

Technical Appendices and Supplementary Material

A Broader Impacts

This work advances spatial reasoning in multimodal large language models by enabling 3D understanding purely from 2D visual inputs. Such capability may broaden the accessibility of spatially aware AI in domains such as robotics, autonomous systems, and visual content understanding, without requiring costly 3D data. As with other vision-language models, considerations of data privacy and ethical deployment remain important to ensure positive social outcomes.

B Additional Method Details

B.1 Details of Space-Aware Frame Sampling

Our space-aware frame sampling algorithm consists of three stages: (1) Scene geometry preprocessing, (2) Voxelization and coverage calculation, and (3) Greedy maximum coverage selection. Beginning with the original video sequence $\mathcal{V} = \{\mathbf{f}_i\}_{i=1}^N$, we first perform uniform subsampling to obtain $N_m = 128$ candidate frames $\{\mathbf{f}_i^m\}_{i=1}^{N_m}$. For each subsampled frame, we leverage the backbone and head of VGGT [32] to compute $\{\mathbf{E}_i^m, \mathbf{K}_i^m\}_{i=1}^{N_m}$ and $\{\mathbf{D}_i^m\}_{i=1}^{N_m}$ as illustrated in the main paper. Then we reconstruct 3D point maps \mathcal{P}_i^m through depth reprojection:

$$\mathcal{P}_i^m = \mathbf{D}_i^m \cdot \mathbf{K}_i^{-1}[\mathbf{u}|\mathbf{v}|1]^\top \cdot \mathbf{E}_i^{-1}, \quad (8)$$

where (\mathbf{u}, \mathbf{v}) denote pixel coordinates. In practice, we also obtain a confidence value $c(p) \in [0, 1]$ for each point $p \in \mathcal{P}_i^m$ from the depth head. Although VGGT [32] can also directly decode point maps from 3D dense features, we find that using depth and camera produces more accurate results.

The voxelization and coverage calculation process first establishes a 3D bounding box encompassing all valid scene points:

$$\mathcal{P}_{\text{valid}} = \bigcup_{i=1}^{N_m} \{p \in \mathcal{P}_i^m \mid c(p) > 0.1 \wedge c(p) \geq \text{Percentile}(\{c(p)\}, 50\%)\}. \quad (9)$$

We then discretize the bounding box into voxels. To handle relative scales in VGGT [32] outputs, we use an adaptive way to set the voxel size Δ to $\frac{1}{\lambda}$ of the minimum dimension of the scene’s bounding box:

$$\Delta = \frac{1}{\lambda} \cdot \min(\max(\mathcal{P}_{\text{valid}}) - \min(\mathcal{P}_{\text{valid}})), \quad (10)$$

where λ is a hyperparameter and we set it to 20. Each frame’s voxel coverage $V(\mathbf{f}_i^m)$ is then calculated by discretizing its valid points:

$$V(\mathbf{f}_i^m) = \left\{ \left\lfloor \frac{p - \min(\mathcal{P}_{\text{valid}})}{\Delta} \right\rfloor \mid p \in \mathcal{P}_i^m \cap \mathcal{P}_{\text{valid}} \right\}. \quad (11)$$

Finally, we can formulate frame selection as the typical maximum coverage problem [62]:

$$\max_{\mathcal{S} \subseteq \{1, \dots, N_m\}} \left| \bigcup_{i \in \mathcal{S}} V(\mathbf{f}_i^m) \right| \quad \text{s.t.} \quad |\mathcal{S}| = N_k, \quad (12)$$

In practice, we set $N_k = 16$ and use a greedy approach [63, 25] to iteratively select the frame that provides the maximum new coverage, which is illustrated in Algorithm 1.

B.2 Details of Feature Fusion

Both the 2D and 3D encoders use a spatial patch size of 14. The 2D encoder further reduces the token sequence length by merging tokens spatially (2×2 adjacent tokens) and temporally (every 2 consecutive frames). As a result, the 2D encoder outputs exactly one-eighth the number of tokens compared to the 3D encoder (we exclude register and camera tokens). To align these tokens, we first apply the same spatial-temporal merging strategy as used in the 2D encoder. After merging, we rearrange the tokens into sequence, ensuring the two sets of tokens are precisely aligned in both position and number. Then we project both tokens into language model’s hidden dimension with a two-layer MLP and fuse them by element-wise addition.

