  onto  $\mathbf{v}$ :  $p_i \leftarrow \text{Project}(o_i, \mathbf{v}, C_{\text{rgb}})$ 
4:   Update view:  $\mathbf{v} \leftarrow \text{Add-Mark}(\mathbf{v}, p_i)$ 
5: end for
6: Sample  $N$  frames:  $\mathbf{I}, \mathbf{D_I} \leftarrow \text{Sample}(\mathbf{V}, \mathbf{D})$ 
7: Initialize key frame set:  $\mathcal{I}_{\text{keys}} \leftarrow \{\}$ 
8: Initialize found objects set:  $\mathcal{O}_F \leftarrow \{\}$ 
9: for  $\mathbf{I_i} \in \mathbf{I}$  and  $\mathbf{D_i} \in \mathbf{D_I}$  do
10:    $b_i \leftarrow \text{False}$ 
11:   for  $o_j \in \mathcal{O}$  and  $o_j \notin \mathcal{O}_F$  do
12:     Project  $o_j$  onto  $\mathbf{I_i}$  and  $\mathbf{D_i}$ :  $p_j^I \leftarrow \text{Project}(o_j, \mathbf{I_i}, C_{\text{rgb}})$ ,  $p_j^D \leftarrow \text{Project}(o_j, \mathbf{D_i}, C_{\text{d}})$ 
13:     if  $p_j^I \in \mathbf{I_i}$  and  $p_j^D \in \mathbf{D_i}$  and  $p_j^D \geq 0$  then
14:        $b_i \leftarrow \text{True}$ 
15:       Update frame:  $\mathbf{I_i} \leftarrow \text{Add-Mark}(\mathbf{I_i}, p_j^I)$ 
16:       Add object to set:  $\mathcal{O}_F \leftarrow \mathcal{O}_F \cup \{o_j\}$ 
17:     end if
18:   end for
19:   if  $b_i$  then
20:     Add to key frame set:  $\mathcal{I}_{\text{keys}} \leftarrow \mathcal{I}_{\text{keys}} \cup \{\mathbf{I_i}\}$ 
21:   end if
22: end for

```

Return: Informative BEV view \mathbf{v} and key frame set $\mathcal{I}_{\text{keys}}$

Qualitative comparison of our Struct2D prompting. To better understand the impact of prompt design on spatial reasoning, we conduct qualitative analyses highlighting two key aspects of our

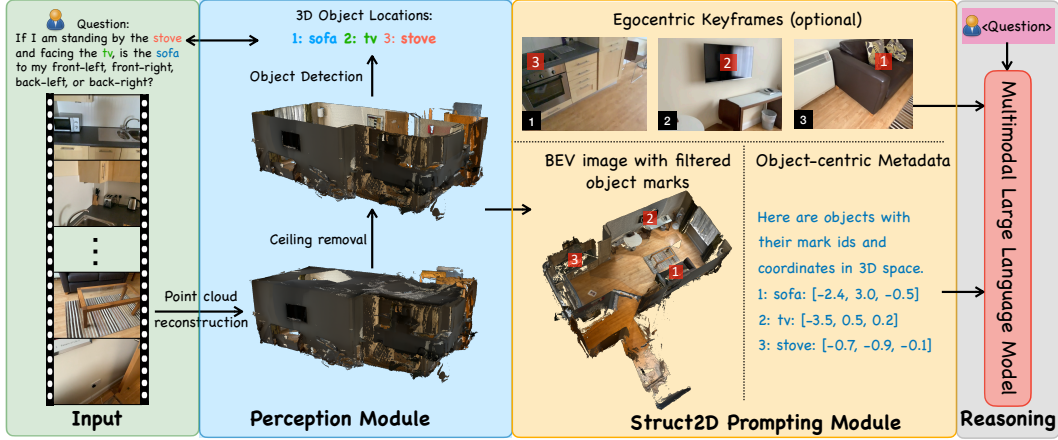


Figure 7: **Overview of the Struct2D Prompting Framework.** Given an egocentric video and a spatial question, we first reconstruct a 3D point cloud and remove the ceiling for a clear top-down view. Objects are detected in 3D space, and a bird’s-eye-view (BEV) image is rendered with object marks projected onto the floor plane. These object marks are filtered based on the content of the question. We also extract egocentric keyframes by projecting 3D bounding box centers onto the video, when appearance cues are needed. Object-centric metadata—including object IDs and 3D coordinates—is encoded as text. The structured 2D visual and textual inputs are then fed into a multimodal large language model for spatial reasoning.

