HiSpatial: Taming Hierarchical 3D Spatial Understanding in Vision-Language Models

– Supplementary Material –

Huizhi Liang[1,2,*‡] Yichao Shen[3,2,*‡] Yu Deng[2] Sicheng Xu[2] Zhiyuan Feng[1,2,‡]
Tong Zhang[4] Yaobo Liang[2] Jiaolong Yang[2]
[1]Tsinghua University [2]Microsoft Research Asia [3]Xi'an Jiaotong University
[4] University of the Chinese Academy of Sciences

1. More Implementation Details

1.1. Model Architecture

In this section, we provide detailed implementation information for our RGB-D VLM, including the parameter settings for the 3D input branch and the inference procedure for incorporating ground-truth point maps.

Auxiliary 3D input branch. The input to our 3D branch is a 3-channel metric point map, where the point values lie within the range $[-250, 250]$ (meters). We apply a 64-dimension sinusoidal positional encoding to each coordinate, resulting in a total of 192 channels. After concatenating an extra 1-channel validity mask, the input becomes a 193-channel feature map, which is then passed through a Conv2D layer with a 14^2 kernel and a stride of 14 to downsample it to the spatial resolution of the RGB feature map from the SigLIP encoder. The downsampled 3D feature map is concatenated with the RGB feature map along the channel dimension, resulting in a final feature map of 2304 channels. This combined feature map is then passed through a linear projection layer to match the embedding space of the language model for subsequent processing.

Inference with ground-truth point map. While our RGB-D VLM uses the point map estimated by MoGe-2 as the default input, it can also utilize ground-truth point maps when they are available. However, the ground-truth depth maps provided by the evaluation benchmarks (Table 1 in the main paper) are sparse or low-resolution (*e.g.*, LiDAR depth in KITTI [8] and nuScenes [2] from SpatialRGPT), which introduce a significant gap compared with the dense point-map inputs used during training. To mitigate this issue, we adopt Prior Depth Anything [18] to densify the raw GT depth maps, and sending the refined point maps derived from them as input to our RGB-D VLM for evaluation.

1.2. More Training Details

In this section, we provide additional training details, including the initialization strategy, other hyper-parameters and computation details, and the task sampling ratios used during training.

Model initialization. The Conv2D layer in our 3D branch is initialized via Kaiming initialization [9]. The linear projector can be naturally split into two parts, for RGB features and point map features respectively. The weights for the RGB part are inherited from the pretrained PaliGemma-2 model, while those for the point map are initialized to zero to ensure that introducing point map at the start of training does not negatively affect the model's performance.

Hyper-parameters and computation. During training, we use AdamW optimizer with a learning rate of 2×10^{-5} and a weight decay of 0.1. The model is trained on 32 NVIDIA H100 GPUs for approximately 2 days.

Task sampling ratios. We re-balance tasks from different levels to maintain a reasonable proportion among them during training. The detailed sampling ratios of different tasks are presented in Table 1.

1.3. Ablation Study Details

Inter-level task dependencies. In this ablation, we remove certain tasks during training as described in the main paper. To ensure a fair comparison, we maintain the default data sampling distribution. Specifically, whenever a data sample belonging to an excluded task level is drawn from the dataloader, we replace it with a general VQA sample. This replacement strategy ensures that the absolute sampling frequencies and relative proportions of all remaining tasks remain strictly identical to the default configuration.

*Equal contribution
‡Work done during internship at Microsoft Research Asia

Table 1. Task statistics of our training dataset.

Task	Level	# of QAs	Sampling Ratio in Training
Pixel-Wise 3D Point Querying	0	Online Generated	10.51%
Pairwise Depth Ordering	0	Online Generated	2.69%
Object Orientation	1	238,909,307	3.51%
Object Size	1	55,187,863	11.56%
Object Localization	1	36,469,102	8.31%
Relative Direction	2	510,297,825	26.09%
Relative Distance	2	344,264,312	13.49%
Relational Comparison	2	507,908,756	11.53%
Perspective Taking	3	709,415,000	4.22%
Spatial Object Counting	3	35,893,426	7.52%
Spatial Problem Solving	3	49,322	0.57%

Relative depth input. For the variant with relative depth input as discussed in Sec. 4.4 in the main paper, we retain the original network architecture but only modify its 3D input branch. Specifically, we replace the 3D point coordinates in the metric point map with three identical copies of the relative depth value. Each depth value is normalized and discretized to an integer in the range $[0, 255]$, following a common normalization convention used in previous approaches [3–5, 10, 19]. All other parts and training settings remain identical to the default configuration.

1.4. Evaluation Details

Benchmarks We evaluate our model on a diverse set of spatial understanding benchmarks. For qualitative tasks, including the full sets of EmbSpatial [6] and 3DSRBench [11], as well as the configuration subset of RoboSpatial-Home [16], we report Accuracy since these are framed as multiple-choice questions. For quantitative tasks involving numerical estimation, we report the Success Rate based on relative error thresholds: following SpatialRGPT [4], a prediction is successful if the relative error is within 25%; for Q-Spatial, we adopt its original criterion where a relative error within 50% is considered a success.

Ablation Study In the ablation study, we select subsets of 3DSRBench to report Level 2 (L2) and Level 3 (L3) performance:

- **3DSR-L2**: height_higher, location_above, location_closer_to_camera, location_next_to, multi_object_closer_to, multi_object_facing, multi_object_parallel, multi_object_same_direction.
- **3DSR-L3**: orientation_in_front_of, orientation_on_the_left, multi_object_viewpoint_towards_object.

2. Dataset Construction Details

2.1. Web Data Preprocessing

Image filtering. We begin with a collection of 15M web images from KosMos-2 [12] and apply a series of filtering steps to remove non-natural photographs (*e.g.*, charts, forms, GUIs, code snippets):

(1) CLIP-based semantic filtering. Following SpatialVLM, we employ CLIP [13] for semantic filtering. We construct two tag sets: an *include* set containing text descriptions indicating that an image is a natural photograph, and an *exclude* set containing those indicating that an image is non-natural. All text descriptions in both sets are encoded using the CLIP text encoder. For each image, we obtain its embedding via the CLIP image encoder and retrieve the top-5 text tags with the highest similarity. An image is retained if more than half of its retrieved tags belong to the *include* set.

(2) Heuristic filtering. We then remove low-quality or non-visual images using simple pixel-based heuristics. Specifically, we discard images in which more than 35% of the pixels are pure white or pure black, as well as images for which over 50% of pixels have invalid depth estimates, according to MoGe-2's validity masks. These empirical rules effectively filter out remaining GUI, chart, table, or blueprint images.

(3) VLM-based filtering. Finally, we adopt an VLM to refine the filtering results. Following RoboRefer, we use Qwen2.5-VL with the same prompt template to identify and remove images that still lack meaningful spatial information.

Spatial information estimation We employ several specialized models to extract object-level point clouds. Specifically, we first use MoGe-2 to estimate both the metric point map and camera intrinsics. Then, following SpatialRGPT, and as described in Sec. 3.2 in the main paper, we compute

the point clouds for all detected objects in the image.