Algorithm 1 Greedy Maximum Coverage Sampling

Input Frame voxel sets $\{V(\mathbf{f}_i^m)\}_{i=1}^{N_m}$, target selection size N_k

Output Selected frame indices $\mathcal{S} \subseteq \{1, \dots, N_m\}$

```
1:  $\mathcal{S} \leftarrow \emptyset$  ▷ Selected frames
2:  $\mathcal{C} \leftarrow \emptyset$  ▷ Covered voxels
3:  $\mathcal{R} \leftarrow \{1, \dots, N_m\}$  ▷ Remaining candidates
4: for  $t \leftarrow 1$  to  $N_k$  do
5:   if  $\mathcal{R} = \emptyset$  then
6:     break ▷ No remaining candidates
7:   end if
8:    $i^* \leftarrow \underset{i \in \mathcal{R}}{\operatorname{argmax}} |V(\mathbf{f}_i^m) \setminus \mathcal{C}|$  ▷ Max coverage gain
9:   if  $|V(\mathbf{f}_{i^*}^m) \setminus \mathcal{C}| = 0$  then
10:    break ▷ No additional coverage
11:  end if
12:   $\mathcal{S} \leftarrow \mathcal{S} \cup \{i^*\}$  ▷ Update selection
13:   $\mathcal{C} \leftarrow \mathcal{C} \cup V(\mathbf{f}_{i^*}^m)$  ▷ Update covered voxels
14:   $\mathcal{R} \leftarrow \mathcal{R} \setminus \{i^*\}$  ▷ Remove from candidates
15: end for
16: return  $\mathcal{S}$ 
```

B.3 Details of Dataset Construction

We follow a similar approach to that used in [18] to construct the self-created part of training dataset. Specifically, the construction involves three main processes: video preprocessing, metadata computation, and QA pair generation.

Video Preprocessing. In this stage, we extract frames from the raw ScanNet [64] scans and convert them into videos at 24 FPS with a resolution of 640×480 .

Metadata Computation. In this stage, we extract spatial and semantic metadata from raw ScanNet scans and their associated semantic annotations. First, we align each raw scene mesh using the provided axis alignment matrices and convert it to the Open3D [73] point cloud. At the room level, we compute the room size using the alpha-shape algorithm and determine the center coordinates. At the object level, we generate oriented bounding boxes (OBBs) for each valid object instance and assign semantic labels from the annotations, excluding structural elements (*e.g.*, *walls*, *floors*) and ambiguous categories (*e.g.*, *otherstructure*). To ensure consistency across categories, we remap the original ScanNet semantic labels to a new label set based on the NYU40 classes [74, 75] (which we manually add and remove some categories to align with VSIBench [18]). In addition, we collect the projected 2D semantic annotation of each scene video for the appearance order task. The final metadata for each scene includes: (1) room size and center coordinates; (2) the projected 2D semantic annotation of the scene video; (3) object instances and their OBB parameters, including rotation matrices, extents, and centers; and (4) semantic labels for each object.

QA Pair Generation. Finally, we generate QA pairs of different tasks, including object counting, object size, room size, absolute distance, appearance order, relative distance, and relative direction.

- *Object counting (numerical)*: We first count how many times each object category appears in the scene, then randomly sample a category that appears at least twice. Question template: “How many <category>(s) are in this room?”
- *Object size (numerical)*: We randomly sample a unique object in the scene and take the longest side of its oriented bounding box (OBB) as the ground-truth length (in cm). Question template: “What is the length of the longest dimension (length, width, or height) of the <category>, measured in centimeters?”
- *Room size (numerical)*: We use the pre-computed room size (in m^2) as the ground-truth value. Question template: “What is the size of this room (in square meters)?”
- *Absolute distance (numerical)*: For a pair of objects, we uniformly sample points inside each OBB and take the minimum Euclidean distance between the two point clouds as the ground-truth

(in m). Question template: “Measuring from the closest point of each object, what is the direct distance between the $\langle \text{category_A} \rangle$ and the $\langle \text{category_B} \rangle$ (in meters)?”

- *Appearance Order (multiple choice)*: We calculate the first appearance timestamp of each category, which is the timestamp when its visible pixel count exceeds a predefined threshold. Using these timestamps, we generate the correct order of appearance among the categories, along with other options. Question template: What will be the first-time appearance order of the following categories in the video: $\langle \text{category_A} \rangle$, $\langle \text{category_B} \rangle$, $\langle \text{category_C} \rangle$, $\langle \text{category_D} \rangle$
- *Relative distance (multiple choice)*: We use an “anchor” object that is unique in the scene and then select four additional objects while enforcing 15-30cm separation thresholds between options. Question template: “Which of these objects ($\langle \text{category_A} \rangle$, $\langle \text{category_B} \rangle$, $\langle \text{category_C} \rangle$, $\langle \text{category_D} \rangle$) is closest to the $\langle \text{anchor_category} \rangle$?”
- *Relative direction (multiple choice)*: For triple $\{\text{position}, \text{facing}, \text{query}\}$ of unique categories, we compute the horizontal angle between the vectors $\text{position} \rightarrow \text{facing}$ and $\text{position} \rightarrow \text{query}$. The angle is then discretized into directional classes (easy: left/right, medium: left/right/back, hard: front-left/front-right/back-left/back-right). Question template (easy example): “If I am standing by the $\langle \text{position_category} \rangle$ and facing the $\langle \text{facing_category} \rangle$, is the $\langle \text{query_category} \rangle$ to the left or the right?”