framework: reasoning guidance, object orientation, and structured metadata. As shown in Figure 8, when the model is prompted only with a BEV image and object marks, it struggles to accurately resolve relative spatial relationships. Adding a structured guide prompt enables the model to decompose the task into interpretable geometric steps, though it may still fail without an aligned reference frame. Once the BEV is rotation-aligned with the agent’s viewpoint, the reasoning becomes more intuitive, leading to the correct answer. Similarly, in Figure 9, we illustrate the benefit of object-centric metadata. Without access to precise coordinates, the model must estimate distances visually, which can lead to errors. When provided with 3D object positions, the model can directly compute spatial relations such as Euclidean distances, significantly improving its accuracy on localization tasks. These examples highlight how prompt structure—through guided reasoning and geometric priors—plays a crucial role in unlocking spatial understanding in MLLMs.

B Details of Struct2D-Set

Overview. Struct2D-Set is a large-scale instruction tuning dataset aimed at enabling spatial reasoning and scene understanding in indoor 3D environments using only 2D projected inputs. It contains over 200K question-answer (QA) pairs derived from 6K richly annotated indoor scenes drawn from ScanNet [19], ScanNet++[88], and ARKitScenes[4]. Each QA instance is paired with structured scene- and object-level metadata, allowing models to learn spatial concepts without relying on explicit 3D feature representations during training.

The dataset spans eight categories of spatial reasoning tasks, such as object attribute identification, relative localization, and egocentric navigation. Each QA pair follows an instruction-style format and includes:

- A natural language question,
- A concise short-form answer,
- A long-form answer, when applicable, containing step-by-step reasoning or contextual elaboration,

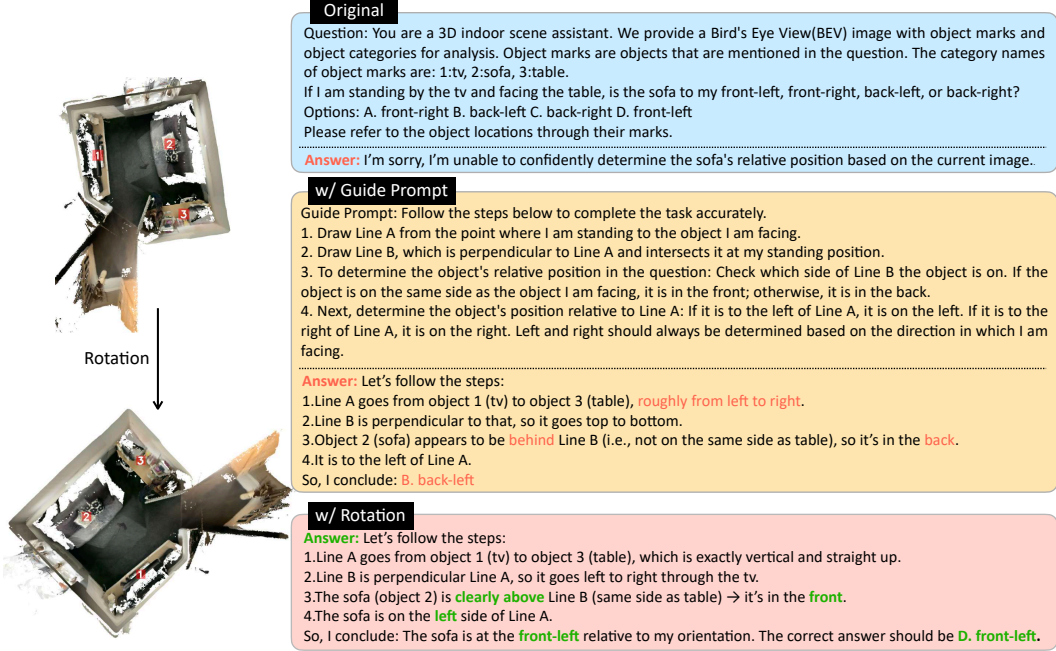


Figure 8: **Effects of guide prompting and rotation correction in relative spatial reasoning of the zero-shot prompting.** We show different variations of our Struct2D. The original model struggles to interpret spatial relationships using the raw BEV image and object marks alone, failing to answer correctly. With the addition of a structured guide prompt, the model is able to decompose the reasoning process into geometric steps, though it still arrives at an incorrect answer. When object orientation is further corrected via rotation alignment, the spatial reference frame becomes more intuitive, and the model’s reasoning becomes clearer and more accurate. This demonstrates the importance of both prompting design and canonical alignment in unlocking MLLMs’ spatial understanding abilities. (Red texts are wrong answers; Green texts are correct ones.)

- Accompanying metadata including relevant object marks, spatial coordinates, and references to visual input modalities (e.g., BEV image, selected keyframes).