After obtaining the raw object-level point clouds, we apply DBSCAN clustering [7] to identify objects containing multiple point cloud clusters (caused by 2D occlusions during segmentation or cases where multiple objects are grouped into a single segmentation). We select the largest point cloud cluster within each object.

Textual reference generation. We first use the Describe Anything model to produce high-quality, detailed captions for each object. Then, following a structured protocol, we use Qwen2.5-VL to generate four levels of object references conditioned on both the image and its corresponding detailed caption. The detailed prompt is shown in Fig. 6.

For each generated reference, we verify its uniqueness using a VLM-based grounding procedure. We prompt Qwen2.5-VL to localize the referred object in the original image. If the model predicts multiple bounding boxes, the reference is discarded, as it may correspond to more than one object. If exactly one bounding box is returned, we further extract the object mask using SAM and compute its IoU with the ground-truth mask. References with IoU greater than 0.7 are retained as valid. The detailed prompt is shown in Fig. 7. Finally, for each object, we select the simplest reference level that passes verification. If no generated reference passes, we fall back to a bounding-box-based reference pattern, such as: "the [object class] highlighted by the [bbox color] box".

2.2. Objects365 Data Preprocessing

Image filtering. Objects365 [15] consists of natural photographs collected from Flickr. Therefore, unlike web data, we do not apply additional image filtering.

Spatial information estimation. We follow the same spatial information estimation pipeline as described in Sec. 2.1.

Textual reference generation. For each annotated object, we directly use Qwen3-VL to generate referring expressions. The prompt template is shown in Fig. 8. We use the same verification procedure as described in Sec. 2.1 to ensure that each reference uniquely identifies a single object. The verification prompt is shown in Fig. 9.

Task-oriented QA synthesis. Compared with web data, Objects365 provides object category labels and 2D bounding box annotations. Based on these annotations, we additionally introduce a *spatial object counting* task.

2.3. CA-1M Data Preprocessing

Category labeling. Although each frame in the CA-1M dataset is annotated with relatively complete 3D bounding boxes, these 3D boxes generally lack category-level labels. In most cases, the category field is simply marked as "object" rather than a specific semantic class. Therefore, it is necessary to assign semantic category labels to these bounding boxes in order to support the subsequent generation of object references.

To begin with, we generate 2D bounding box predictions with explicit semantic categories for frames from CA-1M. Following the data pipeline described in Sec. 3.2 in the main paper, we use RAM to recognized object categories in the images and GroundingDINO to obtain corresponding 2D bounding boxes. Additionally, we leverage Qwen2.5-VL to provide a more comprehensive set of categories for GroundingDINO, complementing RAM's predictions[1]. We then match these category-aware 2D bounding box predictions with GT bounding box annotations provided by CA-1M. We compute the IoU-based cost between boxes and apply Hungarian matching to identify the optimal assignment. To ensure reliable correspondences, any matched pair whose IoU is below 0.4 is discarded. Finally, the semantic category of each matched predicted bounding box is assigned to its corresponding ground-truth box.

Spatial reference generation. Relying solely on category labels is insufficient for accurately referring to objects in downstream QA tasks, especially when multiple instances of the same category are present in a scene. Therefore, we further incorporate spatial references to disambiguate objects, complementing the textual references introduced in Sec. 2.1. When several objects share the same category label, we distinguish among them using their spatial position relationships or their size properties, such as "the second farthest chair" or "the largest pillow". More specifically, the spatial references can be categorized into the following types:

(1) Category-based reference. When there is only a single instance of a given category in the scene, the object can be directly referred to using its category label alone, such as "the remote control", "the sofa", and "the television."

(2) Global intra-category relations. When multiple objects of the same category are present, we generate references based on their global positional or size relationships. These relations primarily include the following three forms:

- *Linear ordering.* If all instances of the same category exhibit a roughly linear spatial arrangement, the object can be referred to by its ordinal position along that line. To detect such linear distributions, we apply PCA to the center positions of all objects in the category. If the second principal component is smaller than 15% of the first principal component, we treat the objects as lying along the dominant principal axis. The orientation of this axis determines whether the linear ordering corresponds to a

[1] We observe that RAM occasionally produces overly coarse labels for indoor objects, *e.g.*, assigning the generic term "furniture".

"left–right," "front–back," or "top–bottom" direction. An example is "the second picture frame from the left".

- *Positional comparison.* We generate spatial references by comparing the relative positions of objects within the same category along canonical spatial axes (*e.g.*, front, back, left, right, top, bottom, near, far). Such comparisons allow us to uniquely identify an object based on its extremal or ranked position in space. For example, "the leftmost bowl" and "the second closest table to the camera".
- *Size comparison.* Since the objects in CA-1M are annotated with accurate length, width, and height measurements, we can distinguish among instances of the same category by comparing their size properties, such as "the widest sofa" and "the largest door".

It is worth noting that the accuracy of spatial references depends on both the recall of all instances in the scene and the completeness of their category annotations, which is facilitated by the detailed spatial bounding box annotations in CA-1M dataset and further enhanced by our category labeling procedure which assigns a reliable semantic category to each bounding box.

Task-oriented QA synthesis. As described in the main paper, we use GPT-4.1 to generate QA pairs for Level-3 spatial problem-solving tasks. We construct spatial reasoning QA pairs using a multi-stage prompting pipeline grounded in structured 3D scene descriptions. Given the 3D object annotations and the corresponding RGB image, we first generate spatially grounded questions using three prompt templates: (1) basic spatial reasoning questions (Fig. 10), (2) quantitative reasoning questions requiring numerical answers (Fig. 11), and (3) prior-guided questions generated with few-shot demonstrations (Fig. 12). These prompts encourage questions that require reasoning over object positions, sizes, orientations, distances, and occlusion relations.

We then produce answers and reasoning traces through a two-stage prompting process. In Stage 1, a teacher model answers each question using privileged 3D information and outputs both a detailed reasoning trace and a set of structured *spatial facts* summarizing the key spatial relations required for solving the question (Fig. 14, 15 and 16). In Stage 2, the model is prompted to generate a simplified student reasoning trace that derives the answer solely from the provided spatial facts, without access to raw 3D geometry (Fig. 17). This pipeline produces QA pairs with structured spatial evidence and aligned reasoning traces for training spatial reasoning models.

2.4. General VQA Data Preprocessing

For the general VQA datasets from LLaVA-Next (Sec. 4 in the main paper), we adopt a simplified processing pipeline. Since these datasets already provide well-structured subsets, we do not apply the CLIP-based filtering used in our web-data pipeline (Sec. 2.1). Instead, we remove subsets containing non–real-world images, such as synthetic images, diagrams, OCR-focused content, or charts and tables, retaining only those consisting of real-world images. We then apply MoGe-2 to each remaining image to generate the corresponding point map as auxiliary 3D input.

3. Custom Benchmark

In this section, we provide more details of our custom benchmark described in Sec. 4.1 in the main paper. The benchmark contains three different types of tasks as described below.