B.4 Details of Cold Start

To align the model with the desired reasoning format, we perform a simple cold start for 200 steps before GRPO training. The key to this stage is the construction of a spatial reasoning dataset with chain-of-thought (CoT) annotations. The construction process is as follows:

Subset Sampling. We begin by sampling a subset $\mathcal{D}_0 = \{\mathcal{I}_i\}_{i=1}^{N_s} = \{\langle \mathcal{Q}_i, \mathcal{A}_i, \mathcal{V}_i, \mathcal{M}_i \rangle\}_{i=1}^{N_s}$ from our training dataset.

Multi-path CoT Generation. For each item $\mathcal{I}_i \in \mathcal{D}_0$, we utilize Qwen2.5-VL-72B [14] to generate K independent reasoning processes $\hat{\mathcal{T}}_i^{(k)}$ and corresponding answers $\hat{\mathcal{A}}_i^{(k)}$. We then compute a reward $r_i^{(k)} = \text{Reward}(\hat{\mathcal{A}}_i^{(k)}, \mathcal{A}_i)$ for each reasoning-answer pair, where $\text{Reward}(\cdot, \cdot)$ is the reward function described in Sec B.5. Consequently, we obtain a set of outputs $\mathcal{O}_i = \{(\hat{\mathcal{T}}_i^{(k)}, \hat{\mathcal{A}}_i^{(k)}, r_i^{(k)})\}_{k=1}^K$ for each $\mathcal{I}_i \in \mathcal{D}_0$.

Adaptive Filtering. Since Qwen2.5-VL-72B [14] may generate incorrect reasoning processes and answers, we apply a filtering process based on the computed rewards. While using a global reward threshold is straightforward, it often results in an imbalance across question types in the selected subset. To mitigate this, we adopt an adaptive filtering strategy. Specifically, for each item $\mathcal{I}_i \in \mathcal{D}_0$, we first keep the output with the highest reward to get $\hat{\mathcal{O}}_i = \{(\hat{\mathcal{T}}_i^{(k^*)}, \hat{\mathcal{A}}_i^{(k^*)}, r_i^{(k^*)})\}$ where $k^* = \arg \max_k r_i^{(k)}$. Let $\hat{r}_i = r_i^{(k^*)}$ denote the maximum reward. We then categorize all items based on their question type and compute a question type-dependent threshold $\tau_{t(i)}$, where $t(i)$ denotes the type of problem i . The item is added into the cold start set if and only if:

$$\hat{r}_i \geq \tau_{t(i)} \quad \text{and} \quad \hat{r}_i > 0,$$

where the type-dependent threshold satisfies $\tau_{t(i)} := \text{Quantile}(\{\hat{r}_j \mid t(j) = t(i)\}, 0.5)$. This rule preserves approximately the top 50% of generations per question type while discarding degenerate (zero-reward) outputs. In practice, we set $N_s = 5000$ and $K = 3$, and finally we get 2459 items in the cold start set. We provide a pseudocode for this process in Algorithm 2:

B.5 Details of SFT and GRPO Training

Reward Calculation. Given predicted answer $\mathcal{A}_{\text{pred}}$ and ground truth answer \mathcal{A}_{gt} , the reward function $\text{Reward}(\mathcal{A}_{\text{pred}}, \mathcal{A}_{\text{gt}})$ consists of a format reward \mathcal{R}_{fmt} and a task-specific reward:

$$\text{Reward}(\mathcal{A}_{\text{pred}}, \mathcal{A}_{\text{gt}}) = \lambda_1 R_{\text{format}} + \lambda_2 \begin{cases} \mathcal{R}_{\text{MC}}, & \text{multiple-choice} \\ \mathcal{R}_{\text{MRA}}, & \text{numerical} \\ \mathcal{R}_{\text{Verbal}}, & \text{verbal} \end{cases} \quad (13)$$

Algorithm 2 Cold Start Dataset Construction

Input Original dataset \mathcal{D}

- 1: Qwen2.5-VL model M
- 2: Reward function $\text{Reward}(\cdot, \cdot)$
- 3: Sample size N_s , Paths per item K