Long-form answers are provided selectively for tasks that benefit from explicit reasoning or contextual understanding. For categories requiring direct factual responses—such as object counting or binary verification—only short-form answers are used. This balanced design ensures effective supervision across tasks of varying complexity, while maintaining interpretability and richness in reasoning. We next describe the construction process for each task category in detail.

Object counting. To construct object counting questions, we begin by sampling a scene from the training split of the source datasets and extracting its ground-truth object annotations. A target object category (e.g., *chair*) is randomly selected from the annotated instances within the scene. A QA pair is then generated using a templated prompt such as “How many class label(s) are there in this room?”, paired with the correct numerical count as the answer. To improve linguistic diversity and fluency, we further augment these questions by prompting ChatGPT to generate alternative phrasings with equivalent semantic meaning.

Spatial Relationship. This category evaluates a model’s ability to reason about the directional relationships between objects in a 3D scene from an egocentric perspective. Following the formulation in VSI-Bench [82], we focus on the subtask of *relative direction*, where the goal is to identify the directional location of a target object based on a specified standing point and facing direction.

To construct each QA pair, we begin by computing the 3D centers of all objects in the scene and projecting them onto the 2D BEV image. We then sample a triplet of objects representing the roles of *standing*, *facing*, and *target*, while filtering out ambiguous categories (e.g., object clusters or large connected instances) and enforcing a minimum pixel-distance threshold to ensure spatial distinguishability. The $\langle \text{standing}, \text{facing} \rangle$ vector defines the forward direction of the agent, and the

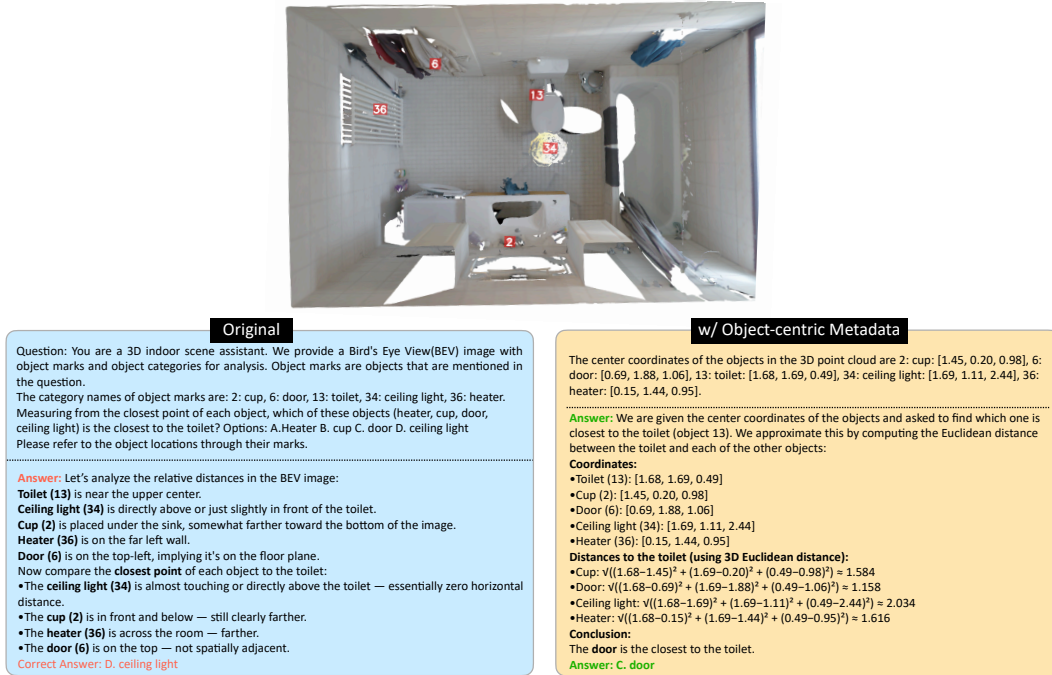


Figure 9: **Effect of object-centric metadata for precise spatial reasoning.** Originally, the model attempts to estimate distances based solely on the spatial layout in the BEV image but fails to identify the correct object closest to the toilet. In contrast, with access to object-centric metadata—specifically, 3D coordinates of each object—the model can compute accurate Euclidean distances and correctly identify the nearest object. This example highlights how structured metadata enhances geometric reasoning and helps avoid ambiguity in visual interpretation. (Red text indicates incorrect reasoning; Green text indicates the correct answer.)

$\langle \text{standing}, \text{target} \rangle$ vector is used to determine the relative orientation of the target object. The angular offset between these vectors is then discretized into directional bins such as *front-left*, *right*, or *back*, producing the correct label.