Object-to-camera distance & relative direction estimation. We constructed these two types of tasks using Omni3D [1], which provides ground-truth point-cloud information and 3D object bounding boxes. The tasks are built upon Omni3D subsets sourced from KITTI, nuScenes, SUN RGB-D [17], and Hypersim [14], covering indoor, outdoor, and synthetic scenarios. We sample images from these sources, and apply the same pipeline as described in Sec. 2.1 to generate textual references.

After generating object references, we create QA pairs using task-specific templates described in Sec. 3.2 in the main paper. For the *distance estimation* task, every object involved in the question is required to have a unique textual reference. For the *relative direction estimation* task, at least one object mentioned in the question must have a unique textual reference, while other objects are referred to by their textual descriptions accompanied by their corresponding 2D bounding boxes.

In addition, we conduct a manual verification step to ensure the correctness of both references and QA pairs. We remove incorrect references as well as ambiguous QA examples (*e.g.*, directional relations between a sofa and the pillow placed on it).

Spatial problem solving. We use the prompt in Fig. 4 to generate problem-solving data from the *validation split* of the CA-1M dataset. The generated QA pairs are then manually inspected to verify their correctness, and inaccurate or ambiguous cases are filtered out.

While the answers to spatial problem-solving tasks may be in free-form, we employ GPT-4.1 as an auxiliary judging agent to more accurately evaluate their correctness, as illustrated in Fig. 5. For *judgement* questions—such as category identification or yes/no queries—we prompt GPT to determine whether the model's prediction matches the ground-truth answer. For *numeric* questions, GPT is instructed to verify whether the numerical prediction falls within 25% relative error of the ground-truth value.

4. More Results

4.1. More Examples of Constructed Data

Figures 1 and 2 present additional examples of the spatial VQA tasks generated by our method, covering diverse environments and tasks across different levels.

4.2. More Examples of Model Responses

Figure 3 shows additional test results of our RGB-D VLM on unseen in-the-wild images. Our model is capable of handling tasks at different levels related to 3D spatial understanding and reasoning, producing reasonable and accurate answers.

5. Limitations and Future Work

First, the model's generalization capability is currently constrained by both task complexity and linguistic diversity. On the one hand, the model primarily focuses on relatively basic spatial understanding tasks; although we include abstract spatial reasoning at Level 3, it does not comprehensively cover all types of complex reasoning. On the other hand, the procedural nature of our data generation may lead to a reliance on fixed instruction patterns. Consequently, while the model performs well on template-consistent data, its robustness in handling the highly diverse and informal language found in real-world scenarios remains to be further improved. In future work, we plan to introduce a broader set of tasks with natural language variations and incorporate reinforcement fine-tuning (RFT) to enhance the model's reasoning capabilities on complex spatial problems.

Second, although we highlight the correlations between tasks at different levels in this paper, providing valuable insights for the design of spatial understanding tasks, the current analysis serves only as a starting point. The finer-grained interactions between tasks across levels, as well as the impact of different training strategies on inter-level task relationships, require further investigation and experiments to be fully clarified.

Finally, our model currently only supports monocular input. Many spatial reasoning tasks, however, require an understanding of multi-view scenes or temporal dynamics in videos. We leave these directions for future exploration.

References

[1] Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3D: A large benchmark and model for 3D object detection in the wild. In *CVPR*, Vancouver, Canada, 2023. IEEE. 4

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 1

[3] Pingyi Chen, Yujing Lou, Shen Cao, Jinhui Guo, Lubin Fan, Yue Wu, Lin Yang, Lizhuang Ma, and Jieping Ye. Sd-vlm: Spatial measuring and understanding with depth-encoded vision-language models. *arXiv preprint arXiv:2509.17664*, 2025. 2

[4] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision-language models. *Advances in Neural Information Processing Systems*, 37:135062–135093, 2024. 2

[5] Erik Daxberger, Nina Wenzel, David Griffiths, Haiming Gang, Justin Lazarow, Gefen Kohavi, Kai Kang, Marcin Eichner, Yinfei Yang, Afshin Dehghan, et al. Mm-spatial: Exploring 3d spatial understanding in multimodal llms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7395–7408, 2025. 2

[6] Mengfei Du, Binhao Wu, Zejun Li, Xuanjing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. *arXiv preprint arXiv:2406.05756*, 2024. 2

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, pages 226–231, 1996. 3

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The international journal of robotics research*, 32(11):1231–1237, 2013. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 1

[10] Yang Liu, Ming Ma, Xiaomin Yu, Pengxiang Ding, Han Zhao, Mingyang Sun, Siteng Huang, and Donglin Wang. Ssr: Enhancing depth perception in vision-language models via rationale-guided spatial reasoning. *arXiv preprint arXiv:2505.12448*, 2025. 2

[11] Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Jieneng Chen, Celso de Melo, and Alan Yuille. 3dsrbench: A comprehensive 3d spatial reasoning benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6924–6934, 2025. 2

[12] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023. 2

[13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 2

[14] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb,

and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10912–10922, 2021. 4

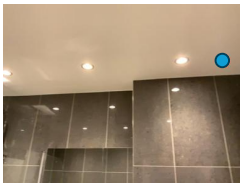
[15] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8430–8439, 2019. 3

[16] Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15768–15780, 2025. 2

[17] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015. 4

[18] Zehan Wang, Siyu Chen, Lihe Yang, Jialei Wang, Ziang Zhang, Hengshuang Zhao, and Zhou Zhao. Depth anything with any prior, 2025. 1

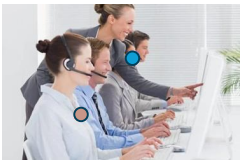
[19] Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, Pengwei Wang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, et al. Roborefer: Towards spatial referring with reasoning in vision-language models for robotics. *arXiv preprint arXiv:2506.04308*, 2025. 2



(Level 0) ***Pixel-wise 3D Point Querying***

Q*: what is the xyz of the pixel in <loc0325><loc0951><loc0325><loc0951>*

A*: 0.61m -0.20m 1.17m*



(Level 0) ***Pairwise depth ordering***

Q*: between the pixel in <loc0725><loc0338><loc0725><loc0338> and the pixel in <loc0352><loc0562><loc0352><loc0562>, which one has a smaller depth value?*

A*: the pixel in <loc0725>><loc0338><loc0725<loc0338> is closer.*

(Level 1) ***Object Localization (grounding)***

Q*: detect the aabb of the flowers*

A*: -2.39m -0.71m 3.54m -0.96m 1.51m 4.68m*

(Level 1) ***Object Size (height)***

Q*: How tall would you say the faucet appears to be?*

A*: We estimate the height of the faucet to be 0.11 meters.*

(Level 1) ***Object Size (width)***

Q*: How far does the door stretch from side to side?*

A*: We estimate the width of the door to be 0.77 meters.*

(Level 1) ***Object Localization (distance)***

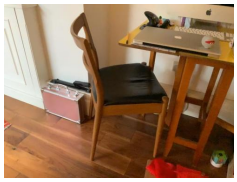
Q*: what is the distance of the signboard to the camera?*

A*: 1.94m*

(Level 1) ***Object Orientation (distance)***

Q*: Which side of the the person is most visible?*

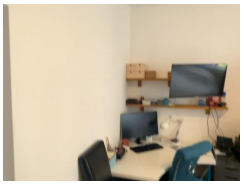
A: *back*



(Level 2) ***Relative Direction (position)***

Q*: Can we say the laptop is to the right of the desk?*

A*: No*



(Level 2) ***Relative Direction (position)***

Q*: Which relative direction best describes the desktop computer from the box?*

A*: The desktop computer can be found down of the box.*

(Level 2) ***Relative Direction (vector)***

Q*: provide the direction vector pointing from the person(highlighted by the red box) to the sign.*

A*: -0.46 0.18 -0.87*

Figure 1. Examples of spatial VQA data constructed using our method, covering different task levels.