Output Filtered dataset $\mathcal{D}_{\text{cold}}$

- 4: Initialize $\mathcal{D}_0 \leftarrow \text{Sampling}(\mathcal{D}, N_s)$
 - 5: $\mathcal{D}_{\text{cold}} \leftarrow \emptyset$
 - 6: **for** each item $\mathcal{I}_i = \langle \mathcal{Q}_i, \mathcal{A}_i, \mathcal{V}_i, \mathcal{M}_i \rangle \in \mathcal{D}_0$ **do**
 - 7: Generate K reasoning paths: $\{\hat{\mathcal{T}}_i^{(k)}\}_{k=1}^K \leftarrow M(\mathcal{Q}_i, \mathcal{V}_i)$
 - 8: Compute rewards: $r_i^{(k)} \leftarrow \text{Reward}(\hat{\mathcal{A}}_i^{(k)}, \mathcal{A}_i), \forall k$
 - 9: Select best path: $k^* \leftarrow \arg \max_k r_i^{(k)}$
 - 10: Record $\hat{r}_i \leftarrow r_i^{(k^*)}, \hat{\mathcal{O}}_i \leftarrow (\hat{\mathcal{T}}_i^{(k^*)}, \hat{\mathcal{A}}_i^{(k^*)})$
 - 11: **end for**
 - 12: Group items by type: $\{\mathcal{G}_t\} \leftarrow \text{GroupByType}(\{\hat{r}_i\})$
 - 13: **for** each question type t **do**
 - 14: Compute threshold: $\tau_t \leftarrow \text{Quantile}(\{\hat{r}_j | j \in \mathcal{G}_t\}, 0.5)$
 - 15: **end for**
 - 16: **for** each item $\mathcal{I}_i \in \mathcal{D}_0$ **do**
 - 17: **if** $\hat{r}_i \geq \tau_{t(i)}$ **and** $\hat{r}_i > 0$ **then**
 - 18: $\mathcal{D}_{\text{cold}} \leftarrow \mathcal{D}_{\text{cold}} \cup \{\hat{\mathcal{O}}_i\}$
 - 19: **end if**
 - 20: **end for**
 - 21: **return** $\mathcal{D}_{\text{cold}}$
-

where λ_1 and λ_2 are hyperparameters, both of which are set to 1 in our implementation. For *multiple-choice questions*, we implement exact match criterion:

$$\text{R}_{\text{MC}}(\mathcal{A}_{\text{pred}}, \mathcal{A}_{\text{gt}}) = \mathbb{I}(\psi(\mathcal{A}_{\text{pred}}) = \psi(\mathcal{A}_{\text{gt}})) \quad (14)$$

where $\psi(\cdot)$ performs answer normalization through whitespace stripping and $\mathbb{I}(\cdot)$ denotes the indicator function. For *numerical tasks*, we compute mean relative accuracy (MRA) [18]:

$$\text{R}_{\text{MRA}}(\mathcal{A}_{\text{pred}}, \mathcal{A}_{\text{gt}}) = \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \mathbb{I}\left(\frac{|\alpha(\mathcal{A}_{\text{pred}}) - \alpha(\mathcal{A}_{\text{gt}})|}{|\alpha(\mathcal{A}_{\text{gt}})| + \epsilon} < \tau\right) \quad (15)$$

where $\alpha(\cdot)$ normalizes numeric values, $\epsilon = 10^{-8}$ prevents division by zero, and $\mathcal{T} = \{0.50, 0.55, \dots, 0.95\}$ defines accuracy thresholds. For *verbal answer questions*, we compute a normalized similarity score using the Levenshtein ratio:

$$\text{R}_{\text{Verbal}}(\mathcal{A}_{\text{pred}}, \mathcal{A}_{\text{gt}}) = 1 - \frac{D_{\text{Lev}}(\phi(\mathcal{A}_{\text{pred}}), \phi(\mathcal{A}_{\text{gt}}))}{|\phi(\mathcal{A}_{\text{pred}})| + |\phi(\mathcal{A}_{\text{gt}})|} \quad (16)$$

where D_{Lev} denotes the Levenshtein edit distance, and $\phi(\cdot)$ represents the text normalization function. In practice, we use the implementation provided by the *Levenshtein* library. In addition to the format and task-specific rewards, we also incorporate a reasoning length reward following Video-R1 [12], which encourages the model to perform more thinking before generating the final answer.

Other Details. Figure 6 presents the prompts used in the SFT and GRPO stages. For both stages, we adopt the default system prompt of Qwen2.5-VL [14], namely, "You are a helpful assistant." In the SFT stage, the user prompt consists of a question and a type template. In the GRPO stage, the user prompt comprises a question, a question post string, and a type template. We conduct all experiments on Intel(R) Xeon(R) Gold 6430 platform with 80G NVIDIA A800 GPUs.