We format each QA pair using a natural language template (e.g., “If I am standing by the TV and facing the refrigerator, is the sink to my left, right, or back?”) and provide the short-form directional answer. To enhance both linguistic variation and model supervision, we further augment each instance using ChatGPT, which paraphrases the question and generates a long-form answer that walks through the step-by-step reasoning process under the egocentric frame of reference.

Comparative Reasoning. This category involves tasks where the model must compare spatial attributes among multiple objects. We focus on *relative distance comparison*, where the objective is to identify which candidate object is closest or farthest from a given reference object.

To construct such questions, we first select a reference object whose identity is unambiguous based on its class label. Next, we sample a set of candidate objects, including multiple instances—potentially of the same class—to encourage instance-level discrimination. In contrast to reference selection, we do not filter ambiguous or repeated categories among the candidates, as the goal is to challenge the model to reason over instance-specific spatial relations.

We compute the 3D centroid of each object using the center of its oriented bounding box and measure pairwise Euclidean distances between the reference and each candidate. Based on the ranking of these distances, we generate a templated question, such as “Measuring from the closest point of each object, which of these objects (candidate labels) is closest to the reference object?”, along with the correct answer derived from the computed rankings.

To enhance linguistic variation and encourage deeper reasoning, we further augment each instance using ChatGPT, which paraphrases the question and generates a long-form answer. These enriched responses guide the model through comparative spatial reasoning before producing the final answer.

Quantitative Spatial Measuring. This category targets tasks requiring the model to reason about metric properties in 3D space, such as object size, spatial extent, and inter-object distance. We focus on the *object absolute distance* subtask, where the model needs to estimate the physical distance between two specified objects within a scene.

To construct these questions, we begin by selecting two distinct objects with clearly identifiable class labels to avoid semantic ambiguity. Using the oriented bounding box annotations, we extract all eight corner points for each object and compute the minimum Euclidean distance across all point pairs—this serves as the ground-truth physical distance between the two objects. Based on this calculation, we generate templated questions such as: “*Measuring from the closest point of each object, what is the distance between the object1 and the object2 (in meters)?*”

To enhance supervision and promote reasoning transparency, we further use ChatGPT to produce long-form answers. These responses walk through the spatial computation process, prompting the model to conceptually simulate pairwise distance comparisons before arriving at the correct numerical answer.

Egocentric Navigation. This category focuses on tasks that require the model to plan navigation routes from an egocentric perspective, reasoning about object references, turning actions, and scene layout. The goal is to simulate how an embodied agent would traverse a 3D space by following instructions grounded in object-level references.

To construct these tasks, we first sample up to 15 candidate objects per scene and project their 3D centers onto the BEV image. Each object is visually marked in the BEV, and a mark-to-label dictionary string is generated to facilitate object identification. These scene representations are then passed to ChatGPT to generate plausible navigation routes in natural language.

Route generation is guided by several constraints: ❶ Each route must consist of a sequence of consecutive object marks (IDs) that an agent can follow. ❷ At each step, the agent must perform a local navigation action (e.g., turn left, turn right, pass by). ❸ Routes must avoid collisions with irrelevant or obstructing objects. ❹ Each path should span 3 to 5 objects to ensure sufficient reasoning complexity.

All generated routes undergo human review to ensure spatial plausibility. Invalid routes are discarded, and valid ones are further augmented via route reversal and sub-segmentation to increase diversity.

To determine the action sequence along the path, we randomly choose a facing object at the starting point to establish the initial egocentric orientation. For each transition between objects, we compute the vector from the current object to the next and compare it with the current facing direction to infer the correct action (e.g., turn left, go forward). These navigation actions form the short-form answer.

For each object along the route, we apply our keyframe selection algorithm to extract egocentric views from the original video. These keyframes, combined with the object labels, are used to prompt ChatGPT to generate rich textual descriptions of each waypoint. Finally, we instruct ChatGPT to produce long-form answers that walk through the full navigation route, step by step, reasoning over orientation shifts and identifying the appropriate navigation action at each stage.

Other Categories. The remaining task types—such as object attribute identification and binary attribute verification—are constructed by augmenting QA pairs from existing 3D vision-language datasets, including ScanQA [3], SQA3D [53], Scan2Cap [14], ScanRefer [10], and Multi3DRefer [95]. These datasets provide scene-specific questions grounded in the ScanNet environment and collectively cover all eight spatial reasoning categories defined in Struct2D-Set.