(Level 2) *Comparison (orientation)*

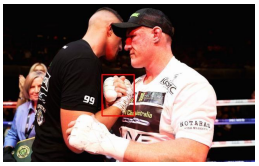
***Q**: Would you say the person (highlighted by a red box) and the person (highlighted by a blue box) are oriented similarly?*
(A). yes
(B). No

***A**: (A)*

(Level 2) *Distance (center distance)*

***Q**: Provide the distance in meters between the laptop and the person with short dark hair and a dark jacket standing near the green bush on the right side.*

***A**: 2.28 meters.*



(Level 2) *Distance (surface distance)*

***Q:** What is the surface-to-surface distance between the white boxing glove with red accents(highlighted by the red box) and the man in a white jersey?*

***A**: The surface-to-surface distance is 0.00 meters.*

(Level 2) *Comparison (height)*

***Q:** Yes or No: the young girl with light brown hair is the highest object in the person with short dark hair (highlighted by a red box) and the young girl with light brown hair.*

***A**: No, the young girl with light brown hair is not the highest one.*

(Level 2) *Comparison (depth)*

***Q:** Comparing the woman (highlighted by a blue box) and the man (highlighted by a red box), which one is farther from the camera?*
(a). the woman
(b). the man

***A:** (b)*

(Level 2) *Comparison (volume)*

***Q:** Identify the largest person.*

***A:** the glasses-wearing woman in striped shirt gently holding a sleeping newborn wrapped in a patterned blanket.*

(Level 3) *Perspective Taking*

***Q:** From the woman's viewpoint, which side is the plaque positioned most prominently?*
(A). above
(B). right
(C). below
(D). Left

***A**: (B)*

(Level 3) *Spatial Object Counting*

***Q:** How many chair are farther from the observer than the desk?*

***A**: 4*

(Level 3) *Problem Solving*

***Q:** Is there enough space between the chair and the sink for a person to walk through comfortably? Why? Please provide a detailed explanation with cot*

***A**: Step1: Plan*
To answer if there's enough space for a person to walk between the chair and sink, we need the widths of both objects and their center-to-center distance. We'll calculate the minimum edge-to-edge separation using these facts.
Step2: Facts
object_size_width: chair is 0.66m wide
object_size_width: sink is 0.16m wide
object_center_distance: chair and sink are 2.11m apart
Step3: Reasoning with facts
Using object_center_distance (2.11m), object_size_width (chair is 0.66m, sink is 0.16m), edge-to-edge separation is approximately: 2.11m - (0.66m/2 + 0.16m/2) = 2.11m - (0.33m + 0.08m) = about 1.7m, which exceeds typical comfortable walking width (~1 meter). Therefore, yes, there is enough space.
Step4: Final answer
Yes, there is enough space between the chair and the sink for a person to walk through comfortably; their edges are separated by at least about 1 meter and their centers by over 2 meters

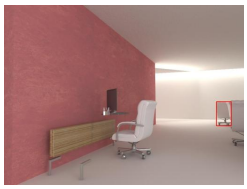
Figure 2. Examples of spatial VQA data constructed using our method, covering different task levels.

Level 3

***Question**: If two adults want to sit side-by-side on top of the ottoman facing forward, what width does each person have available assuming equal sharing? Please think step by step.*

Model Output:
Step1: Plan
We need to know the width of the ottoman and understand that two adults sharing it side-by-side would each take half of this width.
Step2: Facts
object_size_width: ottoman is 0.75m wide
Step3: Reasoning with facts
Using object_size_width: ottoman is 0.75m wide, if two adults share it equally, each person gets 0.75m / 2 = 0.375m, which rounds to 0.38 meters per person.
Step4: Final answer
Each person has 0.38 meters of width available.
<eos>

***Ground Truth Answer**: 0.334 meters per person.*



Level 2

***Question:** what is the direction between the white office chair at the bottom-center of the image and the chair (highlighted by red box)?*

***Model Output**: 0.65 0.01 0.76*

***Ground Truth Answer**: 6.71, 0.30, 8.36*

***Angular Error**: 2.04°*

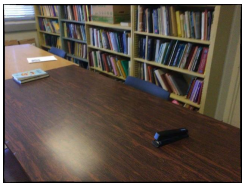
Level 1

***Question:** What is the object to camera distance of the floral picture at the center of the image?*

***Model Output**: 9.03 m*

***Ground Truth Answer**: 8.98 m*

***Relative Error**: 0.006*



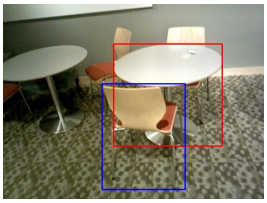
Level 2

***Question:** What is the minimum distance between the stapler and the blue book on the table?*

***Model Output**: The minimum distance between the stapler and the blue book on the table is 1.17 meters.*

***Ground Truth Answer**: 1.20 m*

***Relative Error** : 0.03*



Level 3

***Question:** If an observer is standing at the table at the left, facing toward the table (highlighted by red box), is the chair (highlighted by blue box) left or right relative to the table at the left (observer)?*
A. left
B. right