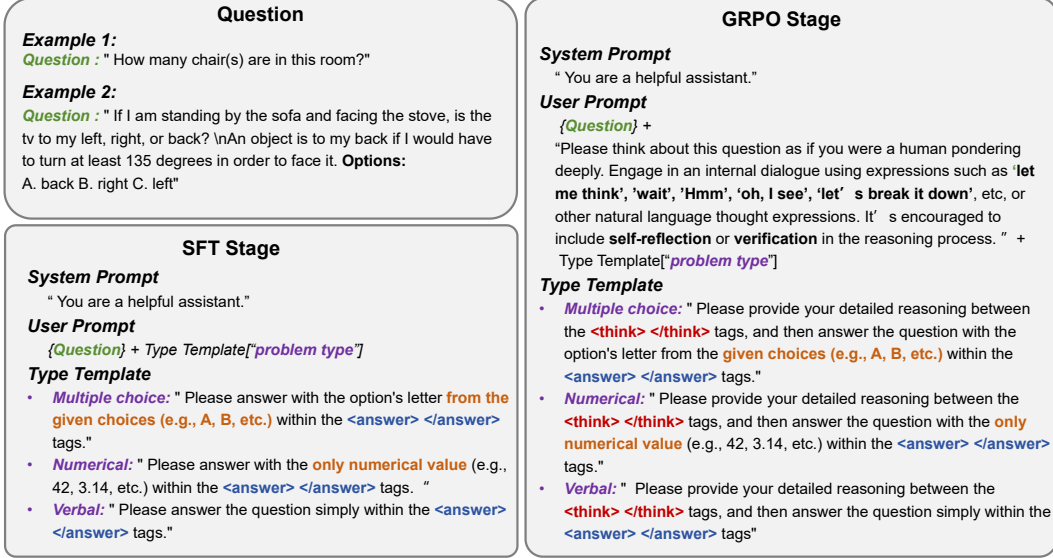


Figure 6: **Illustration of the prompts used in the SFT and GRPO stages.** We use the default system prompt of Qwen2.5-VL [14] (*i.e.*, "You are a helpful assistant") for both stages. In the SFT stage, the user prompt consists of a question and a type template. In the GRPO stage, the user prompt includes a question, a question post string, and a type template.

Table 5: Macro average scores on VSI-Bench [18] for Qwen2.5-VL [14] series and Spatial-MLLM.

Methods	Numerical Question				Multiple-Choice Question				Avg.
	Obj. Cnt.	Abs. Dist.	Obj. Size	Room Size	Rel. Dist.	Rel. Dir.	Route Plan	Appr. Order	
Qwen2.5-VL-3B [14]	24.3	24.7	31.7	22.6	38.3	42.6	26.3	21.2	29.0
Qwen2.5-VL-7B [14]	40.9	14.8	43.4	10.7	38.6	40.1	33.0	29.8	31.4
Qwen2.5-VL-72B [14]	25.1	29.3	54.5	38.8	38.2	39.3	34.0	28.9	34.3
Spatial-MLLM-4B	65.3	34.8	63.1	45.1	41.3	46.9	33.5	46.3	47.3

C Additional Experiments

C.1 Additional Results on VSI-Bench

We present qualitative examples of Spatial-MLLM on the VSI-Bench [18] dataset in Figures 7 to 10. As illustrated, Spatial-MLLM is capable of reasoning with visual information across different task types and producing final answers accordingly. Furthermore, it demonstrates strong abilities in self-verification and task decomposition during the reasoning process.

C.2 Additional Results on ScanQA and SQA3D

We present additional evaluation results on the ScanQA [38] and SQA3D [39] benchmarks in Table 6 and Table 7. As shown, our proposed method consistently outperforms all video-input models, including LLaVA-Video-7B [12] and Oryx-34B [53], both of which incorporate spatial reasoning datasets such as ScanQA [38] during training.

Despite having only 4.2 billion parameters, Spatial-MLLM significantly surpasses Qwen2.5-VL-72B [14] on the ScanQA benchmark, achieving substantial gains across multiple metrics—for instance, +2.3 EM-1, +17.6 BLEU-1, and +24.9 CIDEr. Similarly, on the SQA3D benchmark, Spatial-MLLM consistently outperforms Qwen2.5-VL-72B across all question types and overall performance, including improvements of +4.2 EM-1 and +7.8 EM-R1, with notable gains in the *Is* (+15.3) and *Which* (+13.9) categories.

Table 6: **Additional evaluation results on ScanQA [38] for task-specific models, 3D/2.5D input models, and video-input models.** Reported metrics include EM-1, BLEU-1 to BLEU-4, ROUGE-L, METEOR, and CIDEr.