To adapt these examples for instruction tuning, we first use ChatGPT to rephrase each question into a more natural and instructional style. For tasks requiring reasoning, we also prompt ChatGPT to generate long-form answers that walk through the inference process. For visual grounding, we localize referenced objects using two approaches: when object IDs are available, we apply our keyframe detection method to extract representative egocentric views. In datasets with descriptive references (e.g., SQA3D), we extract referring expressions with ChatGPT and apply Grounding DINO [51] and SAM [36] to identify and segment the mentioned objects.

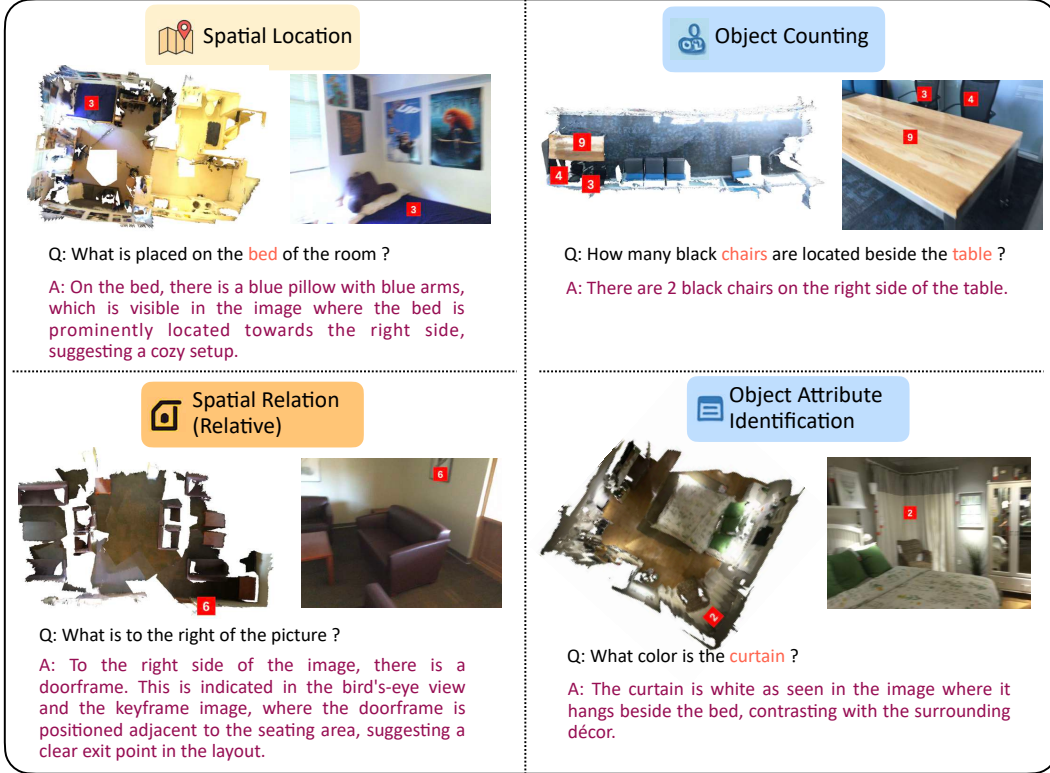


Figure 10: **Examples of QA pairs from Struct2D-Set used for instruction tuning.** Each example illustrates a distinct category of spatial reasoning: spatial localization, object counting, spatial relationship, and object attribute identification. For each question, the model is provided with a BEV image annotated with object marks, and optionally an egocentric keyframe to enhance visual grounding. The answers include descriptive reasoning grounded in object positions and appearances, enabling the model to learn to associate structured 2D inputs with fine-grained spatial understanding.

The resulting keyframes are paired with each QA instance to serve as visual inputs during fine-tuning. This pipeline enables instruction tuning on complex, object-centric spatial tasks while relying only on 2D visual projections and avoiding the need for explicit 3D geometry at training time.

C Implementation Details

We use Qwen2.5VL [72] as the base multimodal large language model (MLLM) for instruction tuning. During training, the model receives BEV images with filtered object marks and object-centric metadata as core inputs. For tasks requiring visual cues such as object color or quantity, we additionally provide egocentric keyframes. The BEV images are resized to 640×640 , with object marks adaptively scaled based on their original image resolution. Keyframes are resized to 256×246 and stitched into compact 1×2 or 2×4 grids, enabling efficient batch loading and reducing GPU memory consumption.

To support reasoning supervision, we adopt a task-specific output format. For complex spatial reasoning tasks—such as relative direction estimation or route planning—we wrap the reasoning process between special tokens `<think>` and `</think>`, followed by the final answer enclosed within `<answer>` and `</answer>`. For tasks focused on appearance or simple measurements, the model is trained to generate direct short-form answers without explicit reasoning traces. The model is trained for one epoch using a base learning rate of $2e-6$ with cosine annealing, taking approximately 8 hours on $8 \times H200$ GPUs.