***Model Output**: B*

***Ground Truth Answer**: B*

Level 1

***Question:** Which direction is the car facing?*
A. Left
B. Right
C. Up
D. Down

***Model Output**: B*

***Ground Truth Answer**: B*

Level 2

***Question:** Are the rider facing to the similar direction with the horse?*

***Model Output**: Yes*

***Ground Truth Answer**: Yes*

Figure 3. Examples of our model's responses on unseen images.

```
You are given:
1. An image of a scene.
2. A list of objects in the scene, where each object has:
  - caption
  - category
  - 3d_box = [cx, cy, cz, sx, sy, sz]
  - orient  (a direction word that directly indicates the object's facing or orientation relative to the camera, e.g., "front", "behind", "left", "right", "above", "down")

  notice: Axis conventions (camera first-person viewpoint):
    • x axis: left (negative) to right (positive)
    • y axis: top (negative) to bottom (positive)
    • z axis: front/near (smaller) to back/far (larger)

3. A list of spatial relations, where each relation describes the relative position of one object
  (the subject) with respect to another (the reference).

Your task:

## Part 1: Object grounding
- For each object, output a grounding note with two separate fields:
  1. **image-plane position**:
    • Approximate where it appears in the 2D image plane (e.g., top-left, center, bottom-right).
  2. **appearance/function description**:
    • Summarize its caption, category, and orientation (orient).
    • Provide a **detailed description** including:
      - Its visual form (shape, material, color if implied).
      - Its typical function in daily life.
      - How people commonly interact with or use it.
    • Keep each description 1–3 sentences, realistic and plausible.

- Important:
  • Do NOT use 3D box values for this part.
  • Only rely on caption/category/orient and plausible image-plane appearance.

## Part 2: Functional QA generation
- Generate **4 new composite QA pairs**.
- Each QA must be:
  • **Realistic, daily-life, and grounded in object functions**.
  • **Function-driven**: the question must be about how a person could use, interact with, or move around the object(s).
  • **NOT allowed**: direct geometry-only questions like "How far is A from B?" unless tied to a meaningful activity.

- Examples of function-driven questions:
  • Lamp → "If someone sits at the chair, is the lamp close enough for reading?"
  • Table with space below → "What is the tallest chair that can slide under the table?"
  • Doorway/passage → "What is the widest box that can pass through?"
  • Sofa/bed → "Can a person of 1.8 m lie down fully on it?"
  • Shelf/cabinet → "What is the tallest item that can fit without hitting the top shelf?"
  • Chair + table → "If someone sits on the chair, which table is easier to reach for writing?"

- Math priors:
  1) **Walking/moving a person**: ground-plane distance (x–z).
  2) **Reaching/placing/fitting/passing**: 3D reasoning with sx, sy, sz.
  3) Always include units ("m"), round to 3 decimals.
  4) Reasoning must restate categories, functions, sizes, relations, and orientations (but NOT image-plane positions).

---
## Output format

### Part 1: Object grounding
For each object:
caption: <object caption>
category: <object category>
orient: <object orientation word>
image-plane position: <...>
appearance/function description: <...>

### Part 2: Functional QA pairs
Output in a JSON-compatible list format for Python (qalist):

qalist = [
 {{
   "id": <int>,
   "question": "<daily-life functional question>",
   "reasoning": "<step-by-step reasoning, including math and spatial logic>",
   "answer": "<final correct answer, with numeric values if applicable>"
 }},
 ...
]

---
OBJECTS:
{objects}
RELATIONS:
{relations}

Output:
```

Figure 4. Prompt used to generate Level-3 spatial problem-solving QA pairs.(test set)

You are an evaluator. Given a Question, a Ground Truth (GT), and a Model Response (Pred), decide whether Pred matches GT.

Step 1 — Determine problem_type:
- If the answer is categorical/boolean/label (e.g., yes/no/true/false, directions like left/right, object names, colors, discrete choices), set problem_type="judgement".
- Otherwise, if it requires a numeric value (distance, height, area, volume, size, time, count, etc.), set problem_type="numeric".

Step 2 — Normalize values:
- For numeric tasks: extract the main scalar that answers the question.
- Evaluate arithmetic expressions if present (e.g., "3*4 cm^{2}", "2 x 1.5 ft^{3}", "0.2 * 3e-1 m").
- Accept approximate symbols (~, ≈, about) or ranges (A–B → use the midpoint).
- Convert to **SI units**:
- For **length**:
  1 inch = 0.0254 m
  1 foot = 0.3048 m
  1 cm = 0.01 m
  1 mm = 0.001 m
  1 m = 1 m
- For **area**:
  1 in^2 = $(0.0254)^2$ m^2
  1 ft^2 = $(0.3048)^2$ m^2
  1 cm^2 = $(0.01)^2$ m^2
  1 mm^2 = $(0.001)^2$ m^2
  1 m^2 = 1 m^2
- For **volume**:
  1 in^3 = $(0.0254)^3$ m^3
  1 ft^3 = $(0.3048)^3$ m^3
  1 cm^3 = $(0.01)^3$ m^3
  1 mm^3 = $(0.001)^3$ m^3
  1 m^3 = 1 m^3
- If unit is unclear, assume meters for linear quantities, m^2 for planar ones (area-related words like “surface”, “floor”), and m^3 for volumetric ones (“volume”, “capacity”, “container”, etc.).

Step 3 — Decide correctness:
- If problem_type="judgement": correct iff GT and Pred are semantically equivalent (case-insensitive, minor wording differences allowed).
- If problem_type="numeric": compute both normalized values (in SI units).
Then relative_error = |response - answer| / max(|answer|, 1e-9).
Correct iff relative_error ≤ 0.25 (25%).

Step 4 — Output JSON ONLY with the following fields:
{{
"problem_type": "judgement | numeric",
"quantity_type": "length | area | volume | other",
"answer_SI_value": "float or null",
"response_SI_value": "<float or null>",
"relative_error": "<float or null>",
"is_correct": "true | false"
"reason": "short explanation of the reasoning process"
}}

Now evaluate:

Question: {question}
GT: {answer}
Pred: {response}

Figure 5. Prompt used to evaluate the correctness of answers in our custom benchmark for spatial problem-solving tasks.

```
referring_prompt_coyo="""
You are an expert in visual description. You will be given an image with a red bounding box highlighting a target object, along with a detailed text description of that object.

Your task is to generate four distinct levels of referring expressions for the object inside the red box.

**Detailed Description of the object:**
{detailed_description}

**--- CRITICAL RULE FOR SPATIAL LOCATION ---**
When describing location, **ONLY use the object's position within the image frame**.
-  **Use terms like**: "at the left", "at the right", "in the center", "at the top", "at the bottom", "top-left", "bottom-right", "leftmost", "rightmost".
-  **DO NOT use object-to-object relations**: Avoid phrases like "on the table", "next to the person", "behind the car".

**Output Requirements:**
Generate a valid JSON object with four keys: "level_1_category", "level_2_property_phrase", "level_3_spatial_phrase", and "level_4_combined_expression". All expressions must be noun phrases and must omit leading articles ('a', 'an', 'the').

* **level_1_category**: A single noun for the basic object type.
   * Examples: "man", "car", "seat".

* **level_2_property_phrase**: A short phrase includes object's type and 1-2 visual properties ONLY (e.g., color, material, state, pattern). **Only object type and visual properties, Do not include any location info here.**
   * Examples: "white SUV", "man in a black jacket", "blue bus seat".

* **level_3_spatial_phrase**: A short phrase describing the object's type and frame-relative location ONLY. **Only object type and location, Do not include any visual properties here.**
   * Examples: "car at the far right", "man at the left", "rightmost seat".

* **level_4_combined_expression**: A more detailed phrase that combines both visual properties (from Level 2) and frame-relative spatial location (from Level 3).
   * Examples: "white SUV parked at the far right", "man in a black jacket looking at his phone at the left".

**Output Examples:**

**Example 1:**
{{
 "level_1_category": "car",
 "level_2_property_phrase": "white SUV",
 "level_3_spatial_phrase": "car at the far right",
 "level_4_combined_expression": "white SUV parked at the far right of the image"
}}

**Example 2:**
{{
 "level_1_category": "man",
 "level_2_property_phrase": "man in a black jacket",
 "level_3_spatial_phrase": "man at the left",
 "level_4_combined_expression": "man in a black jacket looking at his phone at the left"
}}

**Example 3:**
{{
 "level_1_category": "seat",
 "level_2_property_phrase": "blue bus seat with a patterned headrest",
 "level_3_spatial_phrase": "rightmost seat",
 "level_4_combined_expression": "blue bus seat with a patterned headrest on the right side"
}}
"""
```