Methods	ScanQA (val)							
	EM-1	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	CIDEr
<i>Task-Specific Models</i>								
ScanQA [38]	21.1	30.2	20.4	15.1	10.1	33.3	13.1	64.9
3D-Vista [71]	22.4	-	-	-	10.4	35.7	13.9	69.6
<i>3D/2.5D-Input Models</i>								
3D-LLM [48]	20.5	39.3	25.2	18.4	12.0	35.7	14.5	69.4
LL3DA [23]	-	-	-	-	13.5	37.3	15.9	76.8
Chat-Scene [22]	21.6	43.2	29.1	20.6	14.3	41.6	18.0	87.7
3D-LLaVA [21]	-	-	-	-	17.1	43.1	18.4	92.6
Video-3D LLM [25]	30.1	47.1	31.7	22.8	16.2	49.0	19.8	102.1
<i>Video-Input Models</i>								
Qwen2.5-VL-3B [14]	15.4	22.5	13.1	8.1	3.8	25.4	9.7	47.4
Qwen2.5-VL-7B [14]	19.0	27.8	13.6	6.3	3.0	29.3	11.4	53.9
Qwen2.5-VL-72B [14]	24.0	26.8	17.8	14.6	12.0	35.2	13.0	66.9
LLaVA-Video-7B [12]	-	39.7	26.6	9.3	3.1	44.6	17.7	88.7
Oryx-34B [53]	-	38.0	24.6	-	-	37.3	15.0	72.3
Spatial-MLLM-4B	26.3	44.4	28.8	21.9	14.8	45.0	18.4	91.8

Table 7: **Additional evaluation results on SQA3D [39] for task-specific models, 3D/2.5D input models, and video-input models.** In addition to the average EM-1 and EM-R1 across all questions, we also report the average EM-1 for different question types, including *What*, *Is*, *How*, *Can*, *Which*, and *Others*.

Methods	SQA3D (test)							
	What	Is	How	Can	Which	Others	Avg. (EM-1)	Avg. (EM-R1)
<i>Task-Specific Models</i>								
SQA3D [39]	31.6	63.8	46.0	69.5	43.9	45.3	46.6	-
3D-Vista [71]	34.8	63.3	45.4	69.8	47.2	48.1	48.5	-
<i>3D/2.5D-Input Models</i>								
Scene-LLM [49]	40.9	69.1	45.0	70.8	47.2	52.3	54.2	-
Chat-Scene [22]	45.4	67.0	52.0	69.5	49.9	55.0	54.6	57.5
Video-3D LLM [25]	51.1	72.4	55.5	69.8	51.3	56.0	58.6	-
<i>Video-Input Models</i>								
Qwen2.5-VL-3B [14]	34.8	52.1	39.8	52.7	45.6	47.0	43.4	45.9
Qwen2.5-VL-7B [14]	39.7	56.6	41.1	55.9	47.6	47.2	46.5	49.8
Qwen2.5-VL-72B [14]	41.7	56.3	41.5	55.6	44.5	48.0	47.0	50.9
LLaVA-Video-7B [12]	42.7	56.3	47.5	55.3	50.1	47.2	48.5	-
Spatial-MLLM-4B	45.9	71.6	55.1	69.5	52.0	53.0	55.9	58.7