At evaluation time, we follow standard practices from prior work [31, 63], reconstructing point clouds offline using BundleFusion [18], detecting 3D objects using Mask3D[66] and UniDet [37],

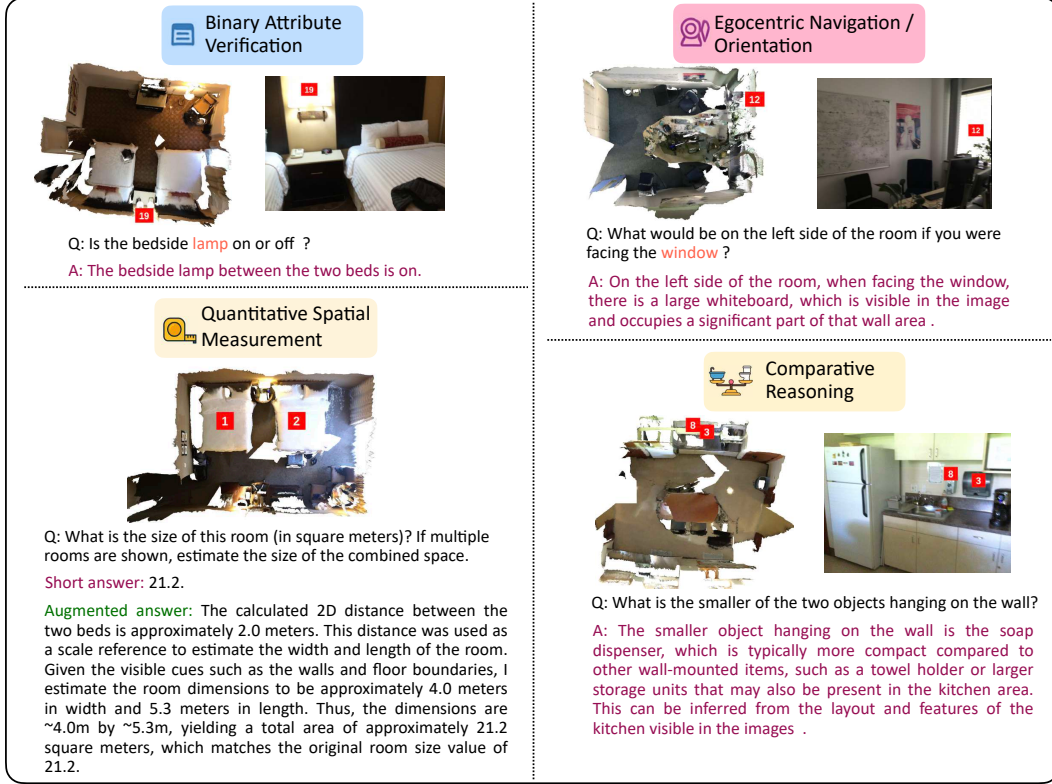


Figure 11: **Additional QA examples from Struct2D-Set covering diverse spatial reasoning categories.** This figure showcases examples from the remaining categories in our dataset: binary attribute verification, egocentric navigation and orientation, quantitative spatial measurement, and comparative reasoning. Each QA pair is grounded in structured 2D visual inputs (BEV views and keyframes) and enriched with object marks and contextual metadata. These examples demonstrate the model’s ability to reason about object states, egocentric spatial references, metric estimations, and relative comparisons—key competencies for embodied spatial understanding.

and projecting the results to produce BEV images and 2D object marks. For object-level grounding, we apply a rule-based method to identify the relevant objects mentioned in each question.

D Results on 3D Grounding and 3D Dense Captioning

Quantitative results. Tables 6 and 7 present our model’s performance on 3D grounding (ScanRefer, Multi3DRefer) and dense captioning (Scan2Cap) benchmarks. While our method does not achieve the highest scores under rule-based metrics such as all F1@0.25/0.5 and BLEU/ROUGE, it consistently delivers competitive results compared to existing vision-language baselines. Importantly, our approach does not rely on point cloud features during training or evaluation, in contrast to task-specific and 3D LLM models that depend heavily on explicit 3D representations. In addition, our approach requires substantially fewer egocentric keyframes on average (2 compared to 8 in GPT4Scene [63]), resulting in a more efficient and scalable training process. Compared to models designed for narrow tasks, our framework is more general and supports a wider range of spatial reasoning types, including relative direction and route planning, which are not covered by these benchmarks. It is also worth noting that the current evaluation metrics are rule-based and limited in expressiveness, which may not fully reflect a model’s capability in spatial understanding.