Figure 6. Prompt used to generate the object reference in kosmos dataset

```
detection_prompt_coyo = """Locate object(s): {referring}, report the bbox coordinates in JSON format.
"""
```

Figure 7. Prompt used to vlm grounding for evaluation in kosmos dataset

```
referring_prompt_o365 = """
You are given an image with a red 2D bounding box highlighting a target object.

There are multiple{class_name} in the image.

Your task is to generate three referring expressions that identify the object inside the red bounding box.

Requirements:
1. First, analyze the object's visual attributes such as:
visual properties (e.g., color, material, state, pattern), position (e.g., left, right, top, bottom), relative location (e.g., left of image, next to another object),
action (e.g., holding, sitting, standing)...
Generate a highly detailed description of the object inside the red bounding box.
2. Then, analyze the detailed description to find which attributes are the most distinctive to differentiate the target object from other similar objects in the
image.
3. Based on the analysis, generate three referring expressions that identify the object inside the red bounding box.
4. The expressions should go from simple to complex:
  - Expression 1: very short and minimal (1~3 words).
  - Expression 2: moderately detailed (5~8 words).
  - Expression 3: highly detailed and unambiguous (at least 8~12 words).
5. All referring expressions must refer to the SAME object inside the red box.
7. Do NOT mention "red box", "bounding box", or any annotation.
8. Do NOT include explanations or extra text.

Output format (exactly):
```json
{{
  "detailed_description": "<Detailed description of visual attributes>",
  "analysis": "<Brief analysis of distinctive attributes>",
  "short": "<Expression 1>",
  "medium": "<Expression 2>",
  "detailed": "<Expression 3>"
}}
"""
```

Figure 8. Prompt used to generate the object reference in objects365 dataset

```
detection_prompt_coyo ="""Locate every instance that belongs to the following reference: {referring_expression}. Report bbox coordinates in JSON format.
"""
```

Figure 9. Prompt used to vlm grounding for evaluation in objects365 dataset
```

```
question_prompt_basic = """
====================
INPUT
====================
You are given:
1) A 3D scene description with multiple objects and their spatial properties.
2) An RGB image of the same scene, where ALL listed objects are already annotated with 2D bounding boxes.
  (The image is only a visual aid; all reasoning must be grounded in the 3D data.)


====================
GOAL
====================


Propose EXACTLY 4 High-level spatially grounded QA.


The generated questions is need to be answered by below **basic spatial information**:
- Knowing the object 3d bbox (object position, size, and orientation)
- Center to center distance between objects (Euclidean distance/ horizontal distance/ vertical distance between objects)
- Object sizes (width, height, depth, volume)
- Occlusion of the objects
- Relative spatial relations (left/right, front/behind, above/below)
- Object orientation (which direction the object is facing towards)
- Object orientation relations (the relation between two objects' orientations)
- Object relation for perspective taking (from the viewpoint of an object, which object is in front/behind/left/right of it)


- and other semantic information inferred from the image and object functions.


The questions must be easy to understand and not ambiguous.
You can ask the question in a senario that related to the given scene, the answer may be easy to answer after you understand the question well.
You can also ask the question that need to combine multiple basic spatial information to answer.
Only one question can be related to the perspective taking.


====================
REQUIREMENTS
====================
- Each question need to be natural in language and be able to understood by humans.
- Do not directly ask for the **basic spatial information**:
- Avoid asking the question that is too hard to answer, such as the question that need to consider the precise shape of the objects.
- Include at least one multiple-choice question (MCQ) with 4 options (A-D), exactly ONE correct.
- Output the answer in JSON format.


{{
  "analysis": "<object functional roles and constraints and reasoning and semantic etc.>",
  "qalist": [
    {{
      "id": 1,
      "question": "..."
    }},
    {{
      "id": 2,
      "question": "..."
    }},
    {{
      "id": 3,
      "question": "..."
    }},
    {{
      "id": 4,
      "question": "...",
      "options": {{
        "A": "...",
        "B": "...",
        "C": "...",
        "D": "..."
      }}
    }}
  ]
}}


objects:
{objects}
"""
```

Figure 10. Prompt used to generate the basic problem solving question

```
question_prompt _quantitative = """
====================
INPUT
====================
You are given:
1) A 3D scene description with multiple objects and their spatial properties.
2) An RGB image of the same scene, where ALL listed objects are already annotated with 2D bounding boxes.
  (The image is only a visual aid; all reasoning must be grounded in the 3D data.)


====================
GOAL
====================


Propose EXACTLY 3 spatially grounded, high-level QA questions that require functional understanding, spatial reasoning, and/or hypothetical thinking in this scene.
You need to ask a quantitative question that requires a **numeric answer**


The generated questions is need to be answered by below **basic spatial information**:
- Knowing the object 3d bbox (object position, size, and orientation)
- Center to center distance between objects (Euclidean distance/ horizontal distance/ vertical distance between objects)
- Object sizes (width, height, depth, volume)
- Occlusion of the objects
- Relative spatial relations (left/right, front/behind, above/below)
- Object orientation (which direction the object is facing towards)
- Object orientation relations (the relation between two objects' orientations)


- and other semantic information inferred from the image and object functions.


Your questions can be asked in a scenario that is related to the given scene, the answer may be easy to answer after you understand the question well.
For the first 2 problems, ask the question in a scenario that is related to the given scene, the answer need to analyze what basic spatial information is needed and do some calculation to get the final numeric answer.


You can also ask the question that need to combine multiple basic spatial information to answer.


====================
REQUIREMENTS
====================
- Each question need to be natural in language and be able to understood by humans.
- Do not directly ask for the **basic spatial information**: size/distance etc. Ask them in a scenario that is related to the given scene.
- Avoid asking the question that is too hard to answer, such as the question that need to consider the precise shape of the objects.
- Output the answer in JSON format.


{{
  "analysis": "<object functional roles and constraints and reasoning and semantic etc.>",
  "qalist": [
    {{
      "id": 1,
      "question": "..."
    }},
    {{
      "id": 2,
      "question": "..."
    }},
    {{
      "id": 3,
      "question": "..."
    }}
  ]
}}


objects:
{objects}
"""
```