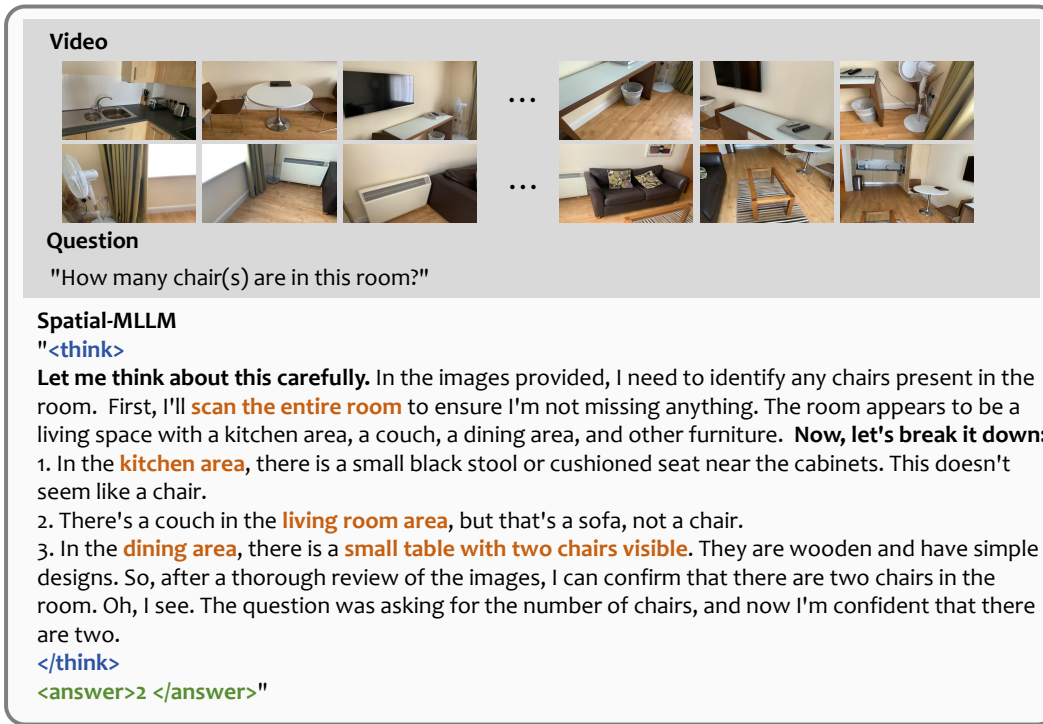


Figure 7: Qualitative example on VSI-Bench [18].

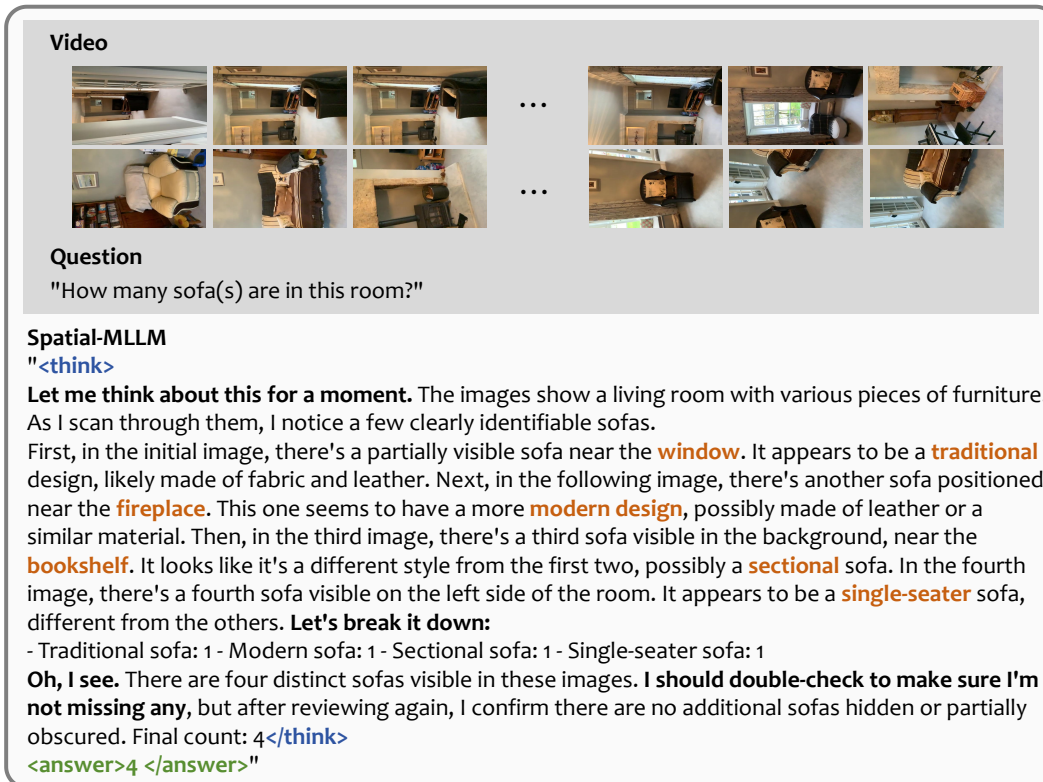


Figure 8: Qualitative example on VSI-Bench [18].