Qualitative results. Figure 12 illustrates qualitative examples of our fine-tuned Qwen2.5-VL-7B model across three major spatial reasoning tasks: 3D dense captioning, object grounding, and 3D question answering. In each case, the model receives a BEV image with object marks, optionally supplemented with egocentric keyframes and metadata, and produces either a descriptive caption, an

Table 6: **3D Grounding Evaluation on ScanRefer [10] and Multi3DRefer [95] datasets.**

Methods	ScanRefer (val)		Multi3DRefer (val)	
	Acc@0.25	Acc@0.50	all F1@0.25	all F1@0.50
<i>Task-Specific Model</i>				
3DVG-Transformer [97]	47.6	34.7	–	25.5
3DJCG [6]	49.6	37.3	–	26.6
D3Net [11]	–	37.9	–	32.2
M3DRef-CLIP [95]	51.9	44.7	42.8	38.4
<i>3D LLM Based Model</i>				
Chat-Scene [31]	55.5	50.2	57.1	52.4
<i>Vision LLM Based Model</i>				
Qwen2-VL-7B [72]	5.4	5.1	21.1	19.9
Qwen2-VL-7B (GPT4Scene [63])	40.5	36.7	45.4	42.1
Qwen2.5-VL-7B (Ours)	51.7	48.5	42.1	40.6

Table 7: **3D Dense Captioning Evaluation on Scan2Cap [14] dataset.**

Methods	IoU@0.25		IoU@0.5	
	BLEU-4	ROUGE	BLEU-4	ROUGE
<i>Task-Specific Model</i>				
Scan2Cap [14]	34.2	55.3	23.3	44.5
3DJCG [6]	40.2	59.2	31.0	50.8
X-Trans2Cap [90]	35.7	54.7	25.1	45.3
3D-VisTA [102]	36.5	57.6	34.0	54.3
Vote2Cap-DETR [12]	39.3	59.3	34.5	54.4
<i>3D LLM Based Model</i>				
LL3DA [13]	41.4	59.5	36.8	55.1
LEO [32]	–	–	36.9	57.8
Chat-Scene [31]	38.2	60.6	36.3	58.1
Robin3D [35]	–	–	38.4	–
<i>Vision LLM Based Model</i>				
Qwen2-VL-7B [72]	3.8	24.7	3.8	24.6
Qwen2-VL-7B (GPT4Scene [63])	36.3	57.6	34.2	55.2
Qwen2.5-VL-7B (Ours)	34.8	57.0	32.7	54.5

object ID, or a short-form answer. The examples demonstrate the model’s ability to reason about visual attributes (*e.g.*, “a brown rectangle”), relative spatial positions (*e.g.*, “the table is to the right of the couch”), and numerical or commonsense questions. We observe that the model often produces answers consistent with the ground truth, and in some cases offers additional descriptive clarity grounded in the visual context. These results highlight the effectiveness of our Struct2D prompting strategy in enabling rich spatial understanding from structured 2D inputs.

E Failure cases

To better understand the limitations of our approach, we conducted a qualitative error analysis on 30 representative questions spanning multiple QA types in VSI-Bench. Among the 16 failure cases, we identified two dominant causes. First, in 11 cases, the underlying 3D reconstruction was noisy or incomplete, producing degraded BEV projections that obscured critical spatial layouts. Second, in 5 cases, missing detections—often involving small or heavily occluded objects—led to incomplete structured inputs. Both factors reduce Struct2D’s ability to encode accurate spatial cues, thereby hindering its reasoning capability. Representative visualizations of these failure modes are shown in

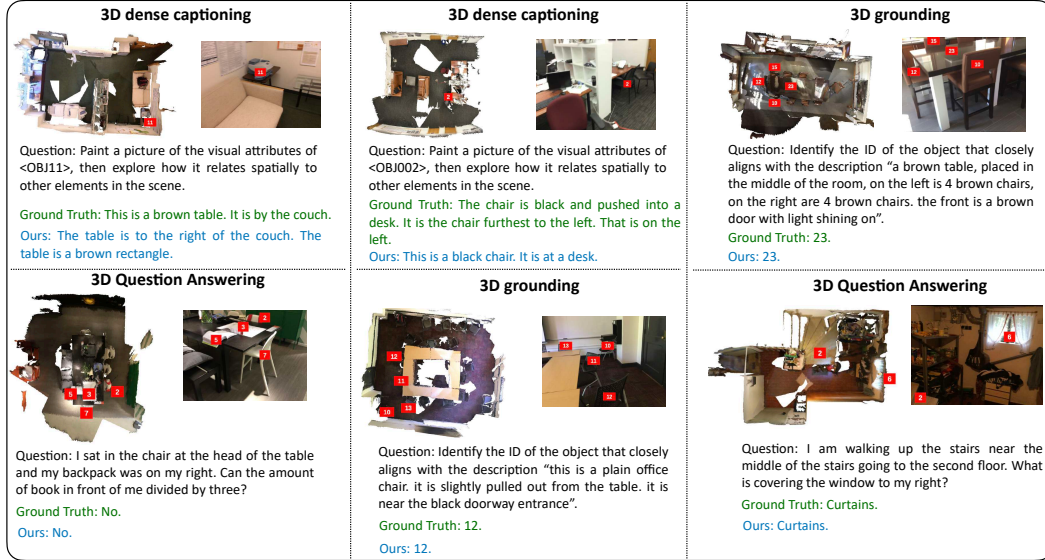