Figure 11. Prompt used to generate the quantative problem solving question

```
question_prompt_prior = """
====================
INPUT
====================
You are given:
1) A 3D scene description with multiple objects and their spatial properties.
2) An RGB image of the same scene, where ALL listed objects are already annotated with 2D bounding boxes.
  (The image is only a visual aid; all reasoning must be grounded in the 3D data.)


====================
GOAL
====================


Propose EXACTLY 3 High-level spatially grounded QA.


The generated questions is need to be answered by below **basic spatial information**:
- Knowing the object 3d bbox (object position, size, and orientation)
- Center to center distance between objects (Euclidean distance/ horizontal distance/ vertical distance between objects)
- Object sizes (width, height, depth, volume)
- Occlusion of the objects
- Relative spatial relations (left/right, front/behind, above/below)
- Object orientation (which direction the object is facing towards)
- Object orientation relations (the relation between two objects' orientations)
- Object relation for perspective taking (from the viewpoint of an object, which object is in front/behind/left/right of it)


- and other semantic information inferred from the image and object functions.


The questions must be easy to understand and not ambiguous.


====================
THE TASK
====================
{task_requirements}


====================
REQUIREMENTS
====================
- Each question need to be natural in language and be able to understood by humans.
- Do not directly ask for the **basic spatial information**
- Output the answer in JSON format.


{{
 "analysis": "<object functional roles and constraints and reasoning and semantic etc.>",
 "qalist": [
  {{
   "id": 1,
   "question": "..."
  }},
  {{
   "id": 2,
   "question": "..."
  }},
  {{
   "id": 3,
   "question": "..."
  }}
 ]
}}


objects:
{objects}
"""
```

Figure 12. Prompt used to generate the problem solving question guided by few shot demonstrations

```
MULTI_CLAUSE_REQUIREMENT = """Rquirement: Ask a question with two or more clauses or sub-questions in one utterance.
Such as
Which man is closer to camera? How far are they?
Can a car fit between the two bicycles? If not, what vehicle can be fit in?
Can two person sit on the sofa at the same time? Why?
How should we adjust the camera to take a nice picture of the woman? Should we move a little bit down?"""


EXPLICITLY_CONDITIONED_REQUIREMENT = """Rquirement: Include an explicit condition or constraint (e.g., size/height/value or a scenario) that the
answer must respect.
Such as
Can a 1 meter tall baby sit in the baby buggy?
Can a bowl of 0.5 meters tall fit into the microwave in this picture?
if a person is sitting in the white car, where will he find the man in black top
Can a bowl of o.25 meters tall fit into the microwave in this picture?"""


HYPOTHETICAL_REQUIREMENT = """Rquirement: Ask under a hypothetical assumption using "if" / "suppose" style framing.
Such as
If I want to remove the closest plant, which one should it be?
If a person just finished washing hands, which towel is most conveniently be used?
Which is more confortable for a person to sit on and read book? Consider the lighting condition in the scene.
What is the total distance one need to walk to first reach the table, then the bed starting from the sofa?
If two adults want to sit side-byside on top of the ottoman facing forward, what width does each person have available assuming equal sharing?
"""


INSTRUCTION_SEEKING_REQUIREMENT = """Rquirement: Ask for actionable guidance about what to do (steps or movement), not just a fact.
Such as
How should we adjust the camera to take a nice picture of the woman?
How to make the man to the center of the picture by moving camera?
I want to change camera to take a better photo, what should I do?
How should we adjust the camera to take a whole picture of the woman?
"""


WHY_REQUIREMENT = """Rquirement: Ask a question about spatial, and include "why" at the end.
Such as
Can a person easily pass between the two chairs? Why?
Will the flower still be visible if we move the TV directly in front of it? Why?
Which towel is most conveniently be used after a person finish washing hands? Why?
"""
```

Figure 13. Examples of few-shot demonstration

```
answer_prompt_stage1="""
You are given:
1) An image with ALL relevant objects annotated by 2D bounding boxes.
2) A precise 3D scene description for objects in the image.
3) ONE spatial question about the scene.


Your task (Stage 1 / Teacher):
- Answer the question using the given spatial information and semantic understanding of the scene.
- You MAY use privileged 3D information (3d_box, centers, sizes, orient_front) and perform any necessary computations.
- Besides the final answer, you MUST output a set of BASIC SPATIAL FACTS (spatial_facts) that will be used to train a student model later.
  These spatial_facts MUST contain ONLY FINAL results (no formulas), and MUST follow the allowed types and formats strictly.


Axis conventions: +x right, +y down, +z depth bigger
- x: left (-) to right (+)
- y: top (-) to bottom (+)
- z: near (smaller depth) to far (larger depth)


======================
3D SCENE DESCRIPTION FORMAT
======================
For each object, you are given:
- id
- caption
- category
- 3d_box = [cx, cy, cz, width, height, depth] in meters (camera coordinate system)
- orient_front = [fx, fy, fz] (unit or near-unit vector; facing direction)


Notes:
- cx/cy/cz are box center coordinates relative to camera center
- width/height/depth are object box sizes in meters


======================
REASONING (cot_detailed) REQUIREMENTS
======================
- You MUST produce cot_detailed as a full teacher reasoning trace.
- In cot_detailed you MAY use complex mathematical operations and raw 3D attributes, including:
  distances, AABB edges, dot/cross products, orientation logic, etc.
- cot_detailed should be logically complete and include the necessary spatial reasoning steps.


======================
BASIC SPATIAL FACTS (spatial_facts) REQUIREMENTS
======================
You MUST output "spatial_facts" as a list of BASIC spatial facts for training.


Each fact MUST be a single string with EXACT format:
"<fact_type>: <fact_value>"


General rules:
- You MUST ONLY use the fact types defined below (no new types).
- spatial_facts MUST contain ONLY FINAL results (no formulas, no intermediate steps).
- Numeric values MUST be rounded to 2 decimals.
- Units:
  - width/height/depth/distance are meters (m)
  - volume is cubic meters (m^3)
- For qualitative relations, use clear discrete labels like:
  left/right/front/behind/above/below, occludes/does not occlude, bigger/smaller/same,
  facing toward/away camera, facing left/right, same/opposite/orthogonal directions, etc.
- NEVER include raw privileged geometry tokens inside spatial_facts, such as:
  "3d_box", "cx", "cy", "cz", "sx", "sy", "sz", "orient_front", "[", "]", "dot", "cross", "sqrt", "vector".
  (These tokens can appear in cot_detailed, but MUST NOT appear in spatial_facts.)
```

Figure 14. Prompt used to generate the answer of the given problem solving question (stage1, part1)

ALLOWED FACT TYPES (ONLY these are allowed)
---

1) object_size_width / object_size_height / object_size_depth / object_volume
Meaning:
- Numeric attributes of ONE object.
Format:
- "object_size_width: <obj_id> is <number>m wide"
- "object_size_height: <obj_id> is <number>m tall"
- "object_size_depth: <obj_id> is <number>m deep"
- "object_volume: <obj_id> is <number>m^3"

2) object_orientation
Meaning:
- Qualitative facing direction of ONE object in camera/global frame.
Format:
- "object_orientation: <obj_id> is <orientation_label>"
Allowed orientation_label examples:
- facing toward the camera / facing away from the camera
- facing left / facing right
- facing forward (toward smaller depth) / facing backward (toward larger depth)

3) objects_relative_position
Meaning:
- Qualitative spatial position between TWO objects in the camera coordinate system:
  left/right/front/behind/above/below.
Format:
- "objects_relative_position: <A> is <relation> of <B>"
Where <relation> is one of:
- left of / right of / in front of / behind / above / below