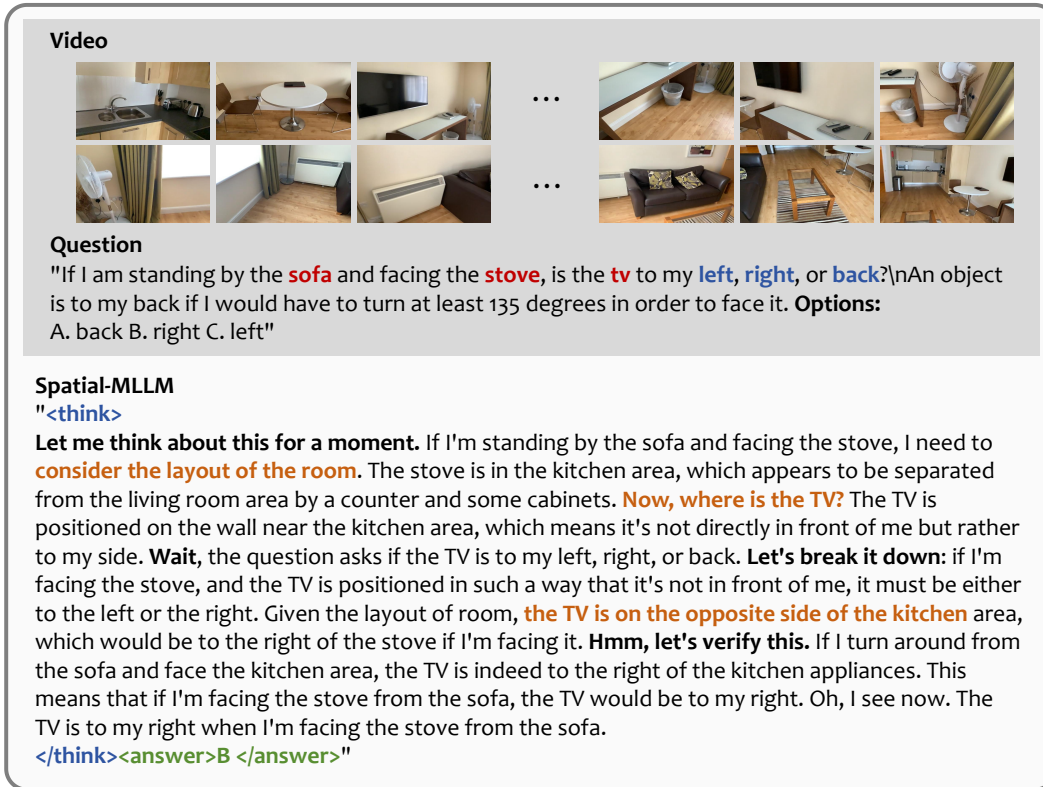


Figure 9: Qualitative example on VSI-Bench [18].

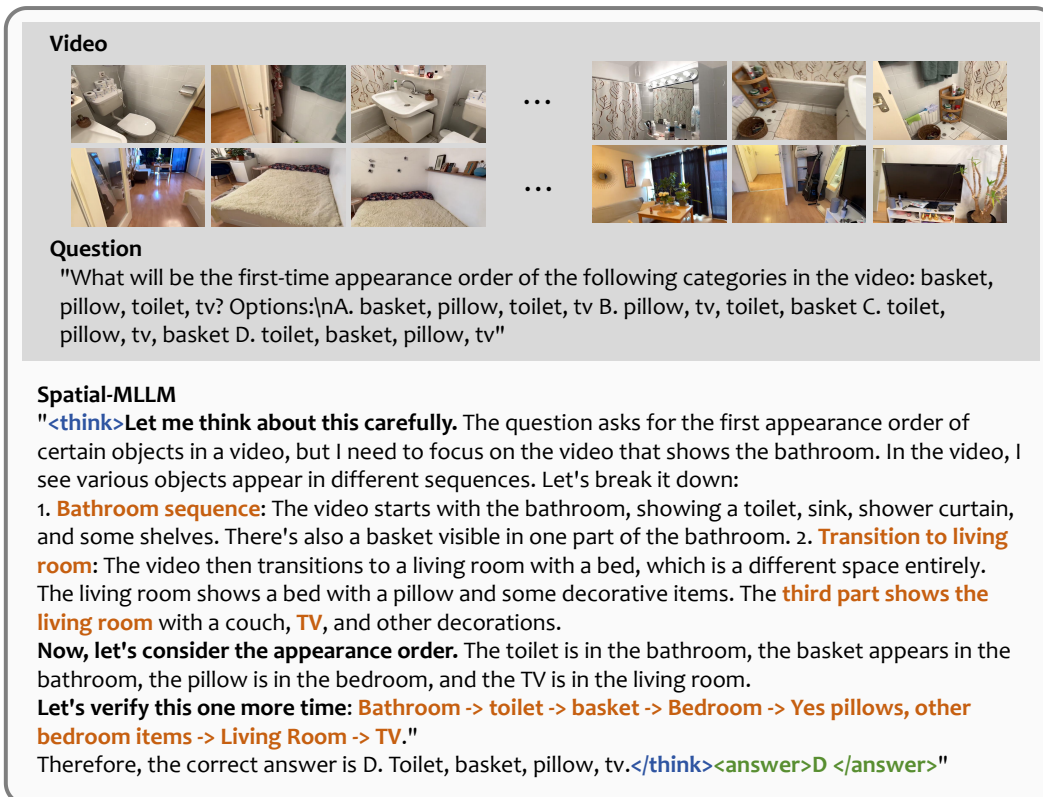


Figure 10: Qualitative example on VSI-Bench [18].