Figure 12: **Output examples from our fine-tuned Qwen2.5-VL-7B model across multiple 3D spatial reasoning tasks.** The figure showcases model responses on 3D dense captioning, object grounding, and 3D question answering tasks. Each example includes the question, BEV and keyframe inputs with object marks, the ground-truth answer, and our model’s prediction. These examples illustrate the model’s ability to localize, describe, and reason about spatial relations using structured 2D prompts derived from 3D scenes. Across tasks, the model demonstrates strong alignment with ground-truth answers, even when questions require appearance attributes, relative spatial context, or numerical reasoning.

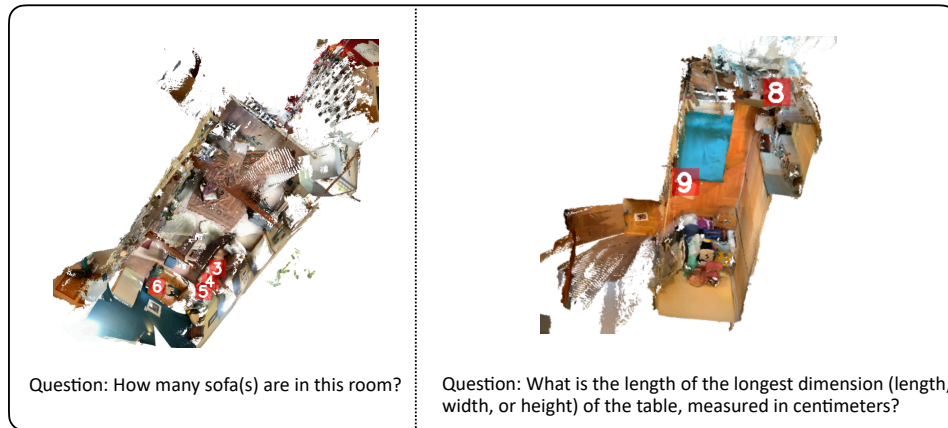


Figure 13: Examples of failure cases caused by 3D reconstruction and detection.

Figure 13. These examples highlight the importance of robust 3D reconstruction and reliable object grounding for spatial reasoning in complex indoor scenes.

F Broader impacts

Our work introduces Struct2D, a perception-guided prompting framework that enables Multimodal Large Language Models (MLLMs) to perform robust spatial reasoning in 3D environments using only structured 2D and text inputs. This direction offers several broader implications for research, society, and the reasonable development of AI systems.

Social Benefits. Struct2D lowers the barrier to 3D spatial reasoning by leveraging RGB-D perception instead of requiring dense 3D annotations or point cloud inputs during inference. This makes spatial understanding more accessible to a wide range of applications, especially in settings where real-time 3D sensing is noisy, sparse, or unavailable. Potential downstream applications include:

- **Assistive robotics**, where spatial-language understanding is critical for navigation and object manipulation in dynamic home environments;
- **Augmented reality interfaces**, where natural-language spatial queries must be resolved in partially reconstructed environments;
- **Accessibility technologies**, especially for users with visual impairments, by enabling robust, language-driven scene understanding with minimal hardware.

Potential Negative Impacts. The preprocessing pipeline relies on egocentric video and 3D reconstruction, which may involve scenes from private homes or workplaces. If deployed in real-world applications, such systems may inadvertently capture sensitive spatial or personal data. Ensuring strict anonymization, access control, and user consent mechanisms is essential.

Research Contributions. By decoupling MLLM training from explicit 3D input requirements, Struct2D promotes research into modular, scalable instruction-tuning pipelines that can generalize across environments with different sensor setups. Furthermore, our public release of Struct2D-Set—a large-scale spatial instruction dataset built with a principled blend of structured prompts, egocentric frames, and metadata—contributes valuable benchmarks to the broader vision-language community.