4) objects_orientation_relation
Meaning:
- Qualitative orientation relation between TWO objects:
  facing same direction / facing opposite directions / orthogonal directions.
Format:
- "objects_orientation_relation: <A> and <B> are <relation>"
Few-shot examples:
- "objects_orientation_relation: chair_1 and chair_2 are facing same direction"
- "objects_orientation_relation: car_1 and car_2 are facing opposite directions"
- "objects_orientation_relation: person_1 and bicycle_1 are orthogonal directions"

5) objects_occlusion_relation
Meaning:
- Whether one object occludes another from the camera view.
Format:
- "objects_occlusion_relation: <A> fully occludes <B>"
- "objects_occlusion_relation: <A> partially occludes <B>"
- "objects_occlusion_relation: <A> does not occlude <B>"

6) objects_size_comparison_width / objects_size_comparison_height / objects_size_comparison_depth / objects_size_comparison_volume
Meaning:
- Qualitative size comparison between TWO objects for a specific attribute.
Format:
- "objects_size_comparison_width: <A> is <bigger/smaller/same> than <B>"
(similarly for height/depth/volume)

7) object_perspective_relation_position
Meaning:
- Egocentric relation from the viewpoint of object A using A's own facing direction:
  from A's perspective, where is B (in front/behind/left/right of A).
Use ONLY when the question explicitly says "from A's perspective" or needs egocentric relation.
Format:
- "object_perspective_relation_position: from <A>'s perspective, <B> is <relation>"
Where <relation> is one of:
- in front of A / behind A / to the left of A / to the right of A

Figure 15. Prompt used to generate the answer of the given problem solving question (stage1, part2)

```
Few-shot examples:
- "object_perspective_relation_position: from person's perspective, car is in front of A"
- "object_perspective_relation_position: from chair's perspective, table is to the right of A"
- "object_perspective_relation_position: from robot's perspective, box is behind A"


8) object_perspective_relation_from_to
Meaning:
- Constructed new perspective:
  standing at position of A, facing towards B, then where is C (left/right/in front/behind).
Use ONLY when the question explicitly describes this "stand at A face B" perspective.
Format:
- "object_perspective_relation_from_to: standing at <A> facing <B>, <C> is <relation>"
Where <relation> is one of:
- in front / behind / left / right


9) object_center_distance / object_horizontal_distance / object_vertical_distance
Meaning:
- Numeric distances between centers of TWO objects.
Units: meters (m), rounded to 2 decimals.
Definitions:
- object_center_distance: full 3D center-to-center distance
- object_horizontal_distance: distance in x-z plane (ignore y)
- object_vertical_distance: absolute difference along y only
Format:
- "object_center_distance: <A> and <B> are <number>m apart"
- "object_horizontal_distance: <A> and <B> are <number>m apart horizontally"
- "object_vertical_distance: <A> and <B> are <number>m apart vertically"
Few-shot examples:
- "object_center_distance: person_1 and chair_1 are 1.27m apart"
- "object_horizontal_distance: cup_1 and plate_1 are 0.18m apart horizontally"
- "object_vertical_distance: lamp_1 and desk_1 are 0.62m apart vertically"


------------------------------------------------------------
COVERAGE REQUIREMENT (VERY IMPORTANT)
------------------------------------------------------------
- spatial_facts MUST include ALL facts necessary to answer the question with no missing spatial dependency.
- If the question involves:
  - relative position -> include at least one objects_relative_position
  - perspective wording -> include object_perspective_relation_position or object_perspective_relation_from_to
  - size comparison -> include a relevant objects_size_comparison_* (optionally also numeric object_size_*)
  - distance -> include at least one of object_center_distance/object_horizontal_distance/object_vertical_distance
  - visibility/occlusion -> include objects_occlusion_relation
  - orientation -> include object_orientation and/or objects_orientation_relation


- Provide a minimal but sufficient set: include all needed facts, but do not add many irrelevant facts.


======================
OUTPUT FORMAT (JSON ONLY)
======================
{{
  "question": "<The same question, rephrased only for clarity if needed, but with identical meaning>",
  "cot_detailed": "<Full teacher reasoning. May include exact numeric computation, distances, dot/cross products, and multi-step spatial logic using privileged 3D info.>",
  "answer": "<Final answer>",
  "spatial_facts": [
    "<fact_type>: <fact_value>",
    ...
  ]
}}


Objects:
{objects}


Question:
{question}
"""
```

Figure 16. Prompt used to generate the answer of the given problem solving question (stage1, part3)

```
answer_prompt_stage2 = """
You are performing Stage 2 / Student trace generation.


You are given:
- An image with ALL relevant objects annotated by 2D bounding boxes.
- A question about a scene
- A teacher-provided final answer (ground truth)
- A teacher-provided list of BASIC SPATIAL FACTS (spatial_facts) in a restricted format


Your task:
- Generate a simplified chain-of-thought cot_simple suitable for small models.
- cot_simple MUST follow the same reasoning logic and MUST use the spatial_facts as the ONLY spatial evidence.
- You MUST NOT invent any additional spatial facts. Use ONLY what is given in spatial_facts.
- You MUST keep the final answer identical to the teacher_answer.


======================
cot_simple STRUCTURE (MUST)
======================
cot_simple MUST have exactly 4 steps:


Step1: Plan
- State what spatial facts are needed and what semantic understanding is needed to answer the question.
- Must be a complete logical chain (no missing key links).


Step2: Facts (STRICT)
- Output ONLY the provided spatial_facts, verbatim, as the entire content of Step2.
- No extra sentences, no calculations, no explanation.


Step3: Reasoning with facts
- Use the facts from Step2 to reason about semantics and derive the answer.
- You may do simple arithmetic (+ - * /) with up to 2 decimals if needed.
- You must cite facts explicitly (e.g., "Using objects_relative_position: ...").


Step4: Final answer
- Output the final answer (must match teacher_answer exactly).


======================
HARD CONSTRAINTS (Step2 MUST PASS)
======================
In cot_simple Step2:
- MUST NOT contain any token related to raw 3D geometry or computation.
- MUST NOT contain any of the following tokens (case-insensitive):
  "3d_box", "cx", "cy", "cz", "sx", "sy", "sz", "orient_front", "[", "]", "dot", "cross", "sqrt", "vector"
- MUST NOT contain reasoning words such as "because", "therefore", "so", "thus".
- MUST contain ONLY the given spatial_facts lines.


Self-check & rewrite rule:
- Before producing final output, mentally verify Step2 obeys all rules.
- If any violation is detected, you MUST rewrite cot_simple until Step2 passes.


======================
OUTPUT FORMAT (JSON ONLY)
======================
{{
  "question": "<same question, may rephrase for clarity but identical meaning>",
  "teacher_answer": "<the provided teacher_answer>",
  "cot_free": <the cot process without any constraints, about how to answer the question>,
  "before_cot_simple": "<briefly explain how Step1-Step4 will use the given facts; include a checklist confirming Step2 has no banned tokens and contains only spatial_facts>",
  "cot_simple": "<the 4-step student chain-of-thought>",
  "answer": "<must be identical to teacher_answer>"
}}


INPUT:
question:
{question}


teacher_answer:
{teacher_answer}


spatial_facts:
{spatial_facts}
"""
```

Figure 17. Prompt used to generate the answer of the given problem solving question (stage2)